Algorithmic and theoretical aspects of sparse deep neural networks

Quoc-Tung Le

THÈSE
en vue de l'obtention du grade de Docteur, délivré par
l'ÉCOLE NORMALE SUPERIEURE DE LYON

École Doctorale N°512
École Doctorale en Informatique et Mathématiques de Lyon

Discipline : Informatique

Soutenue publiquement le 20/12/2023, par :

Quoc-Tung Le

Algorithmic and theoretical aspects of sparse deep neural networks

Aspects algorithmiques et théoriques des réseaux de neurones profonds parcimonieux

Devant le jury composé de :

Chaux, Caroline	Directrice de recherche CNRS	Rapporteure
Gillis, Nicolas	Professeur Université de Mons	Rapporteur
Malgouyres, François	Professeur Université Paul Sabatier	Examinateur
Peyré, Gabriel	Directeur de recherche CNRS	Examinateur
Riccietti, Elisa	Maître de conférences Ecole Normale Supérieure de Lyon	Examinatrice
Gribonval, Rémi	Directeur de recherche INRIA	Directeur de thèse

Contents

Abstract

This thesis studies *sparse (deep) neural networks* from an optimization point of view. Nowadays, training large neural networks on massive data sets is a must for many state-of-the-art machine learning models. While this trend is shown to greatly boost the performance for various tasks, it also comes with many prices: demanding computational resources, high carbon emission and high training cost. These downsides are the consequences of the dependence of neural networks on huge architectures and training data sets.

Sparse (deep) neural networks are among approaches which are proposed to mitigate the downsides of conventional neural networks. Their main difference from their classical counterparts is the promotion of sparsity in the parameter space. Recent studies empirically agreed that sparse deep neural networks can offer a compelling practical opportunity to reduce the cost of training, inference and storage, which are growing exponentially in the state-of-the-art of deep learning. Nevertheless, similar to classical neural networks, theoretical study on sparse ones is way behind their empirical advancement.

To advance the theoretical study of sparse (deep) neural networks, we adopt an optimization viewpoint since many sparsity promotion methods can be done via optimization formulations. More specifically, in this thesis, we propose an approach to study sparse (deep) neural networks through the lens of another related problem: sparse matrix factorization, i.e., the problem of approximating a (dense) matrix by the product of (multiple) sparse factors. As shown in Chapter 2, the benefit of studying the problem of sparse matrix factorization is twofold: on the one hand, the sparse matrix factorization is closely related to sparse deep neural networks and a study of the former, which is simpler for mathematical analysis, potentially helps to unveil the mystery of the latter; on the other hand, sparse matrix factorization is an interesting problem of its own and it is also worthy of an independent study. In particular, we identify and investigate in detail some theoretical and algorithmic aspects of a variant of sparse matrix factorization named "fixed support matrix factorization" (FSMF) in which the set of non-zero entries of sparse factors are known. The results obtained from this problem will allow us to address several fundamental questions with regards to sparse deep neural networks.

The contribution of this thesis comprises three main components:

1. Part I - Challenges of (FSMF): This component is presented in two chapters: Chapter 3 and Chapter 4. Chapter 3 will discuss about the non-existence of optimal solution of (FSMF) and its NP-hardness while Chapter 4 studies the general landscape of (FSMF). The main message of the first part is: the problem of sparse deep neural networks training are difficult since many challenges already arise in a simpler

setting - (FSMF).

2. Part II - Tractability of (FSMF): The content of this part is presented in Chapter 5. In a nutshell, we show that several instances of (FSMF) are polynomially solvable and their corresponding landscapes are benign if certain additional structures are assumed.

3. Part III - Applications of (FSMF) to sparse deep neural networks: Using the results obtained from the previous parts, we study some questions in the setting of sparse deep neural networks. In Chapter 6, we study the *butterfly parametrization*, an approach that consists of replacing (large) weight matrices by the products of *extremely sparse* and *structured* ones in sparse deep neural networks. In Chapter 7, the existence of optimal solutions of sparse deep neural networks training problem and topological properties of its function space are studied. Chapter 6 and Chapter 7 use the results of Chapter 5 and Chapter 3, respectively.

Résumé

Cette thèse réalise des études sur des *réseaux de neurones parcimonieux* du pointe de vue d'optimisation. De nos jours, l'entrainement des grands réseaux de neurones sur des jeux de données massifs est un prérequis pour des états de l'art de plusieurs tâches d'apprentissage. Bien que cette tendence est empiriquement vérifiée d'apporter une amélioration significative de performance de modèle d'apprentissage, il nous faut des ressources computationelles massives, des coûts d'entraînement et des taux d'émissions de carbone élevés. Ces inconveniences sont des conséquences directes de la dépendence de la réussite des réseaux de neurones aux larges architectures et aux jeux de données massifs.

Des réseaux de neurones parcimonieux est l'un des approches qui sont proposées pour alléger les points faibles des réseaux de neurones classiques. Leur difference par rapport aux homologues classiques est la promotion de la parcimonie dans leurs espaces de paramètres. Des études récentes ont aussi montré empiriquement que les réseaux de neurones parcimonieux peut nous offrir la réduction de coût d'entraînement, d'inférence et de stockage, dont croissances sont exponentielles dans les états de l'art de l'apprentissage profond (deep learning). Néanmoins, à l'instar des réseaux neurones classiques, les études théoriques sur les réseaux de neurones paricimonieux sont beaucoup moins avancées que leurs études experimentales.

Notre étude axée sur la théorie des réseaux de neurones est basée sur un point de vue d'optimisation car la promotion de la parcimonie de plusieurs méthodes est réalisée par des formulations d'optimisation. En détail, dans cette thèse, nous proposons une approche dont point de départ est un problème liés mais plus simple: la factorisation de matrices sous des constraints de parcimonie (ou simplement *factorisation parcimonieuse de matrices*), c'est à dire, d'approximer une matrice (potentiellement dense) par des produits des facteurs parcimonieux. Nous montrons dans Chapitre 2 deux avantages d'une étude sur le problème de factorisation parcimonieuse de matrices: d'une part, la factorisation parcimonieuse de matrices est très lié aux l'entrainement des réseaux de neurones parcimonieux et son études, qui est plus simple pour l'analyse mathématique, potentiellement nous donne des informations intéressantes des réseaux de neurones parcimonieux; d'autre part, la factorisation parcimonieuse de matrices est déjà intéréssante ele-même, avec plusieurs possible d'applications. En particulière, nous identifions et faisons la recherche sur quelques aspects théoriques et algorithmiques d'une variante de la factorisation parcimonieuse de matrices, nommé *Fixed support matrix factorization* - FSMF - (ou *la factorisation de matrices sous des constraints de supports fixés des facteurs* en français) dans lequel les ensembles de coefficients non-nuls des facteurs sont accessibles. Les résultats de ce problème nous permets d'addresser quelques questions fondamentales par rapport aux réseaux de neurones parcimonieux.

La contribution de cette thèse comprend trois parties principales:

1. Partie I - Des défis de (FSMF): Cette partie est présenté en deux chapitres: Chapitres 3 et 4. Alors que Chapitre 3 porte des études sur le non-existence d'une solution optimale de (FSMF) et sa characterisation de NP-difficulté, Chapitre 4 présente nos résultats sur le paysage général de la fonction de côut de (FSMF). Le message principal de la première partie est: le problème de l'entrainement des réseaux de neurones parcimonieux est difficile car il y avait plusieurs défis dans un setting plus simple: (FSMF).

2. Partie II - La tractabilité de (FSMF): Le contenu de cette partie est présenté dans Chapitre 5. D'une manière générale, nous montrons que si certaines hypothèses sont satisfaites par les constraints de supports de facteurs, (FSMF) est tractable. De plus, le paysage de sa fonction de coût devient benign.

3. Partie III - Les applications de (FSMF) aux réseaux de neurones parcimonieux: En utilisant les résultats de deux première parties, nous faisons des études sur quelques questions par rapport aux réseaux de neurones paricmonieux. Dans Chapitre 6, nous discutons le *butterfly parameterization*, une approche qui remplace des (grandes) matrices de poids par des produits des facteurs *extrêmement parcimonieux* et *structurés* dans des réseaux de neurones parcimonieux. Dans Chapitre 7, l'existence d'une solution optimale du problème de l'entrainement des réseaux de neurones parcimonieux et des propriétés topologiques de son espace fonctionelle sont étudiés. Chapitre 6 et Chapitre 7 utilisent les résultats de Chapitre 5 et Chapitre 3 respectivement.

Notation

Common	
$[\![n]\!]$	Set $\{1, \dots, n\}$.
$\mathbb{N}$	the set of natural numbers
$\mathbb{Z}$	the set of integer numbers
$\mathbb{Q}$	the set of rational numbers
$\mathbb{R}$	the set of real numbers
$\mathbb{C}$	the set of complex numbers
Spaces	
$C^0(\Omega)$	The set of continuous function from $\Omega \subseteq \mathbb{R}^i \mapsto \mathbb{R}^o$ (the dimension i and o are clear from the context)
$L^p_\mu(\Omega)$	The L^p space w.r.t a measure μ of $\Omega \subseteq \mathbb{R}^d$
$L^p(\Omega)$	The L^p space w.r.t the Lebesgue measure of $\Omega \subseteq \mathbb{R}^d$
$W^{k,p}(\Omega)$	The Sobolev space with the index (k, p) on $\Omega \subseteq \mathbb{R}^d$
Vectors	
$\mathbf{0}_m$	An all-zero vector of size m
$\mathbf{1}_m$	An all-one vector of size m
Matrices	(all notations assume that $\mathbf{W} \in \mathbb{R}^{m \times n}$)
$\mathbf{0}$	An all-zero matrix with appropriate size
$\mathbf{0}_{m \times n}$	An all-zero matrix with size $m \times n$
$\mathbf{1}$	An all-one matrix with appropriate size
$\mathbf{1}_{m \times n}$	An all-one matrix with size $m \times n$
$\text{supp}(\mathbf{W})$	The support of the matrix $\mathbf{W}$
$\|\mathbf{W}\|_F$	The Frobenius norm of the matrix $\mathbf{W}$
$\mathbf{W}[:, i]$	The i column of the matrix $\mathbf{W}$
$\mathbf{W}[i, :]$	The i row of the matrix $\mathbf{W}$
$\mathbf{W}[:, S_c]$	For $S_c \subseteq [\![n]\!]$, this is a matrix of the same size as $\mathbf{W}$, which agrees with $\mathbf{W}$ on columns in S_c. Other coefficients are equal to zero.
$\mathbf{W}[S_r, :]$	Similar to $\mathbf{W}[:, S_c]$, but for rows instead of columns
$\mathbf{W}[\bullet, S_c]$	For $S_c \subseteq [\![n]\!]$, this is a matrix of size $m \times \|S_c\|$, which is equal to the submatrix of $\mathbf{W}$ restricted to the columns indexed in S_c.
$\mathbf{W}[S_r, \bullet]$	Similar to $\mathbf{W}[\bullet, S_c]$, but for rows instead of columns
$\mathbf{W}[S_r, S_c]$	For $S_r \subseteq [\![m]\!], S_c \subseteq [\![n]\!]$, this is a matrix of size $\|S_r\| \times \|S_c\|$, and equal to the sub-matrix of $\mathbf{W}$ restricted to rows and columns indexed in S_r and S_c respectively

Chapter 1

Introduction

The 2010s witness one of the most impressive comebacks in computer science: the re-emergence of neural networks (NNs). Ever since their introduction in 1943 [MP43], their current dominance over the fields of Machine Learning (ML) in particular and Artificial Intelligence (AI) in general is unprecedented. Indeed, most recent advances and important applications in AI involve NNs and its subfield *deep learning*. Figure 1.1 shows several AI breakthroughs in the last ten years. It is not hard for us to notice that the impetus behind most (if not all) of these milestones are NNs.

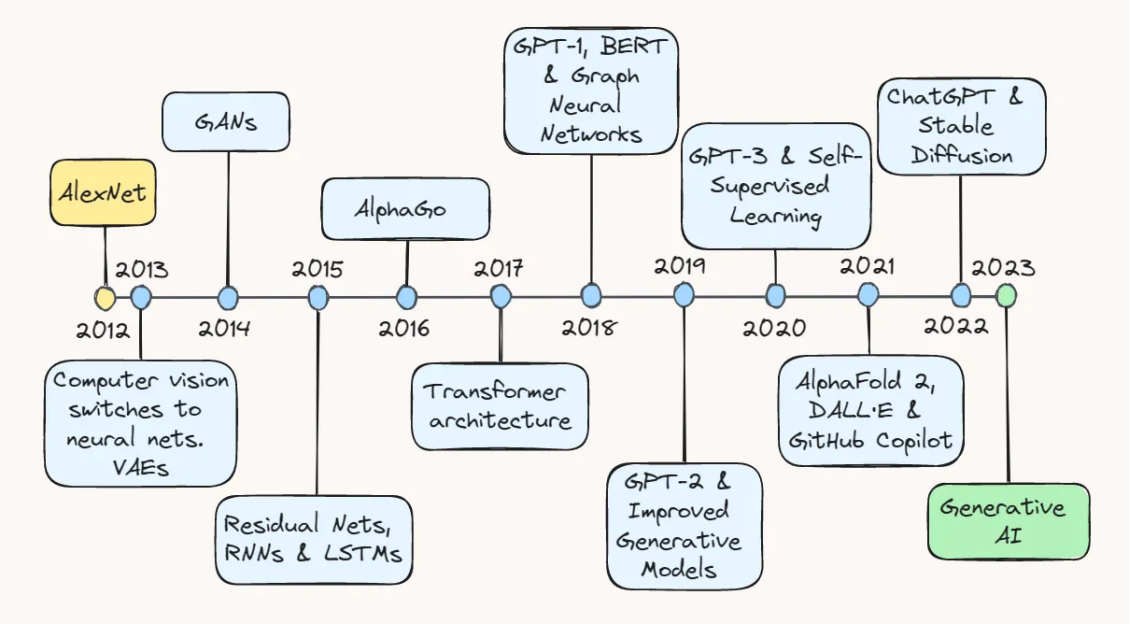


Figure 1.1: Some milestones of AI in 2010s. This image is taken from a blog post of the website `https://towardsdatascience.com` [Dor23].

Why does the performance of NNs become so good nowadays? Let me remind the readers that NNs in particular and AI in general witnessed two periods of the so-called "AI winter": the first in the 1970s and the second in the 1990s where AI research (including studies of NNs) stopped progressing due to the lack of funding and the pessimistic views

from both public and research community (see the Lighthill report [Lig73]). These periods were caused by the *failure* of NNs and AI on excelling in many important problems such as machine translation, image recognition and decision making despite their initial hype. Ironically, the same set of problems is what NNs are known for nowadays. What are the main differences/progress between NNs now and then? Aside from the efforts of our research community, the following reasons also played important roles:

1. **Computing powers for large models**: Progress on computer is vital for the development of NNs. Computing powers nowadays become more and more powerful in all aspects such as memory, RAM, microprocessors. That allows us to perform the training/learning of large NNs, an indispensable component of many state-of-the-art NN models. Moreover, the adoption of Graphical Processing Units (GPUs) into the pipeline of NNs training is also worth mentioning. These units are well-designed to efficiently perform heavy computational tasks such as matrix-matrix/matrix-vector multiplication thanks to their capability of parallelization, fast memory access and large memory bandwidth. As such, they are able to greatly accelerate the training and inference of NNs, thus allowing the use of increasingly larger NNs.

2. **Availability of large datasets**: Another factor adding to the raise of NNs is the abundance of data. In the early age of AI, data was hard to obtain. Models were trained/learned typically with datasets of hundreds to thousands of samples. These numbers fail today's standards in which popular training sets might contain up to billions of data points. It is not a coincidence that the ImageNet dataset [DDS+09] is often accounted for the re-emergence of NNs. This dataset and its associated competition - ImageNet Large Scale Visual Recognition Challenge (ILSVRC), is an important event where NNs first showed its superior performance in comparison to other Machine Learning (ML) methods.

These two factors remain essential to the building of state-of-the-art NN models. Despite being the main impetus behind many recent advances of NNs, the dependence on extensive computing powers for large models and large available datasets of NNs also exerts certain negative impact on our society. Here is a list of undesirable effects of current research on NNs:

1. **Carbon emission**: It is reported in [SGM20] that the amount of carbon emitted by the training of a Large Language Model (LLM) such as a $\mathbf{T2T}_{\text{big}}$ - the Transformer big model [VSP+17] is five times that of a car during its entire lifetime [1]. This is due to the huge energy consumption of supercomputing clusters to train large NNs. It is worth mentioning that this estimation is only for a single training. Research and implementation of deep learning typically requires careful tuning of hyper-parameters and model selections. This implies that the amount of emitted carbon for certain research papers on deep learning can be much higher in practice.

2. **Training cost**: In the same article [SGM20], the training costs of several popular LLMs are also estimated. Among the most expensive LLMs are Transformer

[1] To compute the amount of emitted carbon, authors in [SGM20] uses a conversion between consumed energy and emitted carbon

[VSP+17], ELMo [PNI+18], BERT [DCLT19], GPT-2 [RWC+19] and NAS [SLL19]. Their training costs are shown in Table 1.1. These are the price of cloud computing services to train the models. It is surprising that the cost of one single training of NAS (maximum is $146,848) is already three times larger than my three-year Ph.D. net salary at ENS de Lyon. More importantly, these amounts are gigantic and often unavailable to most academic researchers in the field. That also raises the question of equality on the computational resources access among researchers.

Model	Training Hardware	Training cost (in dollar $)	Original paper
Transformer	P100 x 8	289 - 981	[VSP+17]
ELMo	P100 x 3	433 - 1,472	[PNI+18]
BERT	V100 x 64	2,074 - 6,912	[DCLT19]
GPT-2	TPUv3 x 32	12,902 - 43,008	[RWC+19]
NAS	TPUv2 x 1	44,055 - 146,848	[SLL19]

Table 1.1: Training cost of the most expensive LLMs (extracted from [SGM20, Table 3]). TPU stands for Tensor Processing Unit, a processor that is designed specifically for the calculation and manipulation of tensors.

3. **Annotations of data**: As presented in the previous paragraph, massive datasets are necessary for the success of any deep learning model. However, it is worth emphasizing that size alone is not sufficient. Current machine learning techniques rely heavily on *annotated* or *labelled* data. For example, to perform classification, one needs datasets of images and their correct classes. The more complex is the task, the more complex is the annotation procedure. More importantly, the current annotation procedures are carried out entirely by humans. Thus, the cost and the working hours to produce a large, *high-quality* training set is huge. To lower this cost, certain unmoral practices have been used. It is reported in the Time magazine [Per23] that Kenyan workers were paid less than $2 per hour by OpenAI to make chatGPT less toxic. In fact, these workers have to witness *horrible* contents such as "child abuse, murder, suicide, self-harm and incest" taken from dark websites and they are asked to classify these contents correspondingly.

Up to this point, we wish readers are convinced that it is important to *reduce the dependence of deep learning on large NNs and its data-hungry nature*. Many approaches have been proposed to reduce each dependency *separately*. Among the most prominent approaches to perform *only* model compression are value quantization [SJG+20, GMR23, GBGR23], value compression [HMD16] or model downsizing [HVD15]. Others seek to reduce the number of labelled training data by using semi-supervised [YSKX22] or unsupervised learning [DZDW18]. Nevertheless, in this thesis, our focus is on sparse deep neural networks - an attempt from the research community to reduce both dependencies simultaneously and mitigate the negative effects of NNs on our society. As such, this chapter is devoted to present the concept of sparse deep neural networks and the needs for its study. Then, we discuss the main contributions of this thesis.

1.1 Sparse deep neural networks: An overview

In this section, our goal is to describe what a sparse deep neural network (DNN) is and which advantages it can offer. We place emphasis on deep models because they are the most popular architectures and they are in the regime where many of the pitfalls mentioned above can arise. As suggested by its name, the sparse DNN is a variant of the classical DNN whose parameters are imposed to satisfy certain sparsity constraints, i.e., they have many zero entries. A conceptual illustration of DNNs and the sparse version is given in Figure 1.2.

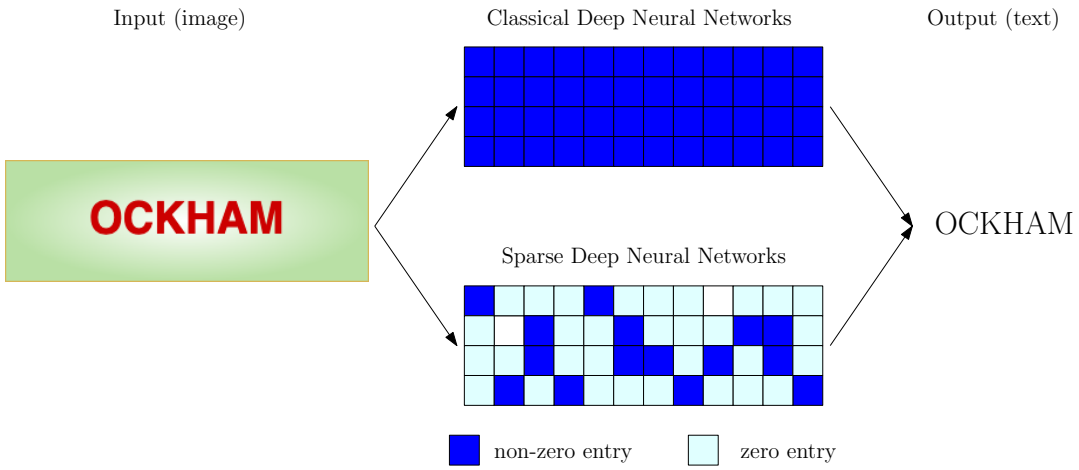


Figure 1.2: Comparison between DNNs and sparse DNNs for text extraction problem. In this figure, a square represents a parameter of DNNs. Blue and white squares indicate parameters with non-zero and zero values respectively. Differently from classical DNNs, sparse DNNs has many zero parameters.

In our opinion, it is very important to talk about the conventional/classical DNNs as well as their strengths and weaknesses. This is exhibited in Section 1.1.1. Such knowledge is essential to understand the idea behind sparse DNNs. More specifically, in Section 1.1.2, readers will see that sparse deep NNs do not only inherit many strong points of their classical counterparts but they also offer a potential remedy for the discussed shortcomings theoretically. On the practical side, we also show some positive empirical results with regards to sparse DNNs. Several challenges and questions of sparse DNNs are discussed at the end of this section.

1.1.1 Deep neural networks: strengths and shortcomings

We start by introducing classical neural networks. For the sake of simplicity, we avoid formal definition of NNs, which will be then provided in Chapter 2. In this introduction, it is sufficient to understand that NNs are (huge) parametric models that encode functions. The input/output and purpose of these functions vary, depending on the machine learning task that we wish to study. For example, one would like to build a model that is capable of extracting sequences of text from images. In that case, the input is an image, the output

is a sequence of text and the function needs to produce exactly the sequence written in the image. This example is illustrated in Figure 1.2.

The next question that we need to address is: How can we assign the parameters of the NNs so that its encoded function can assign the inputs to their correct outputs correctly (with high probability)? This is where data steps in. Given a training data set whose elements are of the form (x, y) (where x is an input, such as image, and y is the corresponding label, such as a text sequence as in the previous example), ideally, we wish to select parameters of the NNs so that its corresponding function f satisfies $f(x) = y$ for every pair in the data set. In practice, it might not be possible to find a set of parameters to ensure the equality for all $f(x)$ and y. Therefore, it is more practical to search for parameters that make $\ell(f(x), y)$ small where ℓ is a function measuring the "distance" between $f(x)$ and y. The function ℓ is also known as *loss function*.

With this high-level description, it is easy to see that NNs possess many strengths:

1. **Flexibility of model choices**: There are a plethora of NN models such as Multi-Layer Perceptron (MLP) [Whi63], Recurrent Neural Networks (RNN) [RM87], Convolutional Neural Networks (CNN) [LB98], Attention mechanism [VSP+17], to name but a few. This is also how we can incorporate our expert knowledge into the models to customize the architecture to adapt to the problems as well as improve the performance.

2. **Generic framework**: Once an architecture is fixed, fitting its parameters to a given data set only requires a differentiable loss function ℓ. The parameters will be optimized automatically via many available optimization algorithms. This framework is very general and can be used in many different settings.

3. **Excellence for large datasets**: NNs and DNNs outplay other ML algorithms and models in the regime of large datasets (for example, the ImageNet dataset with more than $14,000,000$ images and 1000 different objects).

Nevertheless, the high performance of DNNs also comes with several downsides.

1. **Model size**: State-of-the-art deep learning models typically have from millions to billions of parameters [VSP+17, PNI+18, DCLT19, RWC+19] and new models tend to further grow these numbers. This tendency can be observed in Figure 1.3. Thus, they can only run (i.e., compute the function f corresponding to its parameters) efficiently on solid hardware equipped with GPUs. This limitation prevents their applications to mobile or embedded devices where extensive computational resources are usually not available. Moreover, even if one does not run into the computing resources problem, the cost of using deep learning to real-world applications is huge and only accessible to large companies and start-ups with huge funds.

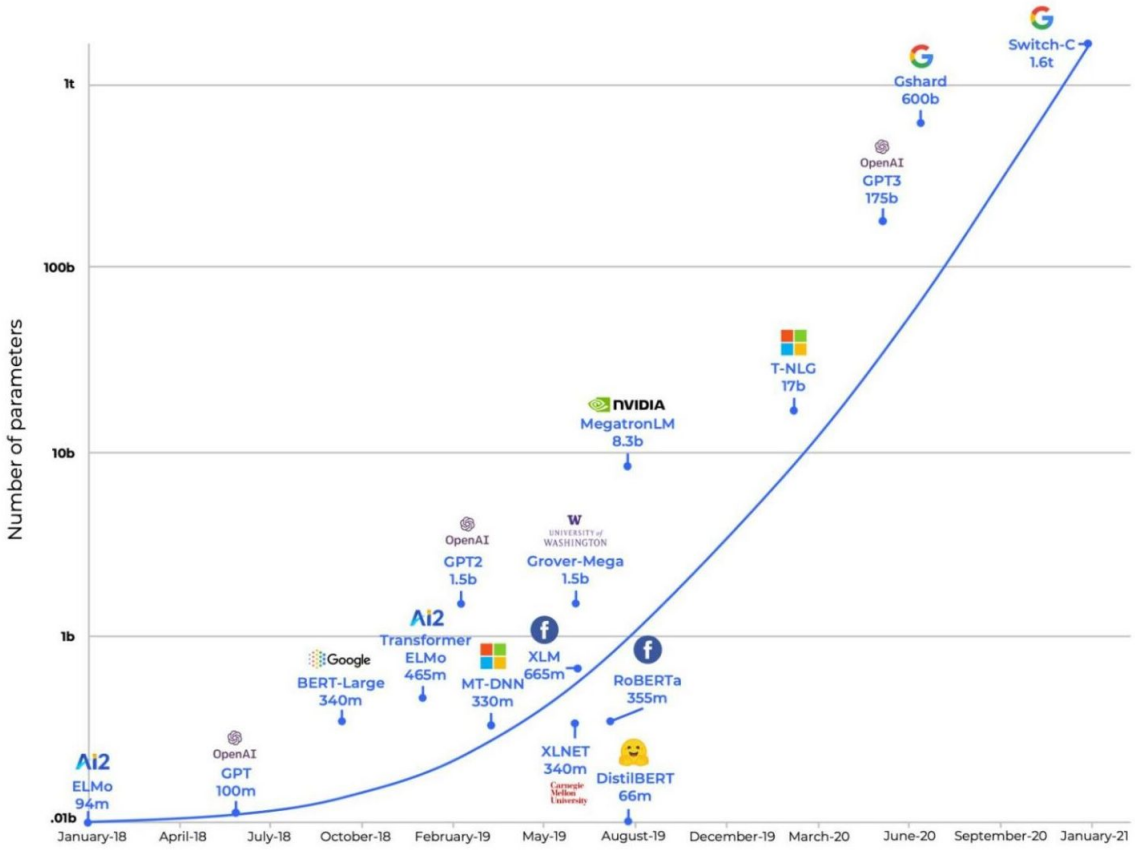


Figure 1.3: Growth of the number of parameters between 2018 - 2021. The image is taken from a blog post in `https://deci.ai/` [Use23].

2. **Massive training data**: DNNs are always hungry for training data to make them generalize well on unseen instances. However, as evoked in the precedent paragraph, a large, high-quality training dataset is costly and usually unavailable for most academic institutes and medium to small-sized enterprises. Thus, the development of DNNs is fairly restricted to/ under the control of huge corporations.

The first and the second points are direct results of the dependence of deep learning on large models and datasets, which we already discussed in the precedent paragraphs. Moreover, there exist other indirect consequences, which are listed below:

3. **Privacy issue:** Performant NNs are typically obtained in the *over-parameterization* regime [BHMM19], i.e., when the number of parameters of NNs is (much) larger than the size of the training set. In fact, the parameters of NNs can then often be chosen so that the corresponding function fits perfectly the training data. This claim is shown theoretically in [AZLS19] (under certain assumptions on the dataset and the size of NNs) and empirically in [ZBH$^+$21]. Therefore, intuitively, a sample whose output from a trained overparameterized NN is *too good*, i.e., close to its real output is likely to belong to the training set of the used NN. This is the intuition for the so-called Membership Inference Attack (MIA) [NSH18, SSSS, HSS$^+$22], which is

capable of identifying samples from training datasets given black-box queries of a NN. With deep learning being increasingly implemented in many services, there is a privacy risk if one can exploit the knowledge leaked from trained NNs.

4. **Lack of interpretability:** While NNs perform well on many learning tasks, its prediction behavior is hardly interpretable from a human point of view. A famous story justifying this claim is the existence of adversarial example [SZS+13]. The main idea in [SZS+13] is that a slight change in the input space (such as setting *one* pixel of an image to zero) can lead to a drastic change of the output. In the setting of image classification problem, many performant DNNs such as AlexNet [KSH12], VGG [SZ15], GoogleNet [SLJ+15], ResNet-152 [HZRS16] can correctly classify a given image but their predictions change if we slightly modify the same image (by adversarially adding noise, or modifying several pixels). In contrast, under these modifications, human vision hardly notices any difference [SZS+13, MDFFF17]. This counter-intuitive behavior shows a huge difference betwenn human and machine perception. In particular, when deploying DNNs to applications with high-stakes predictions such as autonomous driving and health care, the lack of interpretability might be a severe issue.

All of these issues call for a remedy. However, the alternatives should also retain the advantages of NNs since these features are crucial to applicability. This is where sparse DNNs come in handy.

1.1.2 Sparsity in deep neural networks

As stated earlier in this chapter, sparse DNNs is a variant of the usual DNNs whose parameters are sparse, i.e., there are many zero parameters [2]. Thus, it is sufficient to store non-zero ones and their location since zero parameters can be ignored when computing the output of a NN. On the one hand, sparse DNNs inherit many advantages of conventional NNs such as the flexibility of model choices or the genericity of framework because they are simply built on NNs. In the following, we argue that if we can indeed promote the sparsity in the space of parameters of NNs, all the shortcomings listed in Section 1.1.1 are expected to be addressed:

1. **Model compression**: Due to the sparsity in the NNs parameters, the *actual* size of deep learning models can be reduced significantly (as mentioned previously, only non-zero coefficients are needed to evaluate the corresponding parametric functions).

2. **Training data diet**: Since the actual model size is reduced with this sparsity approach, it is intuitively less hungry for data. We justify this intuition by drawing a link with the linear inverse problems [Tib96, BT09] where sparsity is also employed. These two approaches are very similar since their ultimate goal is to find a small subset of all parameters to have non-zero values (the remain will be set to zero). In the setting of linear inverse problems, if the model is known to be sparse in priori, this approach is known to boost the performance, especially when the number of

[2] A mathematical definition of sparse DNNs will have to wait until Chapter 2 since this chapter is designed to be as simple as possible. We also believe that the current definition provides sufficient background at this stage

observations is smaller than the total number of parameters. We expect that a similar phenomenon holds for NNs since there exist certain hypothesis about sparsity in DNNs such as Lottery Ticket Hypothesis [FC19].

3. **Reduction of privacy leakage**: It is shown in a joint work between Ph.D. students in our laboratory (Antoine Gonon [3], Léon Zheng [4], Clément Lalanne [5], Guillaume Lauga [6], Can Poulinquen [7], and myself) that sparsity can reduce the vulnerability of NNs against Membership Inference Attack (MIA) [GZL+23]. Our intuition is very simple: Sparsity forces the model out of the over-parameterization regime where memorization of training data can easily happen. Thus, the information about the training set revealed from the output of trained NNs is reduced. Black-box attacks such as MIA are thus, nullified. Our work [GZL+23] empirically verifies this intuition.

4. **Improvement of interpretability**? It is not clear whether sparsity can help us better understand the behaviour of DNNs. Our only justification for this claim is the similarity between sparse DNNs and dictionary learning and/or sparse coding [Mai10, MBPS09, MBPS10, MBP+08]. In general, these methods search for elementary elements such that samples can be written as their sparse linear combinations. This idea is based on the observation that *natural* data (such as images, documents, speech) usually has such sparse representations. For example, sentences and paragraphs in documents are composed of a small subset of words from a (large) dictionary. That explains the name "dictionary learning". Similar to words in a dictionary that is fathomable for human beings, one expects that the models with sparsity will reveal understandable components, hence, make them more interpretable.

Therefore, on a theoretical side, sparse DNNs are seemingly an upgraded version which not only possesses many strengths but also (partially) addresses the difficulties of conventional DNNs. On the practical side, we quote the conclusion from [HABN+21, page 2]: *Today's sparsification methods can lead to a* $10-100x$ *reduction in model size, and to corresponding theoretical gains in computational, storage, and energy efficiency, all without significant loss of accuracy.* To demonstrate this claim, we present a set of results of algorithms for sparse DNNs. These methods are applied to ResNet-50 [HZRS16] and outputs sparse DNNs (based on ResNet-50) with approximately ten times fewer parameters (or equivalently, with 90% sparsity)[8]. The performance of these models is measured by the probability of predicting correct labels of images (accuracy) in the test sets of ImageNet [DDS+09] (remind that ResNet-50 is designed to perform image classification tasks and it is also trained with the ImageNet dataset). This measurement is reported in Table 1.2. It can be seen that a sparse version of NNs is about to catch up to the performance of the original while reducing their models size significantly. We believe that there are still rooms for improvement with sparse DNNs and they are worthy of a study.

[3] https://agonon.github.io/
[4] https://leonzheng2.github.io/
[5] https://clemlal.github.io/
[6] https://laugaguillaume.github.io/
[7] https://perceptronium.github.io/
[8] A zero vector is 100% sparse, while a dense one is 0% sparse

Methods	Sparsity level (%)	Accuracy	Original paper
Dense baseline	0	77.1	[HZRS16]
STR	90,23	74.31	[KRS+20]
RigL	90	72.0	[EGM+20]
DNW	90	74.0	[WFR19]
GMP	90	73.91	[ZG17]
GraNet	90	74.5	[LCC+21]
ProbMask	90	74.68	[ZZXZ21]
ST-3$^\sigma$	90	76.03	[VV22]

Table 1.2: Results are taken from [VV22, Table 3]. Authors reproduce the results on ResNet-50 with sparse DNNs training algorithms.

1.1.3 Sparsity in deep neural network: some challenges

While being theoretically interesting and empirically promising, sparse DNNs and its study face many challenges. There remain many open (and challenging) questions such as the learning dynamics or the generalization of sparse DNNs in comparison to the dense counterparts. Nevertheless, in this thesis, we take a step back and consider the training problem of sparse DNNs from an optimization viewpoint. In our opinion, a study on the optimization problem of sparse DNNs is the most fundamental because sparsity in DNNs is promoted via the design of the objective functions of the optimization problem and/or the optimization algorithms. Other aspects such as learning dynamics or generalization capability are dependent on these choices. To this end, we are interested in the following list of questions:

1. **Existence of optimal solutions**: Does the optimization problem corresponding to the training of sparse DNNs *always* admit an optimal solution? If it is not *always* the case, given a training formulation, can we decide whether an optimal solution exists?

2. **Provable training of sparse DNNs**: If the training formulation has an optimal solution, does there exist a (tractable) algorithm with provable guarantee?

3. **Landscape of objective functions in sparse DNNs training**: Training both classical and sparse DNNs involves solving a non-convex optimization problem. Therefore, critical points of the objectives function are not bound to be local minima (let alone global ones). However, vanilla first-order methods such as variants of Gradient Descent and their stochastic version seem to be work well in practice [GBC16, Chapter 8]. Does the landscape have any special property that allows these methods to perform?

These questions serve as the motivation for our study. It is noteworthy that these questions were already raised and partially addressed in the setting of classical DNNs (for example, the first question - [LMQ21], the second question - [CB18] and the third question - [Kaw16, VBB20]). Our goal is to generalize existing results to a new setting: sparse DNNs. In the next section, we present the organization of this thesis.

1.2 Contributions

This thesis is based on four research articles: [LRG22, LZRG22, LRG23] and another paper that has not been finished yet. Since the thesis is a continuation of my M2 internship, certain materials in Chapter 2 are also taken from my report [Le20] and its conference version [LG21]. These materials serve as motivations or empirical verification of intuitions and do not count as contributions. The technical parts of this thesis (from Chapter 3 to Chapter 7) are composed of five chapters. Chapter 2 only introduces the problems/questions of interest and provides a literature review for this thesis. The main content of each chapter is given below:

Chapter 2: From sparse deep neural networks to sparse matrix factorization In this chapter, we first provide the formalism of conventional and sparse DNNs as well as their corresponding training problems. Since there is many different types of DNNs, it is worth emphasizing that our focus is on the Multi-layer Perceptron [Whi63] and its sparse variant since it is not only one of the most general models but it is also the *de-facto* model for theoretical study. This is followed by an overview on the research state of sparse DNNs: We discuss several prominent algorithmic and theoretical approaches to contextualize our study.

Sparse DNNs are generally difficult to study since their definition involves both the notions of non-linearity and sparsity. To avoid dealing with two difficulties at the same time, our approach is to simplify the model by dropping the notion of non-linearity in sparse DNNs. This results into the *sparse matrix factorization problem*. In words, this problem seeks to find sparse matrices whose product approximates well a given matrix. As will be shown in Section 2.3, its study is not only beneficial to our study of sparse DNNs but also interesting of its own since sparse matrix factorization problem has many applications in practice such as linear operator acceleration or dictionary learning.

Finally, we further consider a particular case of sparse matrix factorization named "Fixed support matrix factorization" (FSMF) in Section 2.4. This is the most simplified setting where we study in detail and the results will be then used to answer certain questions on sparse DNNs and sparse matrix factorization in Chapter 6 and Chapter 7.

Chapter 3: Existence of optimal solutions and NP-hardness of (FSMF) Once formulating (FSMF) as an optimization problem, we study the first natural question: Does it always admit an optimal solution? This question is previously investigated in other settings such as tensor decomposition problems, robust principal component analysis or even classical NNs as well. The answer for this question is *negative*, i.e., there are instances whose optimal solution does not exist.

Furthermore, in Section 3.2, we go further to answer a more involved question: Under which conditions does an instance of (FSMF) admit at least an optimal solution? The investigation of this question directs us to algorithms of real algebraic geometry, a mathematical domain that deals with polynomials and semi-algebraic sets. The usage of its tools gives an algorithmic answer to our questions.

This last main results of this chapter is the NP-hardness of (FSMF), presented in Section 3.3. This is a surprising result because it implies that *even with the knowledge of support factors, finding the optimal coefficients remains hard*. Our work also shows

the relation between (FSMF) and the matrix completion problem. The main result on the NP-hardness is just an immediate corollary of this relation since matrix completion is already shown to be NP-hard in previous work [GG10].

Chapter 4: General landscape of (FSMF) (FSMF) is a non-convex optimization problem. However, recent studies on the landscape [9] of non-convex optimization problems reveal several cases where the objective functions possess certain properties of convex functions. For example, many variants or problems related to sparse matrix factorization such as matrix completion, phase retrieval or even neural network training problem have objective functions whose local minima are all also global ones [CLC19]. Our work shows that the landscape of a general instance of (FSMF) can have *spurious local minima* and *spurious local valleys*. These objects are undesirable for global optimization.

Chapter 5: (FSMF) with structure is tractable Despite three *bad* news about (FSMF) that are the non-existence of optimal solution, NP-hardness and *difficult* optimization landscape shown in Chapters 3 and 4 respectively, support constraints with structures can make (FSMF) easier to handle. In this chapter, we introduce a family of support constraints whose instances are polynomially tractable (with at least an optimal solution) and such that the landscape of the objective function is *benign*. This structure can be found in various instances of (FSMF) in the literature such as low rank matrix approximation [EY36] or hierarchical $\mathcal{H}$-matrix approximation [BK16, Hac99, HK00].

Chapter 6: Butterfly parameterization in sparse deep neural networks An application of our results on (FSMF): an analysis of butterfly parameterization in deep learning [DGE+19, DSG+20, LRC+21, DCS+22b]. This line of work attempts to replace the linear operators in deep learning by ones which can be factorized into multiple sparse linear operators (sparse matrices/factors). This approach is empirically reported to perform well and reduce the size of NNs significantly [DGE+19, DSG+20, LRC+21, DCS+22b]. Our study first reveals that this approach simply replace the set of hypothesis matrices from the usual $\mathbb{R}^{m\times n}$ to the so-called *low-rank complementary* matrices. This set has an explicit algebraic description which is based on the low rank of sub-matrices. Moreover, we also prove that the optimization problem that projects a given matrix onto the set of low-rank complementary matrices has an approximate algorithm (i.e., the distance from solution obtained by the algorithm to the given matrix is no worse than that of the optimal one, up to a constant factor). This is based on an analysis of a modified version of the classical *butterfly* algorithm. This work is in collaboration with Léon Zheng[10], a PhD student in the same laboratory.

Chapter 7: Existence of optimal solutions in sparse deep neural networks training Another application of our results on (FSMF) to the world of sparse DNNs is an investigation of the existence of optimal solutions for the training problem of sparse ReLU DNNs, which is one of the most popular architecture of sparse DNNs. More specifically, the

[9] In this context, the landscape of a function refers to its global geometry over the entire parameters space. This line of study usually investigates the properties of local minima and/or saddle points.

[10] https://leonzheng2.github.io/

main question in this chapter is: for which NNs architecture does the optimal solution of the training problem always exist? Using the analysis of the *fixed support* version, we show that there exist several architecture of sparse ReLU DNNs that make its training problem always have optimal solutions. Nevertheless, certain architectures fail this property and we provide a criterion and examples for such architectures. Our technique can also be extended to address questions on the closedness of the function space of sparse ReLU DNNs. This allows us to generalize existing results [PRV21] on topological properties of the function space of NNs.

In conclusion, the table below presents some of my publications during three-year Ph.D. as well as their informations and usage in this thesis[11]. Note that there is another on-going work whose material is used in Chapter 6.

Paper	**Type**	**Name of the conferences/journal**	**Used in**
[LZRG22]	Conference	International Conference on Acoustics, Speech, and Signal Processing 2022	Chapter 6
[LRG22]	Journal	SIAM Journal on Matrix Analysis and Applications	Chapters 3 to 5
[LRG23]	Preprint	None	Chapters 3 and 7

Table 1.3: List of my publications related to the thesis.

[11] Only publications related to this Ph.D. thesis are shown.

Chapter 2

From sparse deep neural networks to sparse matrix factorization

In this chapter, we introduce two problems: sparse DNN training and sparse matrix factorization. We summarize relevant works from their corresponding literatures and we establish several links between these two problems. Their strong correlation suggests that studying sparse matrix factorization can provide various interesting information about sparse DNNs. Finally, we discuss a variant of sparse matrix factorization named "Fixed support matrix factorization". This variant will be the first main target of this thesis and the results of this study will shed the light on certain questions on sparse DNNs in later chapters. The content in this chapter is taken from [Le20, LG21, LRG22, LRG23].

2.1 An introduction to neural networks

This chapter begins by recalling the classical definition of NNs and their training problems. It is very important to introduce classical NNs before presenting the concept of sparse NNs. In fact, we will see that sparse NN is a natural variant of its classical version, combined with the notion of sparsity. The introduction of NNs in this section will facilitate the lecture of our presentation of sparse NNs in the following one, especially for readers that are not familiar with classical and/or sparse NNs.

As discussed in Chapter 1, NNs are parametric models, which can be represented by a set of parameters θ. In the following, we only provide the description of the so-called Multi-Layer Perceptron (MLP) [Whi63] or feed-forward NN due to its generality. In fact, most of other architectures can be seen as variants/derivatives of MLP.

2.1.1 Definition of neural networks

The building blocks of NNs are called *layers* (sometimes, also called *fully-connected layers* if the underlying architecture is not an MLP to distinguish with other types of layers such as convolutional layers [KSH12]). Each layer is parameterized by two sets of parameters: a *weight matrix* $\mathbf{W}$ and a *bias vector* $\mathbf{b}$, altogether with a (non-parametric) activation function $\nu : \mathbb{R} \mapsto \mathbb{R}$. Just like the NNs, each layer encodes a function, which can be

computed using $\mathbf{W} \in \mathbb{R}^{m \times n}, \mathbf{b} \in \mathbb{R}^m$ and ν as:

$$f_{\text{layer}} : \mathbb{R}^n \mapsto \mathbb{R}^m : \nu(\mathbf{W}x + \mathbf{b})$$

where ν is applied to $\mathbf{W}x + \mathbf{b}$ in a coordinate-wise manner. In practice, ν is chosen to be a non-linear function for a reason that will be explained right after. Definitions of popular activation functions can be found in Appendix A. In this thesis, we put emphasis on the ReLU activation, the most used one in practice. ReLU activation is defined as:

$$\sigma : \mathbb{R} \mapsto \mathbb{R} : \sigma(x) = \max(x, 0).$$

The notation σ will be used *exclusively* for the ReLU activation.

A (deep) neural network is constructed by simply stacking (a lot of) these layers one after another. For an MLP, we denote $\mathbf{N} = (N_L, \ldots, N_0)$ its architecture, which contains the dimensions of the input layer $N_0 = d$, hidden layers $(N_{L-1}, \ldots, N_1)$ and output layer (N_L), respectively.

We represent an L-layer (or $(L-1)$-hidden layer) neural network by $\theta := \{(\mathbf{W}_i, \mathbf{b}_i)_{i=1}^L\}$ where $(\mathbf{W}_i, \mathbf{b}_i)$ are the parameters of the ith layer. With an abuse of notation, we simply called θ a NN (or a NN θ). An illustration of a NN is given by Figure 2.1. The space of parameters of an architecture θ is denoted by $\mathcal{N}_{\mathbf{N}}$, which can be mathematically formulated as:

$$\mathcal{N}_{\mathbf{N}} := \{\theta = ((\mathbf{W}_i, \mathbf{b}_i))_{i=1,\ldots,L} \mid \mathbf{W}_i \in \mathbb{R}^{N_i \times N_{i-1}}, \mathbf{b}_i \in \mathbb{R}^{N_i}, \forall i = 1, \ldots, L\}. \tag{2.1}$$

We also define $\mathcal{N}_{\mathbf{N}}^0$ a subset of $\mathcal{N}_{\mathbf{N}}$ where all the biases are set to zero, i.e.,

$$\mathcal{N}_{\mathbf{N}}^0 := \{\theta = ((\mathbf{W}_i, \mathbf{b}_i))_{i=1,\ldots,L} \mid \mathbf{W}_i \in \mathbb{R}^{N_i \times N_{i-1}}, \mathbf{b}_i = \mathbf{0}_{N_i}, \forall i = 1, \ldots, L\}. \tag{2.2}$$

It is not hard to verify that $\mathcal{N}_{\mathbf{N}}$ is isomorphic to $\mathbb{R}^K$ where $K := \sum_{i=1}^L (N_i N_{i-1} + N_i)$ is the total number of parameters of the architecture θ.

For each $\theta \in \mathcal{N}_{\mathbf{N}}$ and each sequence of activation functions $\Sigma = (\nu_L, \ldots, \nu_1)$, $\mathcal{R}_\theta^\Sigma : \mathbb{R}^{N_0} \mapsto \mathbb{R}^{N_L}$ is the function implemented by the ReLU network with parameters θ:

$$\mathcal{R}_\theta^\Sigma : x \in \mathbb{R}^{N_0} \mapsto \mathcal{R}_\theta^\Sigma(x) := \nu_L(\mathbf{W}_L \nu_{L-1}(\ldots \nu_1(\mathbf{W}_1 x + \mathbf{b}_1) \ldots + \mathbf{b}_{L-1}) + \mathbf{b}_L) \in \mathbb{R}^{N_L} \tag{2.3}$$

If we denote $f_i : \mathbb{R}^{N_{i-1}} \mapsto \mathbb{R}^{N_i} : f_i(x) = \nu_i(\mathbf{W}_i x + \mathbf{b}_i)$ the function encoded by the ith layer, then $\mathcal{R}_\theta^\Sigma$ can be seen as a composition of the function f_i, i.e.:

$$\mathcal{R}_\theta^\Sigma = f_L \circ f_{L-1} \circ \ldots \circ f_1$$

Moreover, if an activation is linear, it is not difficult to see that f_i is an affine function w.r.t. x. Since the composition of affine functions is an affine function, it is thus preferable (in practice) that $\nu_i, i = 1, \ldots, L-1$ is non-linear (excluding ν_L).

We denote by $\mathcal{R}^\Sigma$ the functional mapping from a set of parameters θ to its function, i.e., $\mathcal{R}^\Sigma(\theta) := \mathcal{R}_\theta^\Sigma$. The function space associated to an architecture $\mathbf{N}$ and a sequence of activation functions Σ is the image of $\mathcal{N}_{\mathbf{N}}$ under $\mathcal{R}^\Sigma$, which is denoted by $\mathcal{F}_{\mathbf{N}}^\Sigma$ and defined as:

$$\mathcal{F}_{\mathbf{N}}^\Sigma := \mathcal{R}^\Sigma(\mathcal{N}_{\mathbf{N}}). \tag{2.4}$$

In words, $\mathcal{F}_{\mathbf{N}}^{\Sigma}$ is the set of implementable functions by a neural network of architecture $\mathbf{N}$ and a sequence of activation functions Σ.

When $\nu_L = \mathrm{Id}$ the identity function and $\nu_1 = \ldots = \nu_{L-1} = \nu$ an activation function, we simply write $\mathcal{R}_\theta^\nu$, instead of $\mathcal{R}_\theta^\Sigma$. We adopt the same simplification for the functional mapping $\mathcal{R}^\Sigma$ and $\mathcal{F}_{\mathbf{N}}^\Sigma$, i.e., $\mathcal{R}^\nu$ and $\mathcal{F}_{\mathbf{N}}^\nu$.

When $\nu = \sigma$ the ReLU activation, we use the shorthand:

$$\begin{aligned} \mathcal{R}_\theta &:= \mathcal{R}_\theta^\sigma, \\ \mathcal{R} &:= \mathcal{R}^\sigma, \\ \mathcal{F}_{\mathbf{N}} &:= \mathcal{F}_{\mathbf{N}}^\sigma. \end{aligned} \tag{2.5}$$

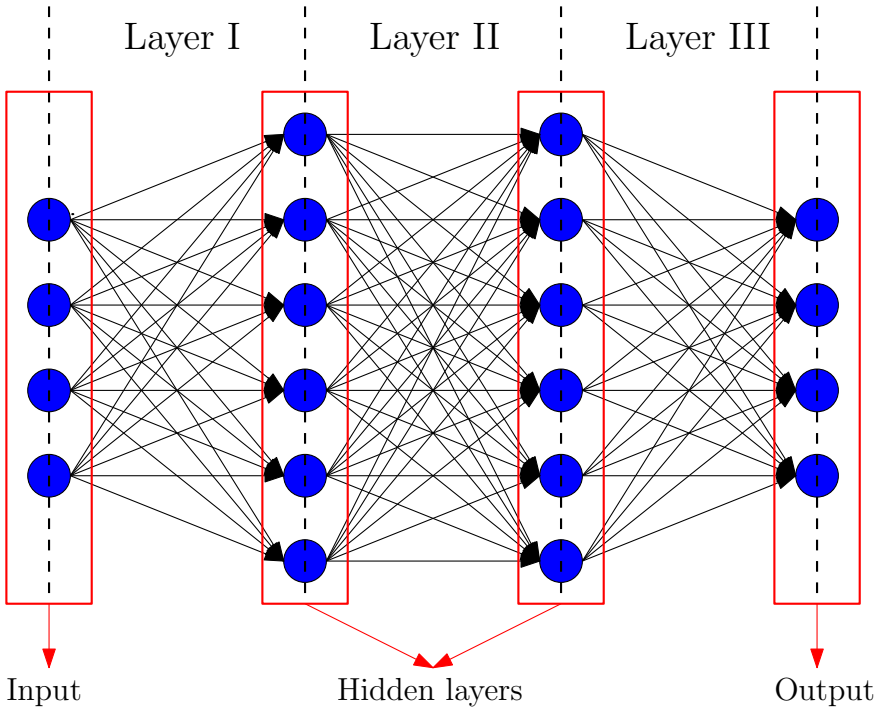


Figure 2.1: An illustration of an MLP with $L = 3$, $\mathbf{N} = (4, 6, 6, 4)$. Each arrow corresponds to a coefficient in the weight matrix $\mathbf{W}_i, i = 1, 2, 3$.

2.1.2 Training problem of neural networks

Similar to most ML models, once an architecture $\mathbf{N}$ is fixed, the training problem for a neural network requires in addition a training dataset and a loss function ℓ. As mentioned in Chapter 1, the goal of solving the training problem of neural networks is to select $\theta \in \mathcal{N}_{\mathbf{N}}$ such that for all samples[1] (x, y) in the training dataset $\mathcal{D} := \{(x_i, y_i)\}_{i=1}^n$, we have: $\mathcal{R}_\theta^\Sigma(x) = y$ in the ideal setting. Nevertheless, such θ might not exist or there might be many θ satisfying such constraints. Therefore, the approach based on optimization is

[1]Remind that x is an input and y is its associated label.

more favorable: the selection of θ is carried out by solving

$$\underset{\theta}{\text{Minimize}} \quad \mathcal{L}(\theta; \mathcal{D}) := \sum_{i=1}^{n} \ell(\mathcal{R}_\theta^\Sigma(x_i), y_i) \tag{2.6}$$

where ℓ can be thought as a distance between $\mathcal{R}_\theta^\Sigma(x)$ and y. In practice, practitioners also consider and minimize a *regularized* version of $\mathcal{L}$, which changes Equation (2.6) into:

$$\underset{\theta}{\text{Minimize}} \quad \mathcal{L}(\theta; \mathcal{D}) := \underbrace{\sum_{i=1}^{n} \ell(\mathcal{R}_\theta^\Sigma(x_i), y_i)}_{\text{data-fitting term}} + \underbrace{\lambda g(\theta)}_{\text{regularization term}} \tag{2.7}$$

where $\lambda > 0$ is a hyper-parameter. In Equation (2.7), the first term (also known as *data-fitting term*) enforces $\mathcal{R}_\theta^\Sigma(x)$ and y to be close to each other while the other (*regularization term*) originates from statistical learning and employed to prevent overfitting [HTF, Chapter 7]. Sometimes, the regularization term is reported to help further decrease the data-fitting term [KSH12, Section 5]. The hyper-parameter λ is used to control the balance between these two terms.

The choice of loss functions depends on the learning task. Among the most popular loss functions are the quadratic loss, i.e.,

$$\ell(y', y) = \|y' - y\|_2^2$$

(for regression problems) and the cross-entropy loss, i.e.,

$$\ell(y', y) = -\sum_{i=1}^{m} y'[i] \log y[i]$$

where $y[i]$ is the value of the ith coefficient of y (for classification problems). The most well-known regularizers are $\ell_2(\theta) = \|\theta\|_2^2$ (ridge regularization) or $\ell_1(\theta) = \|\theta\|_1$ (lasso regularization).

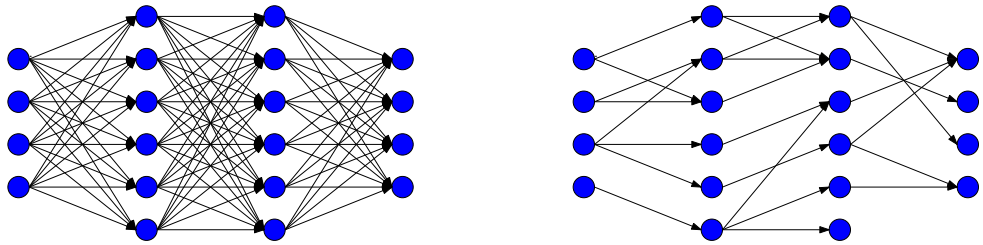
To select θ, it is thus sufficient to solve the optimization problems (2.6) and/or (2.7). However, we will not go into detail about optimization algorithms for DNNs because their presentation is out of scope of this thesis. For an excellent exhibition of these algorithms, we refer to [GBC16, Chapter 8].

2.2 An introduction to sparse neural networks

With the definition of dense NNs presented in the previous section, we proceed to the introduction of sparse NNs. We then, provide the mathematical formulation and several popular approaches to study their corresponding optimization problems, algorithmically and theoretically.

2.2.1 Sparse neural networks: definition and training problems

In a nutshell, the only difference between classical and sparse NNs is: the weight matrices $\mathbf{W}_i$ are *sparse*, i.e., they have many zero entries in the case of sparse NNs. More formally, we would like the weight matrix $\mathbf{W}_i \in \mathcal{E}_i$ where $\mathcal{E}_i$ is a set of sparse matrices. An illustration for this difference is shown in Figure 2.2.



(a) Classical (dense) neural network (b) Sparse neural network

Figure 2.2: An illustration of dense and sparse DNNs.

We make several remarks on this definition:

1. **Why don't we enforce the sparsity for the bias vectors as well?**: In fact, a large percentage of neural networks parameters comes from the weight matrices since for the ith layer, one has $N_i \times N_{i-1}$ parameters from $\mathbf{W}_i$ and *only* N_i parameters from $\mathbf{b}_i$. This observation is still valid for different types of NNs such as Convolutional Neural Networks (CNNs) [KSH12] or Attention mechanism [VSP+17]. Thus, while it is possible to impose the sparsity onto the bias vectors, it does not result into significant gain in term of model compression.

2. **Which kinds of $\mathcal{E}_i$ - the constraint set of sparse matrices - are used in practice?** There are many $\mathcal{E}_i$ used by practitioners. The set $\mathcal{E}_i$ can also be used to add expert knowledge to the models. Below is a list of sparse matrices sets that are used in practice:

   (a) **k-sparse matrices**: is defined as $\mathcal{E}_i = \mathcal{M}_k^{\text{total}} := \{\mathbf{W}_i \mid \|\mathbf{W}_i\|_0 \leq k\}$ for a constant k ($\|\cdot\|_0$ is a semi-norm, equal to the number of non-zero entries). This is by far the most used one in the literature [HPTD15, HMD16, ZG17, FC19]. Choosing $k \ll N_i \times N_{i-1}$ ensures the weight matrix $\mathbf{W}_i$ being very sparse.

   (b) **Fixed and sparse supports matrices**: The problem when we use k-sparse matrices in sparse DNNs is that optimization algorithms have to find the support (i.e., the set of matrix indices whose coefficients are non-zero) and the values of their entries simultaneously. Such algorithms usually take more iterations or extra procedures to perform well [HPTD15, ZG17, HMD16]. This observation sparks the idea of choosing a *fixed support* (also known as *sparsity pattern*) for the weight matrices. We remind readers that support of a matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$, denoted as $\text{supp}(\mathbf{W}) := \{(i,j) \mid \mathbf{W}[i,j] \neq 0\} \subseteq [\![m]\!] \times [\![n]\!]$ is a subset of indices whose coefficients are not zero[2]. Once a set of supports is chosen, training algorithms can "simply optimize" the coefficients *inside* the supports. Note"fixed support" means that the coefficients inside the "supports" are *allowed* to be non-zero. They could be zero as well. These fixed supports can be hand-crafted using expert knowledge [DCS+22b, DGE+19, LRC+21, DSG+20] or found by algorithms [LAT19, TKYG20, WZG20].

[2]Here, $\mathbf{W}[i,j]$ is the value of the matrix $\mathbf{W}$ at index (i,j) and $[\![m]\!] := \{1, \ldots, m\}$, see Notation.

Other potential candidates for $\mathcal{E}_i$ can be the set of *k-sparse per row/column matrices*, i.e., $\mathcal{M}_k^{\text{row}} = \{\mathbf{W}_i \mid \|\mathbf{W}_i[i,:]\|_0 \leq k, \forall i\}/\mathcal{M}_k^{\text{col}} = \{\mathbf{W}_i \mid \|\mathbf{W}_i[:,i]\|_0 \leq k, \forall i\}$. These sets are somewhat in-between the other two since the sparsity pattern is still learned from data (similar to k-sparse matrices) but we access to certain knowledge about them (similarly to fixed support case).

3. **Do we enforce the sparsity on all weight matrices $\mathbf{W}_i, i = 1, \ldots, L$?** Generally, we do not enforce sparsity on all the weight matrices. In practice, it is better to promote sparsity for layers which account for a large number of parameters. These layers might be a large convolutional layer or a fully-connected layer at the end of CNNs. This is the approach in many articles [DCS+22b, DGE+19, LRC+21, DSG+20]. While it is trivial, we would like to point out that if we do not want to promote sparsity for $\mathbf{W}_i$, we can simply set $\mathcal{E}_i = \mathbb{R}^{N_i \times N_{i-1}}$ (or equivalently, $(N_i N_{i-1})$-sparse matrices). Thus, our formalism remains valid.

Similarly to the case of dense DNNs, we introduce corresponding notions for sparse DNNs. We will focus on the case where $\mathcal{E}_i$ is a set of (sparse) matrices with fixed support. This choice is *atomic* in our opinion since all sets of sparse matrices such as $\mathcal{M}_k^{\text{total}}$ - the set of k-sparse matrices is a finite union of sets of sparse matrices with fixed support. Indeed, consider $\mathcal{M}_k^{\text{total}}$, we have:

$$\mathcal{M}_k^{\text{total}} = \bigcup_{I || I| \leq k} \{\mathbf{W} \mid \text{supp}(\mathbf{W}) \subseteq I\}$$

When $\mathcal{E}_i$ is a set of sparse matrices with fixed support, we simply call this type of NNs *fixed support sparse NNs*. It is worth noticing that for a fixed support sparse NNs, $\mathbf{N} = (N_L, \ldots, N_0)$ alone is not sufficient to fully characterize the architecture since it does not carry any information about the support of weight matrices $\mathbf{W}_i, i = 1, \ldots, L$. Therefore, we define $\mathbf{I} = (I_L, \ldots, I_1)$ the collection of binary masks (or support constraints) $I_i \in \{0,1\}^{N_i \times N_{i-1}}, 1 \leq i \leq L$ of weights matrices to specify the architecture, in the sense that if $I_i[\ell, k] = 0$ then $\mathbf{W}_i[\ell, k] = 0$. It is also convenient to think of I_i as the set $\{(\ell, k) \mid I_i[\ell, k] = 1\}$, which is a subset of $[\![N_i]\!] \times [\![N_{i-1}]\!]$. We will use these two interpretations (binary mask and subset) interchangeably and the meaning should be clear from the context. We will even abuse notations by denoting, e.g., $I_l \subseteq \mathbf{1}_{N_l \times N_{l-1}}$ (see the definition of $\mathbf{1}_{N_l \times N_{l-1}}$ in Notation).

Because the support constraint $I \in \{0,1\}^{m \times n}$ can be thought as a binary matrix, for any subset of the row indices $S_r \subseteq [\![m]\!]$ (resp. subset of the column indices $S_c \subseteq [\![n]\!]$), $I[S_r, :]$ (resp. $I[:, S_c]$) represents the support constraint of $I \cap S_r \times [\![n]\!]$ (resp. $I \cap [\![m]\!] \times S_c$). In addition, $I[:, i]$ (resp $I[i, :]$) denotes the support constraints on the ith row (resp. column) of I. Note that these notations do not cause any conflict with $\mathbf{W}[:, S_c]$, $\mathbf{W}[S_r, :], \mathbf{W}[:, i]$ and $\mathbf{W}[i, :]$ defined in Notation.

We denote $\mathcal{N}_{\mathbf{I}} \subseteq \mathcal{N}_{\mathbf{N}}$ the space of parameters of the sparse architecture $\mathbf{I}$, i.e.,

$$\mathcal{N}_{\mathbf{I}} := \{\theta = ((\mathbf{W}_i, \mathbf{b}_i))_{i=1,\ldots,L} : \text{supp}(\mathbf{W}_i) \subseteq I_i, \forall i = 1, \ldots, L.\}. \tag{2.8}$$

A special subset of $\mathcal{N}_{\mathbf{I}}$ is the set of network parameters with zero biases,

$$\mathcal{N}_{\mathbf{I}}^0 := \{\theta = ((\mathbf{W}_i, \mathbf{0}_{N_i}))_{i=1,\ldots,L} : \text{supp}(\mathbf{W}_i) \subseteq I_i, \forall i = 1, \ldots, L.\}. \tag{2.9}$$

The function space associated to a sparse architecture $\mathbf{I}$ and a sequence of activation functions Σ is the image of $\mathcal{N}_{\mathbf{I}}$ under $\mathcal{R}^{\nu}$.

$$\mathcal{F}_{\mathbf{I}}^{\Sigma} := \mathcal{R}^{\Sigma}(\mathcal{N}_{\mathbf{I}}). \tag{2.10}$$

When $\Sigma = (\mathrm{Id}, \nu, \dots, \nu)$ for some activation functions ν, we simple use the shorthand: $\mathcal{F}_{\mathbf{I}}^{\nu} := \mathcal{F}_{\mathbf{I}}^{\Sigma}$. When $\nu = \sigma$ the ReLU activation, we further simplify $\mathcal{F}_{\mathbf{I}}^{\nu}$ into $\mathcal{F}_{\mathbf{I}}$.

When $\nu = \mathrm{Id}$ is the identity map and all biases are zero $\mathbf{b}_i = \mathbf{0}_{N_i}$, we denote:

$$\mathcal{L}_{\mathbf{I}} := \mathcal{R}^{\mathrm{Id}}(\mathcal{N}_{\mathbf{I}}^{0}) \tag{2.11}$$

which is a subset of linear maps $\mathbb{R}^{N_0} \mapsto \mathbb{R}^{N_L}$. Since every linear map corresponds to a matrix, one can think of $\mathcal{L}_{\mathbf{I}}$ as *a subset* of $\mathbb{R}^{N_L \times N_0}$. We will use these two interpretations interchangeably.

Table 2.1 illustrates the parallel between the sets of notations that will be used for classical NNs and sparse NNs.

Meanings	**Classical NNs**	**Sparse NNs**
Architecture specification	$\mathbf{N} = (N_L, \dots, N_0)$	$\mathbf{I} = (I_L, \dots, I_1)$
Parameter space	$\mathcal{N}_{\mathbf{N}}$	$\mathcal{N}_{\mathbf{I}}$
Parameter space with zero biases	$\mathcal{N}_{\mathbf{N}}^{0}$	$\mathcal{N}_{\mathbf{I}}^{0}$
Function space ($\Sigma = (\nu_L, \dots, \nu_1)$)	$\mathcal{F}_{\mathbf{N}}^{\Sigma}$	$\mathcal{F}_{\mathbf{I}}^{\Sigma}$
Function space with ReLU activation functions (i.e. $\Sigma = (\mathrm{Id}, \sigma, \dots, \sigma)$)	$\mathcal{F}_{\mathbf{N}}$	$\mathcal{F}_{\mathbf{I}}$
Linear function space	None	$\mathcal{L}_{\mathbf{I}}$
Function mapping ($\Sigma = (\nu_L, \dots, \nu_1)$)	$\mathcal{R}^{\Sigma}$	
Function mapping with ReLU	$\mathcal{R}$	

Table 2.1: List of notations for classical and sparse NNs.

Training problem of sparse deep neural networks The training problem for sparse DNNs is nearly identical to (2.6) - the one for classical DNNs - except that we impose the constraints $\mathbf{W}_i \in \mathcal{E}_i, \forall i = 1, \dots, L$. More specifically, given a dataset $\mathcal{D} := \{(x_i, y_i)\}_{i=1}^{n}$, training a sparse DNN is bound to solve the following optimization problem:

$$\begin{aligned} \underset{\theta}{\text{Minimize}} \quad & \mathcal{L}(\theta; \mathcal{D}) := \sum_{i=1}^{n} \ell(\mathcal{R}_{\theta}^{\Sigma}(x_i), y_i) \\ \text{subject to:} \quad & \mathbf{W}_i \in \mathcal{E}_i, \forall i = 1, \dots, L \end{aligned} \tag{2.2.SNNT}$$

for some (sparse) matrices sets $\mathcal{E}_i, i = 1, \dots, L$. In fact, (2.2.SNNT) **is the optimization problem that we will study in this thesis.**

Before moving to the next section, we would like to mention that (2.2.SNNT) is not the only optimization formulation for the training of sparse DNNs. In the literature, there exists other approaches that are based on regularizations (i.e., the formulation (2.7)) such as ℓ_1 or ℓ_0 norm of the parameter θ. They will be reviewed in the next section. Nevertheless, (2.2.SNNT) is at the center of our interest since many sparse training algorithms of DNNs are based on this formulation. Many questions arise naturally from (2.2.SNNT) such as:

1. **The existence of optimal solutions**: Under which conditions (the choice of the architectures $\mathbf{N}/\mathbf{I}$, the constraint sets $\mathcal{E}_i$) does the (2.2.SNNT) always admit an optimal solution (for all dataset).

2. **Tractability of** (2.2.SNNT): Under which conditions ($\mathbf{N}/\mathbf{I}/\mathcal{E}_i$), (2.2.SNNT) can be solved efficiently (in polynomial time).

Not only are the responses for these questions theoretically interesting but they can also serve as a guideline for architectures/sparsity constraints in practice.

In the following, our review on sparse DNNs contains two components: practice and theory. We proceed with the algorithmic aspects first, and move to the theoretical side later.

2.2.2 Practical methods for sparse deep neural networks training

We classify several existing approaches into categories as follows:

Fixed support sparse models training

This approach aims to find a *fixed* binary mask/ support constraint I for every layer. Once the binary masks are identified, it is thus sufficient to optimize only the coefficients corresponding to values in the mask equal to one. This is exactly the setting of fixed support sparse NNs that is introduced in the previous section. We emphasize that this approach is *equivalent* to solve (2.2.SNNT) with $\mathcal{E}_i$ being a set of fixed support sparse matrices.

In practice, the same set of algorithms used to optimize classical DNNs [GBC16, Chapter 8] is also employed to solve the optimization problem of fixed support sparse NNs. Thus, what truly distinguishes works based on this approach is: how to choose the binary masks/ support constraints/sparsity patterns given a dense architecture $\mathbf{N}$. We classify methods of this approach into two main categories:

1. *Support constraints hand-crafted with expert knowledge*: Typical works from this category are [DGE+19, DSG+20, LRC+21, DCS+22b]. The main idea for this line of works is to replace a linear operator (the weight matrices/convolutional weights) by a product of *fixed support sparse* matrices. This concept is illustrated in Figure 2.3. Note that this replacement is equivalent to adding more layers whose weight matrices are sparse followed by an identity activation function. Thus, we actually optimize a deeper NN, but the overall number of parameters is reduced thanks to the sparsity promoted in the layers.

   A common choice of fixed supports is for instance those of the factors of well-known linear operators such as the Fourier Transform or the Hadamard Transform. More information on the choice of the support will be provided in Chapter 6, where we study the so-called *butterfly parameterization*, a unifying concept of many works [DGE+19, DSG+20, LRC+21, DCS+22b] in detail.

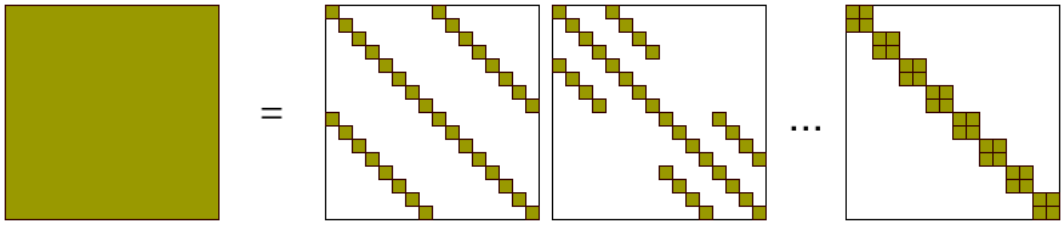


Figure 2.3: Replacing a (dense) weight matrix by the so-called fixed support sparse factors.

2. *Support constraints found by algorithms*: [LAT19, WZG20, TKYG20] are among works in this category. Differently from the previous approach, these works propose algorithms to choose a sparse subnetwork from a dense DNN architecture (i.e., only a subset of θ will be optimized while the others are set to zero). To this end, these works propose desirable properties/criteria that a subnetwork should possess/optimize to be maximally beneficial for the training/inference. Examples of such properties/criteria are gradient flow preservation [WZG20], synaptic flow preservation [TKYG20] or connection sensitivity minimization [LAT19]. Given such a criterion, a typical algorithm in this category will perform three main steps:

   (a) Given a dense architecture (such as Resnet-18,50 [HZRS16] or VGG [SZ15]), initialize the parameters of the architecture (randomly from certain distributions).

   (b) Compute a criterion score for each parameter of the dense architecture. This score can be dependent on the initial values of parameters that we just initialize in the previous step and/or the training dataset.

   (c) Create a binary mask in which parameters whose scores are higher than a certain threshold have value one. Others have their mask values set to zero. That is the binary mask of the chosen sparsity structure.

   It is worth emphasizing that this approach is somewhat similar to the *pruning and re-training* approach (which will be introduced right after) since they both define criteria to choose sparsity patterns. A fairly detailed list of these criteria can be found in the following. We emphasize that works presented in this category perform this selection of supports *once* and *before* the actual training (unlike pruning and retraining doing this selection during and after the optimization phase).

Pruning (and retraining)

A general scheme for works in this line of work is depicted in Figure 2.4. In comparison to the previous approach, pruning and retraining perform both operations simultaneously. It typically contains four to five steps:

1. *Choosing an original (dense) architecture and initialize the weights by either random values or pre-trained ones.* Among popular architectures that are chosen to be pruned are ResNet-18 [HZRS16], ResNet-50 [HZRS16], VGG [SZ15], LeNet [LBBH98], Transformers [VSP+17].

2. *Training the dense architecture*: This step is similar to the training of dense DNNs [GBC16, Chapter 8]. Training can be performed until convergence or for a fixed number of iterations/epochs.

3. *Pruning weights/neurons/filters*: We *remove/prune* weights in the weight matrices by setting them to zero. Practical implementation of pruning involves binary masks where the value zero implies that the corresponding weight is pruned. This notion is already introduced in the previous section. It is noteworthy that weights are not the only objects that can be pruned in a dense neural networks. Early works [MS89, SD88] focus on neurons pruning, i.e., by setting the values of a neuron always equal to zero. However, one can see neuron pruning as *structured* weight pruning since neuron pruning is equivalent to prune all the weights of a column/row of some weight matrices. Likewise for filter pruning (in CNN).

4. *Weight rewinding (optional)*: Some works [FC19] consider reset the remaining weights after pruning to their initial values.

5. *Retraining sparse NNs resulting from pruning*: This step is similar to fixed support sparse models training. Similarly to training, retraining can be performed either until convergence or for a fixed number of iterations. The initial parameters of this step can be either ones after pruning or ones after weight rewinding.

In fact, pruning can be seen as a way to obtain sparse support constraints for the weight matrices while the goal of retraining is to learn the coefficients inside the support constraints. Works using this approach [HPTD15, HMD16, ZG17, GEH19] typically use (2.2.SNNT) and choose $\mathcal{E}_i$ equal to the set of k-sparse matrices (where k is a hyper-parameter controlling the sparsity level of each layer/the whole NN).

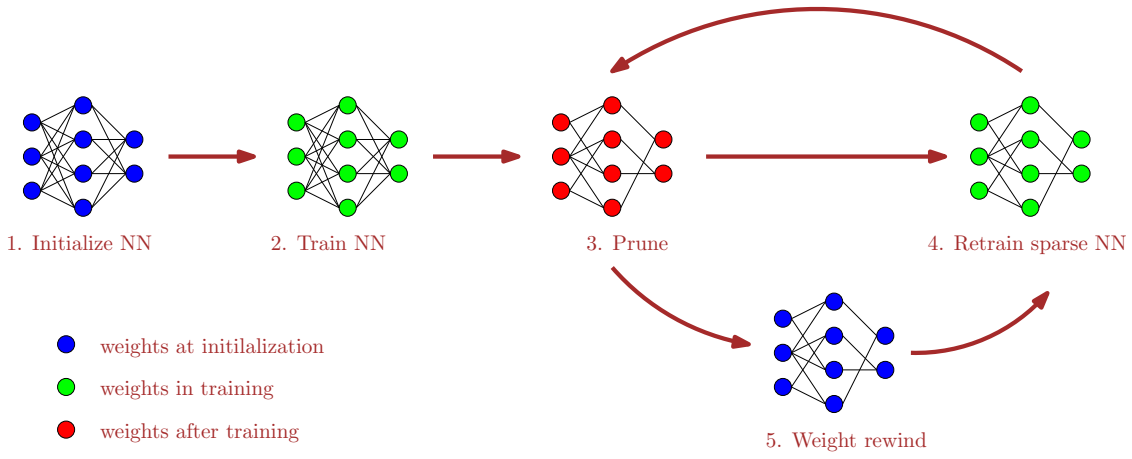


Figure 2.4: A generic framework of sparse DNNs training with pruning and retraining.

In fact, there is *almost* no conceptual difference in the steps $1-3-5$ among all works in this line of research[3]. The main difference between works *usually* lies in the pruning

[3]Here, we mean major differences. Those such as initialization types, optimization algorithms and their hyper-parameters for steps 3 and 5 are counted as conceptually minor (though they might be important factors in practice).

criterion: How important is a (group of) neuron(s)? This can be formulated as a scoring problem where (groups of) weights with lowest scores will be then pruned. Below is a list of existing approaches for such a scoring system:

1. *Magnitude*: This criterion scores weights by their absolute values. Thus, the smaller is the magnitude, the more a connection is likely to be removed (set to zero). For pruning a group of weights (e.g., prune an entire filter in convolutional neural networks or rows/columns of a weight matrix in MLP), one simply uses the sum of individual scores in the group. Works using this approach appeared in early 1990s [Hag93]. It was then revived, especially in the context of modern DNNs by a series of works [HPTD15, HMD16, ZG17, GEH19]. The so-called Lottery Ticket Hypothesis (LTH) [FC19] also used magnitude pruning.

   Magnitude pruning has an interesting connection to compressed sensing [Don06]: it is exactly equivalent to the hard thresholding operator in many algorithms of this field such as Iterative Hard Threshold [BD08], CoSAMP [NT09], Hard Thresholding Pursuit [Fou11].

2. *Input/output sensitivity*: Another idea for the importance metric is to measure how the input/output change over a given dataset. It is the origin of a plethora of measurements such as variance of neuron output (to prune neurons) [SD88] or filter output (to prune filters) [JHLL17], sensitivity matrix [EC96, ZY06, TLFF18].

3. *First order information*: Works in this category are based on the first order Taylor expansion of the loss function to justify their criteria for pruning. One idea is to score a weight w_i by the value $|\mathcal{L}(\theta) - \mathcal{L}(\theta_i)|$ where θ_i is a copy of θ with the exception that w_i is set to zero. Direct evaluation of this score for *all* weights is prohibited. Thus, [MMT+19] proposed to use the first order Taylor expansion at a point θ, which is:

   $$\mathcal{L}(\theta + \delta\theta) \approx \mathcal{L}(\theta) + \nabla_\theta \mathcal{L}^\top \delta\theta + O(\delta\theta^2)$$

   to approximate the score for each weight. By setting $\delta\theta = \theta_i - \theta = \begin{bmatrix} 0 \\ \vdots \\ w_i \\ \vdots \\ 0 \end{bmatrix}$, this score is simply approximated by $|(\nabla_{w_i} \mathcal{L}) w_i|$. This score is subsequently used in [DDZ+19a].

   Another approximation that uses the first order Taylor expansion is due to [MS89]. The authors, however, described the approach to prune neurons (equivalently, prune a whole column/row of weight matrices). Subsequent works [LAT19, XWR19] adapted this approach for (unstructured) weight pruning. In the following, we describe the idea in [LAT19] [4]. Let $M = \mathbf{1}$ (an all-one mask, see Notation) and M_i be the mask that has zero in the ith position and one elsewhere. Then it is obvious that:

   $$\mathcal{L}(\theta_i) - \mathcal{L}(\theta) = \mathcal{L}(\theta \odot M_i) - \mathcal{L}(\theta \odot M) \approx (\nabla_M \mathcal{L})^\top (M_i - M)$$

   where the second equality is obtained by using the first order Taylor expansion for M (and *not* θ). The third term is simply the ith coordinate of the gradient of $\mathcal{L}$ w.r.t. M evaluated at $M = \mathbf{1}$.

[4] We choose not to present the original idea in [MS89] since it requires substantially different notations for the explanation.

4. *Second order information*: Similar to the previous category, the main idea of this line of research is to approximate $|\mathcal{L}(\theta_i) - \mathcal{L}(\theta)|$. It is worth emphasizing that in this setting, the only requirement for θ_i is its ith coordinate value has to be zero. Other coefficients of θ_i are not bound to be equal to those of θ. The main difference is the usage of the second order approximation instead of the first one. This is motivated by the fact that once the model is trained to convergence, it is expected that $\|\nabla_\theta \mathcal{L}\|$ is small. Thus,

$$\mathcal{L}(\theta + \delta\theta) \approx \underbrace{\mathcal{L}(\theta) + \nabla_\theta \mathcal{L}^\top \delta\theta + \frac{1}{2}\delta\theta^\top \mathbf{H}\delta\theta}_{\text{second order Taylor approximation}} \approx \mathcal{L}(\theta) + \frac{1}{2}\delta\theta^\top \mathbf{H}\delta\theta$$

   where $\mathbf{H}$ is the Hessian matrix of $\mathcal{L}$. Thus, we can approximate $|\mathcal{L}(\theta) - \mathcal{L}(\theta_i)|$ by $\frac{1}{2}\delta\theta^\top \mathbf{H}\delta\theta$. Note that one can drop the absolute value symbols for $\delta\theta^\top \mathbf{H}\delta\theta$ since we assume that NNs are trained to reach a local minimum and the corresponding Hessian matrix $\mathbf{H}$ is semi-positive definite.

   Assuming that one wants to find the best w_i to prune that minimizes the difference $|\mathcal{L}(\theta) - \mathcal{L}(\theta_i)|$, the problem can be formulated as the following optimization one:

$$\min_i \left\{ \min_{\delta_i \in \mathbb{R}^K} \frac{1}{2}\delta_i^\top \mathbf{H}\delta_i \quad \text{s.t.} \quad \delta_i^\top \begin{pmatrix} 0 \\ \vdots \\ \text{i} \\ \vdots \\ 0 \end{pmatrix} = w_i \right\}$$

   where K is the total number of parameters of the given architecture. Using the Lagrange multiplier method [NW06, Chapter 12, Equation 12.17], one can show that the optimal value of the inner optimization problem for each w_i is:

$$\rho_i = \frac{w_i^2}{\mathbf{H}^{-1}[i,i]}$$

   and we will use ρ_i as the pruning criterion. It is presented in [LDS90] (under the name *Optimal Brain Damage - OBD* with an assumption that $\mathbf{H}$ is diagonal) and then [HS93] (under the name *Optimal Brain Surgeon - OBS* without assumption on $\mathbf{H}$). The sum of ρ_i in the same column/row is also used to calculate the score for neuron pruning [CSGR96] since we can interpret neuron pruning as column/row pruning.

   In practice, since computing $\mathbf{H}$ and its inversion is computationally prohibited, the Hessian is replaced by the Fischer matrix [HS93], diagonal Fischer matrix [TKTH18] or Kronecker-Factor Approximate Curvature (K-FAC) [WGFZ19, ZU19].

In summary, there is a plethora of pruning criteria, based on many different principles/approaches. It is, however, unclear which criterion is the best since their performance depends on the original dense architectures, the learning task and training data. However, as pointed out by [HABN+21, Figure 18a], it is safe to assume that magnitude is the most popular criterion, especially in the context of modern deep learning thanks to its performance and scalability [HPTD15, HMD16, ZG17].

Regularization-based training

As mentioned previously in Section 1.1.2, Formulation (2.2.SNNT) is not the only possible one to train a sparse DNN. Certain works trains sparse DNNs based on the regularized version (2.7). The choice of the regularization terms $g(\theta)$ is what distinguishes works using this approach. Since the technical contribution of this thesis has little things to do with regularization, we will not go into detail, and just recapitulate several regularizers g and their corresponding papers in Table 2.2.

Regularisation	Definition of g	Additional note	Appears in
ℓ_0	$\|\theta\|_0$	None	[CMDP19, SSB17]
ℓ_1	$\|\theta\|_1$	None	[CK14, LWF+15, CWXC20, AANR17]
Hoyer	$\|\theta\|_1^2/\|\theta\|_2^2$	None	[YWL20]
Group Lasso	$\sum_{I\in\mathcal{I}} \|\theta_I\|_2$	$\mathcal{I}$ is a partition of parameters of θ	[SCHU17, WWW+16, GEN+17]

Table 2.2: Several types of sparsity promoting regularizations and papers employing them. The list of papers is not exhaustive.

2.2.3 Theory on sparse deep neural networks

Theoretical results on sparse DNNs are not as developed as their algorithmic counterpart of which we just provided a brief review. One of the most prominent results that are related to sparse DNNs is the so-called Lottery Ticket Hypothesis (LTH) [FC19] and its follow-up works.

Lottery Ticket Hypothesis (LTH) This line of work evolves around the following hypothesis [FC19]: "*A randomly-initialized, dense neural network contains a subnetwork that is initialized such that - when trained in isolation - it can match the test accuracy of the original network after training for at most the same number of iterations.*"

In words, LTH states that for a dense architecture, one can find a sparse version that is as performant as its dense counterpart, within the same amount of training time. In the original paper [FC19], the authors study LTH *empirically* and use magnitude pruning a posteriori to identify the sparse architecture in their hypothesis on small CNNs, LeNet [LBBH98], ResNet-18 [HZRS16] and VGG-19 [SZ15] with MNIST and CIFAR10 datasets.

[RWK+19] is the first work laying the foundation for a theoretical analysis of LTH (despite the fact that their methodology is empirical!). They show that when the number of parameters of θ grows, there exists a sparse sub-network of a dense, randomly initialized NN that performs much better than random guess *without* re-training. The author proposed an algorithm to find such sub-networks and validated its effectiveness on both small (CIFAR10) and large (ImageNet) datasets with ResNet-50 and Resnet-101 [HZRS16]. The real gem in [RWK+19] is the drop of the re-training step, which remains challenging for theoretical analysis. It is thus possible to study only the pruning part, which greatly simplifies the overall analysis.

[RWK+19] is followed by a sequence of works [MYSsS20, PRN+20, OHR20], which provide qualitative results for what has been established in [RWK+19]: How large should a neural network be to ensure the existence of a performant subnetwork at random initialization (or the LTH, but we do not even need to train the parameters)? These results are crucial since they partly uncover the mystery of LTH. Nevertheless, the analysis is somewhat unsatisfying since it treats the pruning and (re)-training separately. In practice, it is reported that re-training is "a crucial part of a sparsification schedule"[HABN+21, Section 8.9].

To the best of our knowledge, theoretical works that study the dynamics of the training of pruned (and thus, sparse) DNNs remain limited. We only found one paper [EKT21] that studies jointly pruning and retraining. However, the authors of [EKT21] consider a *linear* model (i.e., no hidden-layers) to reuse tools from compressed sensing. The nature of this setting is entirely different from those of NNs.

In this thesis, we attempt to take the (re)-training step more seriously and incorporate it into the overall analysis of sparse DNNs training. However, confronting this question in the full setting of a classical DNNs (with non-linear activation function) is already challenging. In addition, another difficulty in our setting is the sparsity pattern that we need to take into account. Thus, to have a finer analysis, we introduce the problem of sparse matrix factorization, which is a simplified setting of sparse DNNs. Its formalism will be introduced in the following section.

2.3 Sparse matrix factorization and its relation to sparse deep neural networks

This section is devoted to the introduction of sparse matrix factorization and its relation to sparse DNNs. This motivates our study of sparse matrix factorization: we believe results in this simpler setting allows us to get closer to the mystery around sparse DNNs. In addition, we also argue that sparse matrix factorization is also a problem of interest on its own, with many applications in practice. We also discuss several problems related to sparse matrix factorization. Some of them are crucial for our analysis in the later technical chapters. Most notations and definitions in this section are re-used from [LRG23].

2.3.1 Problem formulation and the first relation to sparse deep neural networks

Formulation In words, sparse matrix factorization seeks sparse matrices (or factors) $\mathbf{W}_i, i = 1, \dots, L$ whose product approximates well a given (dense) matrix $\mathbf{A}$. In fact, there are many possible mathematical formulations of this problem. In this thesis, our point of view is optimization, since we want to study the optimization of sparse matrix factorization to get back to that of sparse DNNs. For such reason, we will use the formalism borrowed from [LMG16], which is:

$$\begin{aligned} &\underset{\mathbf{W}_1,\dots,\mathbf{W}_L}{\text{Minimize}} \ \|\mathbf{A} - \mathbf{W}_1 \dots \mathbf{W}_L\|_F^2 \\ &\text{subject to: } \mathbf{W}_i \in \mathcal{E}_i, \ \forall 1 \leq i \leq L. \end{aligned} \tag{2.3.SMF}$$

Relation to sparse DNNs training To justify the relation between sparse matrix factorization and sparse DNNs, let's consider the following assumption:

Assumption 2.3.1. *Assume the two following conditions hold:*

1. *The sequence of activation functions contains only the identity function:* $\Sigma = \{\mathrm{Id}, \dots, \mathrm{Id}\}$.
2. *All biases are equal to zero:* $\mathbf{b}_i = \mathbf{0}_{N_i}, \forall i = 1, \dots, L$.

We claim that under Assumption 2.3.1, (2.3.SMF) and (2.2.SNNT) are similar. Indeed, in that case, $\mathcal{R}_\theta^\Sigma$ is simplified to:

$$\begin{aligned}\mathcal{R}_\theta^\Sigma(x) &= \nu_L(\mathbf{W}_L\nu_{L-1}(\dots\nu_1(\mathbf{W}_1x + \mathbf{b}_1)\dots + \mathbf{b}_{L-1}) + \mathbf{b}_L)\\ &= \mathbf{W}_L(\dots(\mathbf{W}_1x + \mathbf{b}_1)\dots + \mathbf{b}_{L-1}) + \mathbf{b}_L\\ &= \mathbf{W}_L\mathbf{W}_{L-1}\dots\mathbf{W}_1x, \forall\theta \in \mathcal{N}_{\mathbf{N}}^0.\end{aligned}$$

where $\mathcal{N}_{\mathbf{N}}^0$ is defined in Equation (2.2). Assume that the loss function ℓ is our familiar quadratic loss: $\ell(y, y') = \|y - y'\|^2$ and the dataset $\mathcal{D}$ is represented by two matrices $\mathbf{X}$ and $\mathbf{Y}$ whose ith columns are x_i and y_i respectively, one can write the objective function of (2.2.SNNT) in a more compact form:

$$\begin{aligned}\mathcal{L}(\theta; \mathcal{D}) &= \sum_{i=1}^n \ell(\mathcal{R}_\theta^\Sigma(x_i), y_i)\\ &= \sum_{i=1}^n \|y_i - \mathbf{W}_L\mathbf{W}_{L-1}\dots\mathbf{W}_1x_i\|_2^2\\ &= \|\mathbf{Y} - \mathbf{W}_L\mathbf{W}_{L-1}\dots\mathbf{W}_1\mathbf{X}\|_F^2.\end{aligned}$$

Therefore, the following formulation is trivially equivalent to (2.2.SNNT) (under Assumption 2.3.1) and the loss function ℓ being quadratic:

$$\begin{aligned}\underset{\mathbf{W}_1,\dots,\mathbf{W}_L}{\text{Minimize}} \quad & \|\mathbf{Y} - \mathbf{W}_L\mathbf{W}_{L-1}\dots\mathbf{W}_1\mathbf{X}\|_F^2\\ \text{subject to:} \quad & \mathbf{W}_i \in \mathcal{E}_i, \forall i = 1, \dots, L.\end{aligned} \tag{2.12}$$

It is not hard to see the similarity between (2.3.SMF) and (2.12). In particular, (2.12) becomes (2.3.SMF) when $\mathbf{X} = \mathbf{I}_{N_0}$ is equal to the identity matrix and $\mathbf{Y} = \mathbf{A}$ in (2.3.SMF). This relation is not new since it has been exploited to study the optimization landscape of the training problem of classical DNNs (see, for example, [Kaw16]). Our novelty lies in the notion of sparsity: how do things change when we add sparsity constraints $\mathbf{W}_i \in \mathcal{E}_i$ to the problem? This question will be addressed throughout this thesis. The similarity between the optimization problem is, however, not the only relation between sparse matrix factorization and sparse DNNs training. In the following section, we are going to see another link between the two problems through a more algorithmic lens.

2.3.2 Algorithms for sparse matrix factorization and a relation to pruning/retraining approach in sparse DNNs training

In the following, we discuss existing methods/algorithms to solve (2.3.SMF). To the best of our knowledge, we are only aware of one relevant work [LMG16] that directly confronts with (2.3.SMF). Thus, we will devote most of this section to describe the algorithm introduced in [LMG16].

Proximal Alternating Linearized Minimization (PALM) for sparse matrix factorization The idea of [LMG16] is to apply Proximal Alternating Linearized Minimization (PALM) algorithm, which is proposed in [BST14] for (2.3.SMF). To simplify the introduction, let's only consider (2.3.SMF) with two factors (i.e., $L = 2$). PALM is initially proposed to solve the following type of problem:

$$\underset{x,y}{\text{Minimize}} \quad \Phi(x, y) := f(x) + g(y) + h(x, y) \tag{2.13}$$

where $f : \mathbb{R}^{n_1} \mapsto (-\infty, +\infty], g : \mathbb{R}^{n_2} \mapsto (-\infty, +\infty]$ (they can receive $+\infty$ value) are proper [5] and lower semi-continuous functions [6], $h : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \mapsto \mathbb{R}$ is a continuously differentiable function. PALM resembles a coordinate descent method and repeatedly performs two following steps:

$$x^{k+1} \in \text{prox}^f_{\alpha_k}\left(x^k - \frac{1}{\alpha_k}\nabla_x h(x^k, y^k)\right)$$
$$y^{k+1} \in \text{prox}^g_{\beta_k}\left(y^k - \frac{1}{\beta_k}\nabla_y h(x^{k+1}, y^k)\right)$$

where prox^f_α (reads *proximal operator* associated to a proper and semi-continuous function f) is a set-valued map and it is defined as:

$$\text{prox}^f_\alpha(x) := \arg\min_u \left\{ f(u) + \frac{\alpha}{2}\|u - x\|_2 \right\}.$$

A particular case of proximal operator is the projection operator to a non-empty closed set. Indeed, consider $\mathcal{E}$ a non-empty closed subset of $\mathbb{R}^n$ and its indicator/characteristic function:

$$\delta_\mathcal{E}(x) := \begin{cases} 0 & \text{if } x \in \mathcal{E} \\ +\infty & \text{otherwise} \end{cases}$$

then, we can verify easily that $\text{prox}^{\delta_\mathcal{E}}_\alpha(x)$ is equal to $\mathcal{P}_\mathcal{E}(x)$, the projection operator onto $\mathcal{E}$. In that case, PALM becomes a combination between coordinate descent method and projected gradient descent.

To apply PALM to (2.3.SMF), we first reformulate (2.3.SMF) (with $L = 2$) into the form of Formulation (2.13) as:

$$\underset{\mathbf{X}\in\mathbb{R}^{m\times r},\mathbf{Y}\in\mathbb{R}^{n\times r}}{\text{Minimize}} \quad \delta_{\mathcal{E}_1}(\mathbf{X}) + \delta_{\mathcal{E}_2}(\mathbf{Y}) + \|\mathbf{A} - \mathbf{X}\mathbf{Y}^\top\|_F^2. \tag{2.14}$$

It is trivial that (2.3.SMF) is equivalent to (2.14). Moreover, (2.14) has a similar form to Equation (2.13) where $f = \delta_{\mathcal{E}_1}, g = \delta_{\mathcal{E}_2}, h = \|\mathbf{A} - \mathbf{X}\mathbf{Y}^\top\|_F^2$. Thus, PALM is applicable to sparse matrix factorization. When applying to sparse matrix factorization with $L = 2$, two steps of PALM are translated as:

$$\mathbf{X}^{k+1} \in \mathcal{P}_{\mathcal{E}_1}\left(\mathbf{X}^k - \frac{1}{\alpha_k}\left(\nabla_\mathbf{X}\|\mathbf{A} - \mathbf{X}\mathbf{Y}^\top\|_F^2\right)|_{\mathbf{X}^k,\mathbf{Y}^k}\right)$$
$$\mathbf{Y}^{k+1} \in \mathcal{P}_{\mathcal{E}_2}\left(\mathbf{Y}^k - \frac{1}{\beta_k}\left(\nabla_\mathbf{Y}\|\mathbf{A} - \mathbf{X}\mathbf{Y}^\top\|_F^2\right)|_{\mathbf{X}^{k+1},\mathbf{Y}^k}\right)$$

[5] A function $f : \mathbb{R}^n \mapsto [-\infty, +\infty]$ is proper if $\{x \mid f(x) < +\infty\} \neq \emptyset$ and $\{x \mid f(x) = -\infty\} = \emptyset$

[6] A function $f : \mathbb{R}^n \mapsto [-\infty, +\infty]$ is lower semi-continuous if $\forall x_0, \liminf_{x\mapsto x_0} f(x) \geq f(x_0)$

This can be generalized for any $L > 2$. For such a generalization, we refer readers to [LMG16, Figure 4]. Under certain assumptions on f, g, h and two sequences $\{\alpha_k\}_{k\in\mathbb{N}}, \{\beta_k\}_{k\in\mathbb{N}}$ [7], any *bounded* sequence of PALM applied to (2.14) can be shown to converge to a stationary point (i.e. a point whose gradient is zero). While this convergent result is very strong, the assumption of bounded iterates does not always hold (thus, nullify the convergence result). In Chapter 3, we show an instance of (2.3.SMF) for whose iterates generated by PALM diverge due to the non-existence of an optimal solution. Our main point is to highlight theoretical and practical problems related to sparse matrix factorization, since optimal solution does not even exist in the first place. This finding also implies that the assumption on boundedness is necessary to establish convergence result.

A relation between magnitude pruning and retraining with PALM We end this discussion on PALM and its application to sparse matrix factorization by drawing some relations between PALM and the previous pruning and retraining algorithm that uses *magnitude* as criterion. As discussed previously, a popular choice of $\mathcal{E}_i$ for sparse DNN training is the set of k-sparse matrices $\mathcal{M}_k$. In fact, the projection onto the set $\mathcal{M}_k$ is equivalent to prune all coefficients except the k-largest magnitude coefficients.

Indeed, let $\mathbf{B}$ be a k-sparse matrix that is closest to $\mathbf{A}$ (in term of Euclidean distance), then it is trivial that: $\mathbf{B}[i,j] = \mathbf{A}[i,j]$ if $(i,j) \in \text{supp}(\mathbf{B})$. Thus, for a given binary mask $\mathbf{M}$ representing a matrix support of a k-sparse matrix, the closest matrix $\mathbf{B}$ of $\mathbf{A}$ such that $\text{supp}(\mathbf{B}) \subseteq \mathbf{M}$ is $\mathbf{B} = \mathbf{A} \odot \mathbf{M}$ (remind that $\odot$ is the Hadamard product between matrix). The smallest distance (given the binary mask $\mathbf{M}$) is thus: $\|\mathbf{A} \odot (\mathbf{1} - \mathbf{M})\|_F^2 = \sum_{(i,j)\notin\mathbf{M}} |A_{i,j}|^2$. Thus, the best mask is one whose one entries coincide with the top-k largest (in term of magnitude) coefficients. In words, the closest k-sparse matrix of $\mathbf{A}$ is one obtained by pruning all coefficients except top-k largest coefficients.

Therefore, the generic framework illustrated in Figure 2.4 (without weight re-initialization) really resembles PALM. However, there exist critical differences:

1. Retraining of sparse DNNs typically has multiple steps, not just one such as PALM.
2. All the layers are optimized simultaneously. It is not in the (block) coordinate-wise manner as in PALM.

Nevertheless, this relation is quite intriguing and it further motivates us to study sparse matrix factorization in detail. In our opinion, any advance in the study of sparse matrix factorization is very likely to result into interesting finding for sparse DNNs. However, it is not the only reason that motivates our study sparse matrix factorization problem. As shown in the next section, the sparse matrix factorization problem can have many applications in practice. Its impact can be much broader than just being a step to have a closer look to sparse DNNs.

2.3.3 Other applications of sparse matrix factorization

Besides the motivation of studying sparse DNNs, we further argue that sparse matrix factorization also has many applications, which are:

[7] We refer to [LMG16, Section III.B] for more detail

1. *Linear operator acceleration*: If one can approximate a dense matrix $\mathbf{A}$ by a product of sparse factors $\mathbf{W}_L \dots \mathbf{W}_1$, i.e., $\mathbf{A} \approx \prod_{i=1}^{L} \mathbf{W}_i$ then for every vectors x, we have:

$$\mathbf{A}x \approx \mathbf{W}_L(\dots(\mathbf{W}_1 x)) \tag{2.15}$$

Since the product sparse matrix - vector can be efficiently calculated, the RHS is faster to evaluate than the LHS. In fact, this is the underlying principle of many fast algorithms for linear operator evaluation.

Let's illustrate this claim by examining the Discrete Fourier Transform (DFT) and the famous Fast Fourier Transform (FFT). The *following paragraph* is partly taken from [DGE+19, Section 3]. Recall that given a vector $x \in \mathbb{R}^N$, the (linear) Fourier Transform of x is calculated as:

$$y = \mathbf{F}_N x \in \mathbb{R}^N, \qquad \mathbf{F}_N = (e^{-\frac{i2\pi}{N}kn})_{k,n}, 0 \le k, n \le N-1.$$

Due to the definition of F_N, if one assumes that N is even, then it can be proved that:

$$\mathbf{F}_N x = \begin{pmatrix} \mathbf{F}_{N/2} x_{\text{even}} + \mathbf{D}_{N/2}\mathbf{F}_{N/2} x_{\text{odd}} \\ \mathbf{F}_{N/2} x_{\text{even}} - \mathbf{D}_{N/2}\mathbf{F}_{N/2} x_{\text{odd}} \end{pmatrix}$$

where $x_{\text{even}} = \begin{bmatrix} x[0] \\ x[2] \\ \vdots \\ x[N-2] \end{bmatrix}, x_{\text{odd}} = \begin{bmatrix} x[1] \\ x[3] \\ \vdots \\ x[N-1] \end{bmatrix}$ and $\mathbf{D}_{N/2} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & e^{-2i\pi/N} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & e^{-2i\pi(N/2-1)/N} \end{bmatrix}$ is a diagonal matrix. A simple algebraic manipulation will give us the following factorization:

$$\mathbf{F}_N = \begin{pmatrix} \mathbf{I}_{N/2} & \mathbf{D}_{N/2} \\ \mathbf{I}_{N/2} & \mathbf{D}_{N/2} \end{pmatrix} \begin{pmatrix} \mathbf{F}_{N/2} & \mathbf{0} \\ \mathbf{0} & \mathbf{F}_{N/2} \end{pmatrix} \mathbf{P}_N$$

where $\mathbf{P}_N$ is a permutation matrix that permutes x to a new vector x' having the first $N/2$ entries equal to x_{even} and the remaining $N/2$ ones equal to x_{odd}. Denote the leftmost factor as $\mathbf{B}_N$, if we assume that $N = 2^p, p \in \mathbb{N}$ is a power of two, we can unroll this recursion as:

$$\begin{aligned}
\mathbf{F}_N &= \mathbf{B}_N \begin{pmatrix} \mathbf{F}_{N/2} & \mathbf{0} \\ \mathbf{0} & \mathbf{F}_{N/2} \end{pmatrix} \mathbf{P}_N \\
&= \mathbf{B}_N \begin{pmatrix} \mathbf{B}_{N/2} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_{N/2} \end{pmatrix} \begin{pmatrix} \mathbf{F}_{N/4} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{F}_{N/4} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{F}_{N/4} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{F}_{N/4} \end{pmatrix} \begin{pmatrix} \mathbf{P}_{N/2} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_{N/2} \end{pmatrix} \mathbf{P}_N \\
&= \dots \\
&= \left(\mathbf{B}_N \dots \begin{pmatrix} \mathbf{B}_2 & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{B}_2 \end{pmatrix} \right) \underbrace{\left(\begin{pmatrix} \mathbf{P}_2 & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{P}_2 \end{pmatrix} \dots \mathbf{P}_N \right)}_{\mathbf{P}}.
\end{aligned}$$

The matrices in the left bracket are called *butterfly factors/matrices*. These matrices are very structured and sparse because they have exactly two non-zero entries per

row and per column. They are the main topics of Chapter 6. Each matrix in the right bracket is a permutation matrix (i.e., a binary matrix having exactly one non-zero coefficients per row and per column). Since the set of permutation matrices is stable under product operator, the product $\mathbf{P}$ in the right bracket is also a permutation matrix itself. The matrix $\mathbf{P}$ is also known as the *bit-reversal permutation* matrix [DGE+19]. Thus, $\mathbf{F}_N = \mathbf{W}_p \dots \mathbf{W}_1 \mathbf{P}$ admits a sparse factorization into $\log N + 1$ factors whose columns and rows have at most two non-zero coefficients. In fact, the fast Fourier algorithm is equivalent to compute the RHS of Equation (2.15). One can, thus, see sparse matrix factorization as an universal approach to enable the acceleration of linear operators (if they do admit a sparse factorization). Current methods can already reverse-engineer the DFT and the Hadamard Transform for certain choices of $\mathcal{E}_i$ (in the formulation Equation (2.3.SMF)) [LG15, LZRG22, LG21].

2. *Dictionary learning*: We already introduced the main concept of dictionary learning in Chapter 1 to justify why sparse DNNs are potentially more interpretable than their dense counterparts. Here, we provide its formalism. Assume that we have a dataset $\mathcal{D} = \{y_i \in \mathbb{R}^d \mid i = 1, \dots, n\}$[8] and we represent $\mathcal{D}$ as a matrix $\mathbf{Y}$ of size $d \times n$ by stacking y_i as columns of $\mathbf{Y}$. Dictionary learning seeks for a matrix $\mathbf{D}$ and a sparse matrix $\mathbf{X}$ such that:

$$\begin{aligned} \underset{\mathbf{D},\mathbf{X}}{\text{Minimize}} \quad & \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 \\ \text{subject to:} \quad & \mathbf{X} \text{ is a sparse matrix.} \end{aligned} \tag{2.16}$$

   On the one hand, the matrix $\mathbf{D}$ acts as a *dictionary* whose columns act as its words. On the other hand, the matrix $\mathbf{X}$ can be seen as a look-up table, which contains many zero coefficients (since we wish it to be sparse). Each data point is, thus, a *sparse* linear combination of columns of $\mathbf{D}$. For more technical details as well as applications of dictionary learning, we refer to [Mai10, MBPS09, MBPS10, MBP+08].

   It is not hard to see that (2.16) is a particular case of the general (2.3.SMF) where $L = 2$ (there are two factors) and there is only sparsity constraint on the second factors.

To finish this section, we introduce several related problems of sparse matrix factorization. Certain will play an important role in the following technical chapters.

2.3.4 Related works for sparse matrix factorization

We review several classes of problems that are related to sparse matrix factorization.

Matrix decompositions in linear algebra We discuss some types of matrix decomposition which arise in linear algebra. It is worth mentioning that the list below is by no means exhaustive since we focus on decompositions that will be useful for the discussion of upcoming chapters. In particular, these problems usually seek for *not necessarily sparse*

[8] Here, a data sample is just a vector x_i and *not* a pair of vectors (x_i, y_i). The former is usually seen in the setting of *unsupervised learning* [HTF, Chapter 14] while the latter is a classical example of *supervised learning* [HTF, Chapter 2]).

factors $\mathbf{X}_i, i = 1, \ldots, L$ and the product $\mathbf{X}_1 \ldots \mathbf{X}_L$ is equal to $\mathbf{A}$ - the target matrix, up to machine precision. In contrast, (2.3.SMF) typically looks for sparse factors whose product approximates well $\mathbf{A}$.

1. *LU decomposition* [GVL96, Section 3.2]: Given a *square* matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, the LU decomposition seeks two square factors $\mathbf{L} \in \mathbb{R}^{n \times n}$ and $\mathbf{U} \in \mathbb{R}^{n \times n}$ such that $\mathbf{A} = \mathbf{LU}$ and $\mathbf{L}$ and $\mathbf{U}$ are lower triangular and upper triangular matrices respectively. Note that the existence of such $\mathbf{L}$ and $\mathbf{U}$ is not guaranteed (see [GVL96, Section 3.2.12]). However, if all principle minors of the matrix $\mathbf{A}$ are invertible, then one can recover $\mathbf{L}, \mathbf{U}$ by the Gaussian elimination process [GVL96, Theorem 3.2.1]. The complexity of calculating an LU decomposition of an $n \times n$ matrix is $\frac{2}{3}n^3$ [GVL96, Section 3.2.11].

2. *Singular value decomposition* (SVD) [GVL96, Section 2.5.3]: Given a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, the SVD aims to find three matrix $\mathbf{U} \in \mathbb{R}^{m \times m}, \mathbf{D} \in \mathbb{R}^{m \times n}, \mathbf{V} \in \mathbb{R}^{n \times n}$ such that:

   (a) The matrix $\mathbf{D} \in \mathbb{R}^{m \times n}$ is a generalized diagonal matrix, i.e., $\mathbf{D}[i, j] = 0$ if $i \neq j$. The entries in the diagonal of $\mathbf{D}$ are positive and sorted in a decreasing order. These entries are called singular values of $\mathbf{A}$.

   (b) The matrices $\mathbf{U}, \mathbf{V}$ are orthogonal matrices, i.e., $\mathbf{U}^\top \mathbf{U} = \mathbf{I}_m, \mathbf{V}^\top \mathbf{V} = \mathbf{I}_n$.

   (c) The matrix $\mathbf{A} = \mathbf{UDV}^\top$. Since $\mathbf{D}$ is a diagonal matrix, this condition can be written equivalently as:

$$\mathbf{A} = \sum_{i=1}^{\min(m,n)} \mathbf{D}[i, i]\mathbf{U}[:, i]\mathbf{V}[:, i]^\top \tag{2.17}$$

   where $\mathbf{U}[:, i], \mathbf{V}[:, i]$ are the ith column of $\mathbf{U}$ and $\mathbf{V}$ respectively. They are also known as the ith left and right eigenvectors of $\mathbf{A}$ respectively.

   This definition is also known as the complete SVD. If the matrix $\mathbf{A}$ is not full rank, i.e., $r := \text{rank}(\mathbf{A}) < \min(m, n)$, then the *compact SVD* seeks $\mathbf{U} \in \mathbb{R}^{m \times r}, \mathbf{D} \in \mathbb{R}^{r \times r}, \mathbf{V} \in \mathbb{R}^{n \times r}$ such that $\mathbf{A} = \mathbf{UDV}^\top$ where $\mathbf{D}$ is a diagonal matrix, $\mathbf{U}, \mathbf{V}$ are column orthonormal matrices (or equivalently, $\mathbf{U}^\top \mathbf{U} = \mathbf{V}^\top \mathbf{V} = \mathbf{I}_r$).

   The complexity to calculate the compact SVD is $O(mn \min(m, n))$. In reality, it is expensive to compute even the compact SVD, let alone the complete version. Usually, we are only interested in the first k (largest) eigenvalues and eigenvectors. In that case (also known as *k-truncated SVD*), the complexity can be reduced to $O(kmn)$. This speed-up is significant when $k \ll \min(m, n)$.

There are also many other types of decompositions such as Interpolative Decomposition (ID) [LWM+07], QR factorization [GVL96, Section 5.2], Schur decomposition [GVL96, Section 7.1]. The introduction of these decompositions, however, is out of the scope of this thesis.

Matrix completion In words, the problem of matrix completion is to recover a matrix $\mathbf{A}$ that is assumed to have numerically small rank and of which we known just a (small) subset of its entries. This problem is widely used in recommendation system. In the Netflix Prize competition [KBV09], it is shown to be "superior to classical nearest neighbor techniques" and allows "the incorporation of additional information such as implicit feedback, temporal effects, and confidence levels".

In the following, we present a mathematical formulation for matrix completion. Again, we will introduce an optimization formulation since it will be later used in Section 3.3. Let $\mathbf{W} \in \{0,1\}^{m\times n}$ be a binary mask that represents the set of observed entries i.e., if we observe the value of $\mathbf{A}[i,j]$, then $\mathbf{W}[i,j] = 1$. Otherwise, $\mathbf{W}[i,j] = 0$ (and in that case, we *temporarily* set $\mathbf{A}[i,j] = 0$). Then finding the rank-r matrix that matches the observed entries of $\mathbf{A}$ can be done by solving the following problem.

$$\underset{\mathbf{B}\in\mathbb{R}^{m\times n},\mathrm{rank}(\mathbf{B})\le r}{\text{Minimize}} \quad \|(\mathbf{A}-\mathbf{B})\odot\mathbf{W}\|_F^2 = \sum_{(i,j)|\mathbf{W}[i,j]=1} (\mathbf{A}-\mathbf{B})[i,j]^2.$$

Since all rank r matrix admit a parameterization: $\mathbf{B} = \mathbf{X}\mathbf{Y}^\top$ where $\mathbf{X}\in\mathbb{R}^{m\times r}, \mathbf{Y}\in\mathbb{R}^{n\times r}$ [BM03], the above formulation can be written equivalently as:

$$\underset{\mathbf{X}\in\mathbb{R}^{m\times r},\mathbf{Y}\in\mathbb{R}^{n\times r}}{\text{Minimize}} \quad \|(\mathbf{A}-\mathbf{X}\mathbf{Y}^\top)\odot\mathbf{W}\|_F^2 = \sum_{(i,j)|\mathbf{W}[i,j]=1} (\mathbf{A}-\mathbf{X}\mathbf{Y}^\top)[i,j]^2. \tag{2.18}$$

This formulation is sometimes also called *low rank matrix completion with noise*. It is shown in [GG10] that (2.18) is NP-hard, even for $r = 1$. This fact will be used in Section 3.3.

Low rank matrix approximation and principal component analysis (PCA) Low rank matrix approximation is, in fact, equivalent to matrix completion when $\mathbf{W} = \mathbf{1}_{m\times n}$, i.e., all coefficients of $\mathbf{A}$ are observed. In that case, the formulation (2.18) becomes:

$$\underset{\mathbf{X}\in\mathbb{R}^{m\times r},\mathbf{Y}\in\mathbb{R}^{n\times r}}{\text{Minimize}} \quad \|\mathbf{A}-\mathbf{X}\mathbf{Y}^\top\|_F^2. \tag{2.19}$$

Different from the general matrix completion problem, low rank matrix approximation is polynomially tractable. It is shown in [EY36] that the best rank-r matrix $\mathbf{B}$ can be found by computing the r-truncated SVD of $\mathbf{A}$. This fact will be elaborated and generalized in Chapter 5 where we discuss the tractability of sparse matrix factorization.

An application of low rank matrix approximation (and SVD) is principle component analysis (PCA) [JC16]. Given a dataset $\mathcal{D}$ represented by a matrix $\mathbf{A}\in\mathbb{R}^{m\times n}$ (whose columns are data points), the goal of PCA is to find a linear subspace $\mathcal{U}$ and approximate each data point by its orthogonal projection to $\mathcal{U}$. This can be seen as a data dimension reduction technique because if the dimension of $\mathcal{U}$ (denoted by k) is much smaller than m (i.e., $k \ll m$), then we reduce the dimension of data from m to k. It can be shown in [JC16] that if one chooses $\mathcal{U}$ to minimize the total distance between datapoints and their projections to $\mathcal{U}$, i.e.,

$$\underset{\mathcal{U}}{\text{Minimize}} \quad \sum_{i=1}^{n} \|\mathbf{A}[:,i] - \mathcal{P}_{\mathcal{U}}(\mathbf{A}[:,i])\|_2^2$$

where $\mathcal{P}_{\mathcal{U}}$ is the orthogonal projection onto the linear subspace $\mathcal{U}$, then $\mathcal{U}$ is exactly spanned by the first k left-eigenvectors of $\mathbf{A}$. Equivalently, one can also say that PCA seeks for a rank-k matrix $\mathbf{B}$ (thus, all columns of $\mathbf{B}$ live in a k-dimensional linear subspace) that is closest to $\mathbf{A}$. These views explain why PCA is a direct application of low rank matrix approximation and SVD.

Robust principal component analysis (RPCA) Robust principle component analysis (RPCA) is a variant of PCA, proposed in [CLMW11]. This approach is introduced to address the following problem: If a few entries of $\mathbf{A} \in \mathbb{R}^{m\times n}$ (the data representative matrix) are grossly corrupted (or modified by a huge value), then the solution obtained by SVD will change greatly. To fix this problem, authors in [CLMW11] model the matrix $\mathbf{A}$ as the sum of a low rank matrix and a sparse matrix, i.e.

$$\mathbf{A} = \mathbf{L} + \mathbf{S}$$

where $\mathbf{L}$ is a low rank matrix and $\mathbf{S}$ is a sparse one. To recover $\mathbf{L}$ and $\mathbf{S}$, it is sufficient to solve the following optimization problem:

$$\begin{aligned} \underset{L,S\in\mathbb{R}^{m\times n}}{\text{Minimize}} \quad & \|\mathbf{A} - \mathbf{L} - \mathbf{S}\|^2 \\ \text{Subject to:} \quad & \operatorname{rank}(\mathbf{L}) \leq r, \|\mathbf{S}\|_0 \leq s \end{aligned} \tag{2.3.RPCA}$$

where r and s are the desired rank and sparsity of the matrices $\mathbf{L}$ and $\mathbf{S}$ respectively. Our discussion of (2.3.RPCA) (notably about the existence of its optimal solution) can be found in Chapter 3.

Tensor decomposition problem Similar to the case of matrices, tensors (of higher order) [9] arise naturally in many situations. For the sake of simplification, our presentation will be restricted order three tensors, i.e., $\mathbf{A} \in \mathbb{R}^{m\times n\times p}$. All argument can be naturally generalized to higher order tensors.

In practice, finding a compact representation and efficient manipulation for tensors is an important problem since storing a tensor alone is already expensive $O(mnp)$. One of the most used representations is the *canonical polyadic decomposition* (CPD), which seeks to write a tensor $\mathbf{A}$ as:

$$\mathbf{A} = \sum_{i=1}^{r} x_i \otimes y_i \otimes z_i \tag{2.20}$$

where $\otimes$ is the Kronecker product and $x_i \in \mathbb{R}^m, y_i \in \mathbb{R}^n, z_i \in \mathbb{R}^p$. The smallest natural number r such that there exists $(x_i, y_i, z_i), i = 1, \ldots, r$ to make Equation (2.20) hold is called the rank of $\mathbf{A}$. Equation (2.20) can be seen as a generalization of Equation (2.17) of SVD (but use the Kronecker product of three vectors instead of two).

It is thus desirable to assume that $\mathbf{A}$ has low rank and to recover (x_i, y_i, z_i). This will allow the storing cost to decrease to $O((m + n + p)r)$ instead of a full storage $O(mnp)$. Nevertheless, finding the rank of a given tensor is NP-hard [Has90]. Thus, in practice, given a tensor $\mathbf{A}$, one fixes a target rank r and solve the following optimization problem:

$$\underset{(x_i,y_i,z_i)\in\mathbb{R}^m\times\mathbb{R}^n\times\mathbb{R}^p}{\text{Minimize}} \quad \|\mathbf{A} - \sum_{i=1}^{r} x_i \otimes y_i \otimes z_i\|^2. \tag{2.3.TD}$$

[9] In fact, matrix is a tensor of order two

We will discuss about (2.3.TD) in Chapter 3 where we talk about the existence of optimal solution.

Due to the relation with sparse DNNs and its wide coverage of many problems of theoretical and practical interests, we hope that readers find it convincing to study sparse matrix factorization problem. However, directly studying sparse matrix factorization using the generic (2.3.SMF) seems too *vague* since many properties of (2.3.SMF) (such as loss function landscape, complexity) are dependent on the choice of the set of sparse matrices $\mathcal{E}_i$. It is, thus, desirable to identify a particular family of $\mathcal{E}_i$ that is *universal* in the sense that an analysis of its corresponding instances of (2.3.SMF) can somehow shed light on the study of other choices of $\mathcal{E}_i$. Such a choice is decided and justified in the following.

2.4 Fixed support matrix factorization

As stated earlier, in this section, we propose a family of $\mathcal{E}_i$ of (2.3.SMF) that allows us to study in detail the problem of sparse matrix factorization. We name the problem corresponding to this family *fixed support matrix factorization* (FSMF). In fact, one of the most notable difficulties of (2.3.SMF) is to deal with the exponential number of admissible supports for the factors. As suggested by its name, FSMF makes the assumption that the support of the factors are known in advance (or equivalently, they are *fixed*). This section introduces this new problem. First we show how many classical matrix decomposition/factorization problems introduced in Section 2.3.4 find themselves in the framework of FSMF. Therefore, we revisit them from the point of view of FSMF. Then, we demonstrate the importance and necessity for its study, notably in the context of (2.3.SMF) and (2.2.SNNT). Most of the contents of this section is re-used from [LRG22, Section 1].

2.4.1 Problem formulation

Given a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, fixed support matrix factorization problem (FSMF) seeks two sparse factors $(\mathbf{X}, \mathbf{Y})$ that solve the following problem:

$$\begin{aligned} \underset{\mathbf{X}\in\mathbb{R}^{m\times r},\mathbf{Y}\in\mathbb{R}^{n\times r}}{\text{Minimize}} \quad & L(\mathbf{X},\mathbf{Y}) = \|\mathbf{A}-\mathbf{X}\mathbf{Y}^\top\|_F^2 \\ \text{Subject to:} \quad & \mathtt{supp}(\mathbf{X}) \subseteq I \text{ and } \mathtt{supp}(\mathbf{Y}) \subseteq J \end{aligned} \tag{2.4.FSMF}$$

where we remind readers that $\|\cdot\|_F$ is the Frobenius norm, $I \subseteq [\![m]\!]\times[\![r]\!]$, $J \subseteq [\![n]\!]\times[\![r]\!]$ are given support constraints, i.e., $\mathtt{supp}(\mathbf{X}) \subseteq I$ implies that $\forall (i,j) \notin I, \mathbf{X}[i,j] = 0$. Figure 2.5 illustrates an instance of (2.4.FSMF). It is, in fact, a restricted class of instances of Problem (2.3.SMF), in which just two factors are considered ($L = 2$) and with *prescribed supports*.

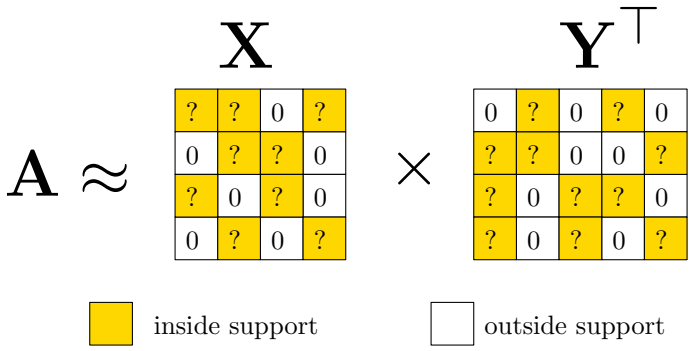


Figure 2.5: An illustration of Problem (2.4.FSMF). Yellow and white squares indicate the indices inside and outside the support constraints (I, J) respectively.

Readers might notice that (2.4.FSMF) uses $\mathbf{Y}^\top$ instead of $\mathbf{Y}$ as in the formulation (2.3.SMF). In fact, this choice of formulation (with or without the transpose) does not change the nature of the problem. Nevertheless, it will greatly simplify our presentation of many results in the upcoming chapters.

Thanks to our access to (I, J), which are essentially the sets of variables of $(\mathbf{X}, \mathbf{Y})$ that *are allowed to* from zero, it is sufficient to only minimize the function $L(\mathbf{X}, \mathbf{Y})$ w.r.t these variables. Therefore, one can also view (2.4.FSMF) as an unconstrained optimization problem.

In the next section, we will argue why studying (2.4.FSMF) might be useful for a study of (2.3.SMF). After that, we will provide as an example several existing problems that can be viewed as or reduced to (2.4.FSMF). These examples demonstrate that while being a class of Sparse matrix factorization, (2.4.FSMF) has a wide coverage of interesting problems and it merits a study of its own.

2.4.2 Motivations for the study of fixed support matrix factorization

In this section, we explain the impetus behind the study of (2.4.FSMF). We believe that it is interesting for at least three following reasons:

(2.4.FSMF) **is a natural subproblem of** (2.3.SMF) : In fact, any heuristic algorithm for the solution of (2.3.SMF) will eventually need to deal with a subproblem of the form (2.4.FSMF), one way or another. Indeed, matrix factorization with sparsity constraints somehow generalizes the linear inverse problem, in which we want to recover a sparse vector $x \in \mathbb{R}^n$ from the knowledge of its measurement vector (possibly corrupted by noise) $y = \mathbf{A}x \in \mathbb{R}^m$ with known measurement matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$. Mimicking the decomposition of the classical linear inverse problem into a *support recovery step* and a *coefficient recovery step* (which are employed in many algorithms such as Hard Thresholding Pursuit (Section 2.3), CoSAMP [NT09] or Subspace Pursuit [DM09]), Problem (2.3.SMF) can also be split into two subproblems:

1) Determine the supports of $\mathbf{X}$ and $\mathbf{Y}$, i.e. the set of indices $\texttt{supp}(\mathbf{X})$, $\texttt{supp}(\mathbf{Y})$ whose coefficients are different from zero. For instance, if $\mathcal{E}_\mathbf{X} = \mathcal{E}_\mathbf{Y} = \mathcal{M}_k^{\text{total}} = \{\mathbf{Z} \mid$

$\|\mathbf{Z}\|_0 := |\text{supp}(\mathbf{Z})| \leq k\}$ are the set of matrices with at most k non-zero entries, we need to identify the position of (at most) k non-zero coefficients of $\mathbf{X}$ and $\mathbf{Y}$.

2) Determine the value of the coefficients in the supports of $\mathbf{X}$ and $\mathbf{Y}$.

The solution of a problem in the form of (2.4.FSMF) will be needed both for *one-step* algorithms that jointly estimate the supports and coefficients, and for the *two-step* algorithms that solve the two problems successively. Also, as it happens in sparse linear regresssion, many common post-processing methods consist in "debiasing" the solution obtained by a two-step approach. Thus, we always have to face (2.4.FSMF) when solving (2.3.SMF), regardless the choice of $\mathcal{E}_i, i = 1, \dots L$. The study of Equation (2.4.FSMF), in our opinion, is *universal*.

Characterization of the asymptotic behavior of heuristics such as PALM [BST14, LMG16] : When applied to (2.3.SMF), the asymptotic behavior of PALM can be understood by studying an instance of (2.4.FSMF). Indeed, PALM updates the factors alternatively by a projected gradient step onto the set of the constraints. It is experimentally observed that for many instances of the problem, the support becomes constant after a certain number of iterations.

Let us clarify this claim with an example. We consider an instance of Problem (2.3.SMF) with $N = 2, \mathbf{X}^i \in \mathbb{R}^{100\times100}, i = 1, 2$ and the constraints $|\text{supp}(\mathbf{X}^i)| \leq 1000, i = 1, 2$. In this setting, running PALM is equivalent to an iterative method in which we consecutively perform one step of gradient descent for *each* factor, while keeping the other fixed, and project that factor onto $\{\mathbf{X} \mid \mathbf{X} \in \mathbb{R}^{100\times100}, |\text{supp}(\mathbf{X})| \leq 1000\}$ by simple hard-thresholding[10]. Figure 2.6 illustrates the evolution of the difference between the support of each factor before and after each iteration of PALM through 5000 iterations (the difference between two sets $\mathbf{B}_1$ and $\mathbf{B}_2$ is measured by $|(\mathbf{B}_1 \setminus \mathbf{B}_2) \cup (\mathbf{B}_2 \setminus \mathbf{B}_1)|$, i.e., the number of indices which belongs to exactly one set $B_i, i = 1, 2$). We observe that when the iteration counter is large enough, the factor supports do not change (or equivalently they become *fixed*): further iterations of the algorithm simply optimize an instance of (2.4.FSMF).

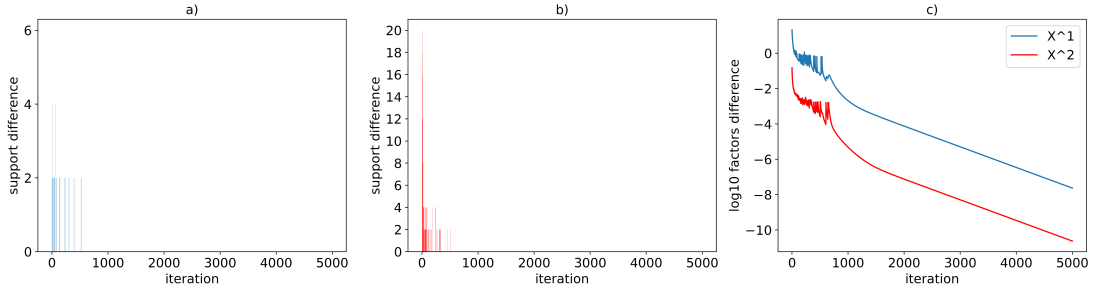


Figure 2.6: Support change for the first (a) and the second (b) factor in PALM. The norm of the difference between two consecutive factors updates is depicted in (c) (logarithmic scale).

[10] Code for this experiment can be found in [LGR22]

Behavior of pruning and retraining based on magnitude As shown in Section 2.3.2, there is a connection between PALM applied to (2.3.SMF) and the pruning and retraining approach based on magnitude applied to (2.2.SNNT). It is natural to expect an analogous phenomenon in the case of Sparse Deep Neural Networks. Indeed, it is reported that sparse training generally happens in two phases where the former discovers important connections and the latter simply fine-tunes this pattern [ARS18]. This observation were echoed in other researches, which emphasizes that sparsity pattern is formed in the early phase of the training and rarely changed after that [MLN19, DDZ+19b]. In the context of Sparse Deep Neural Networks, this phenomenon is called *early structure adaptation* [HABN+21, Section 2.4.2]. In addition, optimizing the coefficients inside a fixed support is exactly the retraining phase, which is widely used [ZG17, HPTD15, FC19] and shown to be crucial for the success of the pruning approach [HABN+21, Chapter 2.4.6].

Therefore, to develop a more precise understanding of the possible convergence of PALM in (2.4.FSMF) and pruning/training approach in (2.2.SNNT), it is important to understand properties of (2.4.FSMF). For instance, in the example illustrated in Figure 2.6, once the supports stop to change, the factors $(\mathbf{X}_n^1, \mathbf{X}_n^2)$ converge inside this fixed support (Figure 2.6c). However, there are cases in which PALM generates iterates $(\mathbf{X}_n^1, \mathbf{X}_n^2)$ diverging to infinity. Such an example will be presented in Chapter 4 (where we study the landscape of optimization function). This is not in conflict with the convergence results for PALM in this context [BST14, LMG16] since these are established *under the assumption of bounded iterates* [LMG16, Section III.B].

2.4.3 Well known instances of fixed support matrix factorization

We present in the following several well-known problems, that are either direct instances or closely related to (2.4.FSMF).

Low rank matrix approximation (LRMA) [EY36] (or cf. Section 2.3.4): By taking $I = [\![m]\!] \times [\![r]\!]$, $J = [\![n]\!] \times [\![r]\!]$ (i.e., no support constraints on $(\mathbf{X}, \mathbf{Y})$), addressing (2.4.FSMF) is equivalent to looking for the best rank r matrix approximating $\mathbf{A}$, cf. Figure 2.7(a). Indeed, a matrix $\mathbf{B}$ of rank at most r can be parameterized as $\mathbf{X}\mathbf{Y}^\top$ where $\mathbf{X} \in \mathbb{R}^{m\times r}, \mathbf{Y} \in \mathbb{R}^{n\times r}$. This is also known as the *Burer Monteiro* factorization [BM03] in the literature. In the rest of this thesis, we will also refer to this instance as the *full support case* or *full support matrix factorization* [11]. More interestingly, this problem is known to be polynomially tractable, cf. Section 2.3.4. This is not the only instance to have this property, in Section 5.2, we enlarges the family of supports for which (2.4.FSMF) remains tractable.

LU decomposition [GVL96, Chapter 3.2] (or cf. Section 2.3.4): Considering $m = n = r$ and $I = J = \{(i, j) \mid 1 \leq j \leq i \leq n\}$, it is easy to check that (2.4.FSMF) is equivalent to factorizing $\mathbf{A}$ into a lower and an upper triangular matrix ($\mathbf{X}$ and $\mathbf{Y}^\top$ respectively, cf. Figure 2.7(b)).In this case, the *infimum* of (2.4.FSMF) is always zero (cf. Chapter 3). It is worth noticing that there exists a non-empty set of matrices for

[11] Since previous works also considered the case $r \geq m, n$, low rank approximation might be misleading sometimes. That is why we occasionally use the name full support matrix factorization to emphasize this fact., where no support constraints are imposed.

which this infimum is not attained (or equivalently matrices which do not admit the **LU** decomposition [GVL96]). This behaviour will be further discussed in Chapter 3 and Chapter 4. More importantly, our analysis of (2.4.FSMF) will also cover the non-zero infimum case as well.

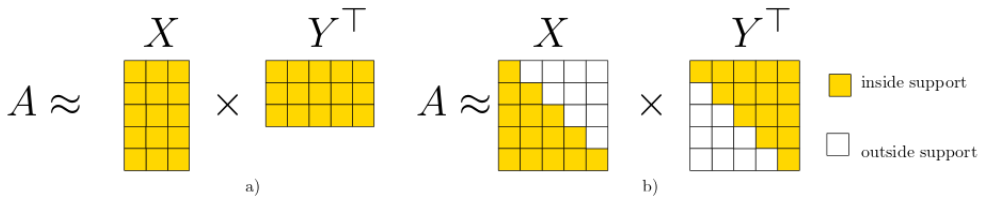


Figure 2.7: Illustrations for(a) LRMA and (b) **LU** decomposition as instances of (2.4.FSMF).

Butterfly structure and fast transforms [DGE$^+$19, CDL$^+$22, DSG$^+$20, LG15, CDY09]: Many linear operators admit fast algorithms since their associated matrices can be written as a product of sparse factors whose supports are known to possess the *butterfly structure* (and they are *known* in advance). This is the case for instance of the Discrete Fourier Transform (DFT) or the Hadamard transform (HT) that we already introduced in Section 2.3.4. Figure 2.8 illustrates such a factorization of the Hadamard Transform of size $2^L \times 2^L, L = 3$. Moreover, this structure can also be found in variety of domains/problems such as fast integral [CDY09], non-uniform Fourier transform [PST01], matrix factorization [LG15] and deep learning [DGE$^+$19, CDL$^+$22, DSG$^+$20].

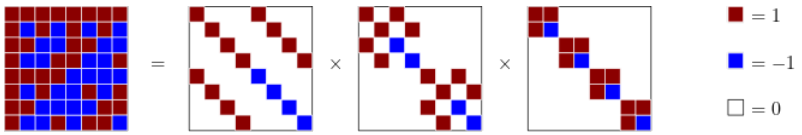


Figure 2.8: The factorization of the Hadamard transform of size 8×8 ($N = 3$).

Although our analysis of (2.4.FSMF) only deals with $L = 2$, the butterfly structure allows one to reduce to this case in a recursive manner [LZRG22, ZGR21]. We will discuss about the butterfly structure and its generalization in detail in Chapter 6.

Hierarchical matrices [Hac99, HK00]: Hierarchical matrices are a set of *data-sparse* matrices, i.e., matrices are not sparse themselves, but th can be described by much fewer variables than the total number of their coefficients. Thus, their storage, multiplication and even inversion operator can be performed efficiently [Hac99, BK16, HK00]. In practice, these matrices arise naturally from integral operators or the inversions of the matrices of boundary value problems [Hac99, HK00]. For demonstration purposes, we only report the definition of the class of hierarchically off-diagonal low-rank (HODLR) matrices (defined in [BK16, Section 3.1], [Hac99, Section 2.3]), a class of hierarchical matrices [12]. For convenience, we report the definition only for a *square* matrix whose size is a power of two, i.e $n = 2^N, N \in \mathbb{N}$.

[12] There are many other classes of hierarchical matrices such as $\mathcal{H}$, HSS or $\mathcal{H}^2$ matrices [BK16]

Definition 2.4.1 (HODLR matrices)**.** A matrix $A \in \mathbb{R}^{2^N \times 2^N}$ is called an HODLR matrix if either of the following two holds:

- $N = 0$, i.e., $A \in \mathbb{R}^{1\times 1}$.
- A has the form $A = \left[\begin{smallmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{smallmatrix}\right]$ for $A_{i,j} \in \mathbb{R}^{2^{N-1}\times 2^{N-1}}, 1 \leq i, j \leq 2$ such that A_{21}, A_{12} (so-called off-diagonal block matrices) are of rank at most one and $A_{11}, A_{22} \in \mathbb{R}$ are HODLR matrices.

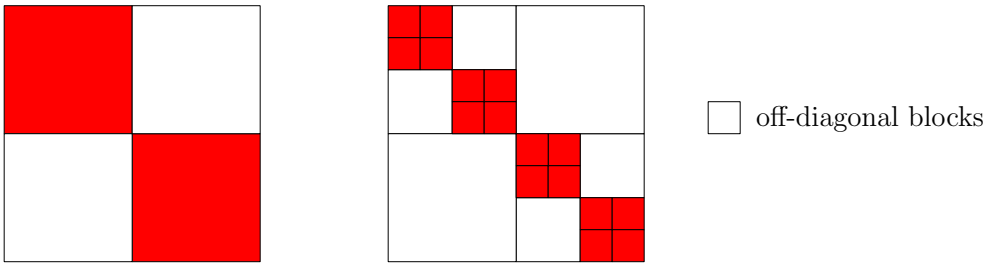


Figure 2.9: Illustration of two levels of recursion of HOLDR Definition 2.4.1.

Using Definition 2.4.1, we prove in Lemma 5.2.4 that the class of HODLR matrices can be expressed as *the product of two factors with fixed supports*, that are illustrated on Figure 2.10. Therefore, the task of finding the closest $\mathcal{H}$-matrix from this class to approximate a given matrix is reduced to (2.4.FSMF).

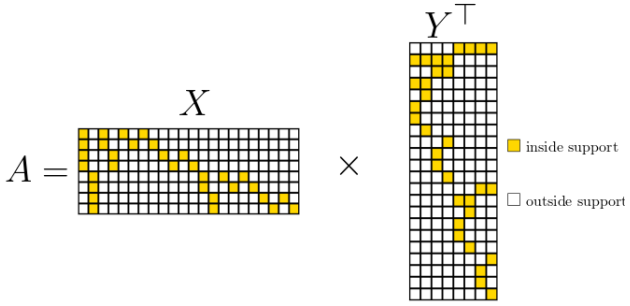


Figure 2.10: Two fixed supports for factors of a HODLR matrix of size 8×8 illustration based on analysis of Lemma 5.2.4

Matrix completion (cf. Section 2.3.4): Readers which are not familiar with the matrix completion problem can find more information about it in Section 2.3.4. We show that matrix completion can be reduced to (2.4.FSMF), which is the main result of Section 3.3.

2.5 An outlook of the thesis

Throughout this thesis, our main objective is to address some fundamental questions about sparse matrix factorization (2.3.SMF) and sparse deep neural network (2.2.SNNT):

1. **Existence**: Given sparsity constraints, does the loss functions of (2.3.SMF) and (2.2.SNNT) admit a (global) minimizer?

2. **Tractability**: If they admit global minimizers, are there efficient (polynomial) algorithms capable of finding one of these minimizers?

3. **Tractability by cases**: If the answers for these previous questions are negative, are there sub-cases where the answers are positive?

4. **Landscape**: What do the landscapes of the loss functions of (2.3.SMF) and (2.2.SNNT) look like?

Our previous discussions imply that the key to unlock the answers to these questions might be the study of (2.4.FSMF). Therefore, the rest of this thesis is devoted to seek for the answers to these questions in the setting of (2.4.FSMF). More specifically, in the next three chapters, we will address the questions of **Existence** and **Tractability** (Chapter 3), **Tractability by cases** (Chapter 5) and **Landscape** (Chapter 4 - Chapter 5) for (2.4.FSMF). These results allow us to return to answer same questions in the settings of (2.3.SMF) (Chapter 4) and (2.2.SNNT) (Chapter 7).

Part I

Fixed support matrix factorization: the challenges

Chapter 3

Ill-posedness and NP-hardness

In this chapter, we first consider the question of existence of optimal solutions of the instances of (2.4.FSMF). In general, the answer depends on the instance input: the target matrix $\mathbf{A}$ and the support constraints (I, J). The opening of this chapter presents two examples of (2.4.FSMF), one in which optimal solutions exist and another in which they do not. These two examples are famous instances of (2.4.FSMF) introduced in Section 2.4: low rank matrix approximation and **LU** factorization. Thus, we investigate the following question: given a pair of constraints (I, J), does the corresponding (2.4.FSMF) problem always admit a global minimizer regardless of the matrix $\mathbf{A}$? In particular, we show that this problem is decidable by using the *quantifier elimination algorithm*, a celebrated result in Real Algebraic Geometry [BPR06]. Secondly, we prove that (2.4.FSMF) is NP-hard to solve: given an arbitrary (I, J), finding the infimum (2.4.FSMF) up to some additive error ϵ is *NP-hard*. This implies that even if the supports are known, finding the best factorization remains challenging. Finally, we discuss several open questions and relate them to the existing literature. The materials in this chapter are developed from [LRG22, Remark A.1, Section 2] and [LRG23].

3.1 Preliminary: well-posedness or ill-posedness?

Existence of optimal solutions is an important element when it comes to design and analyze algorithms for optimization problems. In particular, consider (2.4.FSMF):

1. **Practical viewpoint**: If an instance of (2.4.FSMF) does not admit at least one global minimizer, then the iterates (satisfying the support constraints) obtained by any iterative method that seeks to optimally approximate the matrix $\mathbf{A}$ will eventually diverge. This is formulated and proved in the following:

   Claim 3.1.1. *If an instance of* (2.4.FSMF) *corresponding to the input* $(\mathbf{A}, I, J)$ *does not admit any optimal solution, then any sequence* $(\mathbf{X}_n, \mathbf{Y}_n)_{n\in\mathbb{N}}$ *such that* $L(\mathbf{X}_n, \mathbf{Y}_n)$ *converges to the infimum of* (2.4.FSMF) *is necessarily unbounded.*

   Proof. The idea of this proof can be found in many sources. For example, in [SL06], author uses the same idea to show the inherent numerical instability for the tensor decomposition problem. In [PRV21], this idea is applied in the context of neural

network training. We adapt it correspondingly to (2.4.FSMF) for convenience. Note that the argument works for any continuous function (not just $L(\mathbf{X}, \mathbf{Y})$). The proof is by contradiction.

Let $l \geq 0$ be the infimum of (2.4.FSMF), for the sake of contradiction, we suppose that there exists a bounded sequence $(\mathbf{X}_n, \mathbf{Y}_n)_{n\in\mathbb{N}}$ such that:

$$\lim_{n\to\infty} L(\mathbf{X}_n, \mathbf{Y}_n) = l.$$

Since the sequence is bounded, by compactness, there exists a convergent subsequence $(\mathbf{X}_{\varphi_n}, \mathbf{Y}_{\varphi_n})_{n\in\mathbb{N}}$, whose limit points are $(\mathbf{X}^\star, \mathbf{Y}^\star)$. By continuity of L, we have $L(\mathbf{X}^\star, \mathbf{Y}^\star) = l$. This implies that the infimum l is attainable, a contradiction. □

Thus, absence of an optimum might cause numerical instability for an optimization algorithm (since the variables might diverge).

2. **Theoretical viewpoint**: the existence of optimal solutions is crucial for the analysis of algorithms and their properties (for example, the properties of convergence, or the characterization of the properties of the optimum).

Unfortunately, such existence of an optimum is *not* guaranteed for (2.4.FSMF). Before going further, we propose and justify some terminology that will be used throughout this chapter.

Definition 3.1.2 (Well-posedness of an instance of (2.4.FSMF))**.** An instance of (2.4.FSMF) with an input $(\mathbf{A}, I, J)$ is *well-posed* if it admits a global optimizer. Otherwise, we say that it is *ill-posed*.

In other words, we will use "well-posed" or "well-posedness" to indicate the situation where optimal solutions exist. Otherwise, we use "ill-posed" or "ill-posedness". It might be strange for certain readers since these notions *literally* are not related to the absence/existence of optimum. In fact, our choice is motivated by two reasons:

1. The origin of these terms is due to J. Hadamard in [Had02] on mathematical modelling of physical phenomena. According to [Had02], a mathematical model is well-posed if it satisfies:

   (a) **Existence**: A solution exists.

   (b) **Uniqueness**: The solution is unique.

   (c) **Stability**: The solution's behavior changes continuously with the initial conditions.

   If we *translate* these conditions into the context of (2.4.FSMF), one can realize that the condition on uniqueness is never satisfied. In fact, if a pair of factors $(\mathbf{X}, \mathbf{Y})$ yields the optimal solution, so does any pair $(\mathbf{XD}, \mathbf{YD}^{-1})$ for any $\mathbf{D}$ invertible diagonal matrix. Since uniqueness does not hold, neither does stability. Thus, there is only one condition of well-posedness that (2.4.FSMF) can have: existence of an optimal solution. Thus, there is a logic behind this terminology when it come to (2.4.FSMF).

2. We choose the terminology "well-posedness" and "ill-posedness" to conform with the literature. These words have been adopted to indicate the non-existence of optimal solutions for the tensor decomposition problem [SL06], robust principal component analysis/matrix completion [JAS19] and neural network training [LMQ21]. Our choice is in line with the literature and prevents possible divergence in terms of terminology.

Remark 3.1.3. In the following, we shorten: "an instance of (2.4.FSMF) with input $(\mathbf{A}, I, J)$ is well-posed (resp. ill-posed)" as in Definition 3.1.2 to the simple "$(\mathbf{A}, I, J)$ is well-posed (resp. ill-posed)".

In the following, we give examples for two scenarios: $(\mathbf{A}, I, J)$ is well-posed and ill-posed.

Well-posedness of low rank matrix approximation: As shown in Section 2.4.1, given a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, finding its best rank-$r$ approximation is equivalent to solving (2.4.FSMF) with $I = [\![m]\!] \times [\![r]\!]$ and $J = [\![r]\!] \times [\![n]\!]$. This famous problem can be solved optimally by using Singular Value Decomposition ([EY36] or cf. Section 2.3.4 for more detail). This proves that an optimal solution always exists and the input $(\mathbf{A}, I, J)$ is well-posed for *all* matrices $\mathbf{A} \in \mathbb{R}^{m\times n}$.

Ill-posedness of LU factorization : In Section 2.4.1, we have shown that (2.4.FSMF) with $m = n = r$ and $I_n = J_n = \{(i, j) \mid 1 \leq j \leq i \leq n\}$ is equivalent to the LU decomposition problem (cf. Section 2.3.4). We show that there exists a matrix $\mathbf{A}$ such that $(\mathbf{A}, I_2, J_2)$ is ill-posed. Consider the following example:

Example 3.1.4. Consider the following matrix $\mathbf{A}$:

$$\mathbf{A} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

The infimum of (2.4.FSMF) with $(\mathbf{A}, I_2, J_2)$ is zero, which can be shown by considering the following sequence:

$$\mathbf{X}_k = \begin{pmatrix} -k & k \\ 0 & \frac{1}{k} \end{pmatrix}, \mathbf{Y}_k = \begin{pmatrix} k & k \\ 0 & \frac{1}{k} \end{pmatrix}$$

In the limit, when k goes to infinity, we have:

$$\lim_{k\to\infty} \|\mathbf{A} - \mathbf{X}_k \mathbf{Y}_k^\top\|_F^2 = \lim_{k\to\infty} \frac{1}{k^4} = 0.$$

Yet, there does not exist any pair $(\mathbf{X}, \mathbf{Y})$ such that $\|\mathbf{A} - \mathbf{X}\mathbf{Y}^\top\|^2 = 0$. Indeed, any such pair would need to satisfy:

$$\begin{aligned} \mathbf{X}[1, 2]\mathbf{Y}[2, 2] &= \mathbf{A}[1, 2] = 1 \\ \mathbf{X}[2, 2]\mathbf{Y}[1, 2] &= \mathbf{A}[2, 1] = 1 \\ \mathbf{X}[2, 2]\mathbf{Y}[2, 2] &= \mathbf{A}[2, 2] = 0 \end{aligned}$$

However, the third equation implies that either $\mathbf{X}[2, 2] = 0$ or $\mathbf{Y}[2, 2] = 0$, which makes either $\mathbf{X}[2, 2]\mathbf{Y}[1, 2] = 0$ or $\mathbf{X}[1, 2]\mathbf{Y}[2, 2] = 0$. This leads to a contradiction.

These two scenarios illustrates two categories for a pair of support constraints (I, J), which are defined as follows:

Definition 3.1.5 (Well-posed and ill-posed support constraints)**.** A pair of support constraints (I, J) is well-posed if $(\mathbf{A}, I, J)$ is well-posed for *all* $\mathbf{A}$. Otherwise, (I, J) is ill-posed.

In the following, we provide a necessary and sufficient condition to determine the well-posedness (or equivalently, ill-posedness) of a support pair (I, J).

Proposition 3.1.6 (Conditions for support constraints)**.** *A pair of support constraints (I, J) is well-posed if and only if the set:*

$$\mathcal{E}_{I,J} := \{\mathbf{X}\mathbf{Y}^\top \mid \mathrm{supp}(\mathbf{X}) \subseteq I, \mathrm{supp}(\mathbf{Y}) \subseteq J\} \subseteq \mathbb{R}^{m\times n} \tag{3.1}$$

is closed (in the usual topology of $\mathbb{R}^{m\times n}$).

A formal proof for Proposition 3.1.6 can be found in Appendix B.1. From high level, it is based on the fact that (2.4.FSMF) is equivalent to the problem of projection onto $\mathcal{E}_{I,J}$. It is well-known that to make the projection operator to a set $\mathcal{E}$ well-defined, it is necessary and sufficient that the set $\mathcal{E}$ is closed and non-empty. It is clear that $\mathcal{E}_{I,J}$ is non-empty for all (I, J) (since $\mathbf{0} \in \mathcal{E}_{I,J}$, regardless of (I, J)). Therefore, there is only the closedness of $\mathcal{E}_{I,J}$ to be checked.

Remark 3.1.7. Another way to show that (I, J) of LRMA is well-posed is to use a well-known characterization of the set $\mathcal{M}_r$ of matrices whose rank is at most r. In fact, a matrix $\mathbf{M} \in \mathcal{M}_r$ if and only if all of its square sub-matrices of size $(r + 1) \times (r + 1)$ have zero determinant. Because the set of non-invertible (zero determinant) matrices is closed, $\mathcal{M}_r$ is equal to a finite (the number of sub-matrices is finite) intersection of closed sets, thus, closed as well. Using Proposition 3.1.6, we can also prove the well-posedness of (I, J).

In fact, Example 3.1.4 is taken from [LRG22, Remark A.1]. While we were working on this paper, we were unaware that this matrix $\mathbf{A}$ was already mentioned in [GVL96, Chapter 3.2], as a typical example where $\mathbf{LU}$ decomposition is not possible. Using existing results on the $\mathbf{LU}$ decomposition, one can prove that (I_n, J_n) (remind that $I_n = J_n = \{(i, j) \mid 1 \leq j \leq i \leq n\}$) is ill-posed with many different methods (for example, constructing a *pathological* matrix $\mathbf{A}$ as in Example 3.1.4 for arbitrary n). However, to highlight the importance of Proposition 3.1.6, we will prove this result by a topological argument, i.e., by proving $\mathcal{E}_{I_n,J_n}$ is not closed. To this end, we need the following result, which provides a sufficient and necessary condition for the existence of the $\mathbf{LU}$ decomposition:

Theorem 3.1.8 ([OJ05] Necessary and sufficient condition for the existence of the $\mathbf{LU}$ decomposition)**.** *Consider $\mathbf{A} \in \mathbb{R}^{n\times n}$, the matrix $\mathbf{A}$ admits an $\boldsymbol{LU}$ decomposition if and only if:*

$$\mathrm{rank}(\mathbf{A}[[\![k]\!], [\![k]\!]]) + k \geq \mathrm{rank}(\mathbf{A}[[\![n]\!], [\![k]\!]]) + \mathrm{rank}(\mathbf{A}[[\![k]\!], [\![n]\!]]), \forall k \in [\![n]\!] \tag{3.2}$$

Theorem 3.1.8 results into the following classical one:

Theorem 3.1.9 ([GVL96] A sufficient condition for the existence of LU decomposition)**.** *A matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ admits an LU decomposition if $det(\mathbf{A}[[\![k]\!], [\![k]\!]]) \neq 0, \forall 1 \leq k \leq n - 1$.*

Proof. Here is a simple proof using Theorem 3.1.8. If $\det(\mathbf{A}[[\![k]\!],[\![k]\!]]) \neq 0$, then $\operatorname{rank}(\mathbf{A}[[\![k]\!],[\![k]\!]]) = k$. Since $\operatorname{rank}(\mathbf{A}[[\![n]\!],[\![k]\!]), \operatorname{rank}(\mathbf{A}[[\![k]\!],[\![n]\!]]) \leq k$, the assumption of Theorem 3.1.8 holds for all $k \leq n$. This proves the result. □

Combining Theorem 3.1.9 and Theorem 3.1.8, we have:

Lemma 3.1.10. *For $n \geq 2, n \in \mathbb{N}$, $\mathcal{E}_{I_n,J_n}$ is not closed.*

Proof. It is sufficient to prove that $\mathcal{E}_{I_n,J_n}$ is dense in $\mathbb{R}^{n\times n}$ (i.e., $\overline{\mathcal{E}_{I_n,J_n}} = \mathbb{R}^{n\times n}$) but $\mathcal{E}_{I_n,J_n} \neq \mathbb{R}^{n\times n}$.

1. Proof of $\overline{\mathcal{E}_{I_n,J_n}} = \mathbb{R}^{n\times n}$: for any matrix $\mathbf{A}$, consider $\mathbf{A}_\epsilon = \mathbf{A} + \epsilon\mathbf{I}_n$. Since one can choose ϵ arbitrarily small such that $\mathbf{A}_\epsilon$ satisfies the condition of Theorem 3.1.9, we conclude that $\mathbf{A}_\epsilon \in \mathcal{E}_{I_n,J_n}$ for arbitrary small ϵ. Hence, $\overline{\mathcal{E}_{I_n,J_n}} = \mathbb{R}^{n\times n}$.

2. Proof of $\mathcal{E}_{I_n,J_n} \neq \mathbb{R}^{n\times n}$: Consider matrix $\mathbf{A}$:

$$\mathbf{A} = \begin{pmatrix} 0 & \mathbf{1}_{n-1}^\top \\ \mathbf{1}_{n-1} & \mathbf{A}' \end{pmatrix}$$

   where $\mathbf{A}' \in \mathbb{R}^{(n-1)\times(n-1)}$. It is easy to see that $\operatorname{rank}(\mathbf{A}[[\![1]\!],[\![1]\!]]) = 0, \operatorname{rank}(\mathbf{A}[[\![1]\!],[\![n]\!]]) = \operatorname{rank}(\mathbf{A}[[\![n]\!],[\![1]\!]]) = 1$. Therefore, the assumption of Theorem 3.1.8 does not hold for $k = 1$ and thus, $\mathbf{A} \notin \mathcal{E}_{I_n,J_n}$. □

By invoking Proposition 3.1.6, we can conclude that (I_n, J_n) is ill-posed by using Lemma 3.1.10. Proposition 3.1.6 and its proof show the origin of the ill-posedness: the non-closedness of $\mathcal{E}_{I,J}$. Therefore, to decide whether (I, J) is well-posed, it is sufficient to verify the closedness of $\mathcal{E}_{I,J}$. The central question that will be addressed in this chapter is, thus, the following:

Question 3.1.11. Given (I, J), is $\mathcal{E}_{I,J}$ closed or not?

In the next section, we will give an algorithmic answer for Question 3.1.11. Our approach is based on tools taken from Real Algebraic Geometry.

3.2 An algorithm for deciding whether a pair of support constraints is ill-posed or not

In this section, we introduce an algorithm to decide whether a pair of support constraints of (2.4.FSMF) is ill-posed or not. Our main tools come from real algebraic geometry, a branch of mathematics that deals with *polynomials* of real numbers [1]. We start by introducing notions and results of real algebraic geometry that are relevant for our problem. After that, we apply these results to answer Question 3.1.11.

[1] In fact, this field considers *real closed fields*, which are the generalization of $\mathbb{R}$. However, since giving such an introduction of real algebraic geometry to its full scale is out of scope of this thesis, we restrain ourself to the minimal setting, which is simply $\mathbb{R}$.

3.2.1 Real Algebraic Geometry

This section is organized and presented as in the textbook [BPR06, Chapter 2] (with slight modifications to better suit our needs).

Definition 3.2.1 (Basic semi-algebraic set)**.** A subset of $\mathbb{R}^n$ is a *basic* semi-algebraic set if it has the following form:

$$\{x \in \mathbb{R}^n \mid P(x) = 0 \text{ and } Q_i(x) > 0, \forall i = 1, \ldots, \ell\}$$

where $P, Q_i : \mathbb{R}^n \mapsto \mathbb{R}, i = 1, \ldots, \ell, \ell \in \mathbb{N}$ are polynomials.

In the previous definition, it is possible to have $\ell = 0$ and a basic semi-algebraic set is the zeros of a polynomial in that case. In addition, if one chooses $P(x) = 0, \forall x$ the zero polynomial, then a basic semi-algebraic set is an intersection of sets of the form: $\{x \in \mathbb{R}^n \mid Q(x) > 0\}$ (with Q polynomial).

Example 3.2.2. Consider the two following sets:

$$A = \{x \in \mathbb{R}^n \mid \sum_{i=1}^{n} x_i^2 = 1 \text{ and } x_i^2 > 1/2\}$$
$$B = \{x \in \mathbb{R}^n \mid \sum_{i=1}^{n} \exp(x_i) \leq 1\}$$

The set A is a basic semi-algebraic set while B might not be one (because $\sum_{i=1}^{n} \exp(x_i)$ is not a polynomial).

Definition 3.2.3 (Semi-algebraic sets)**.** A subset of $\mathbb{R}^n$ is a semi-algebraic set if it is equal to finite union of basic semi-algebraic sets.

Example 3.2.4. Most of the familiar subsets of $\mathbb{R}^n$ are semi-algebraic sets. The following list presents some well-known semi-algebraic sets:

1. Unit ball: $\mathcal{B}(\mathbf{0}, 1) := \{x \in \mathbb{R}^n \mid \sum_{i=1}^{n} x_i^2 \leq 1\}$ is a basic semi-algebraic set, thus, a semi-algebraic set.

2. Polytope (defined as finite intersection of N half-spaces: $\{x \mid a_i^\top x + b_i \leq 0\}$ for some $a_i \in \mathbb{R}^n, b_i \in \mathbb{R}, i = 1, \ldots, N$) is generally not a basic but a semi-algebraic set since it can be written as:

$$\emptyset \subseteq \bigcup_{I \subseteq [\![N]\!]} \{x \mid \sum_{i \in I} (a_i^\top x + b_i)^2 = 0 \wedge \bigwedge_{i \notin I} a_i^\top x + b_i < 0\}$$

   where both $\sum_{i \in I} (a_i^\top x + b_i)^2$ and $a_i^\top x + b_i$ are polynomial functions.

3. The set of matrices of size $m \times n$ of rank at most r (denoted as $\mathcal{M}_{m,n}^r$) is also a basic semi-algebraic since it can be written as:

$$\mathcal{M}_{m,n}^r := \{\mathbf{X} \in \mathbb{R}^{m \times n} \mid \bigwedge_{R,C} \det(\mathbf{X}[R, C]) = 0, \forall R \subseteq [\![m]\!], C \subseteq [\![n]\!], |R| = |C| = r + 1\}$$

   and the determinant of a matrix is a polynomial w.r.t its coefficients.

4. The set of k-sparse matrices of size $m \times n$ is generally not a basic semi-algebraic set but a semi-algebraic set since it can be written as:

$$\bigcup_{I \subseteq [\![m]\!] \times [\![n]\!], |I|=k} \{\mathbf{X} \mid \sum_{(i,j) \notin I} \mathbf{X}[i,j]^2 = 0\}$$

As one can see, semi-algebraic sets are omnipresent. However, to identify them, one needs to find suitable equalities (and inequalities) of polynomials to describe them. In many cases, such equalities/inequalities are non-trivial to find. The following theorem, also known as the projection theorem of semi-algebraic sets, provides an easier way to recognize (or construct) them. In words, the theorem states that: The projection of a semi-algebraic set to a lower dimension is still a semi-algebraic set (of lower dimension).

Theorem 3.2.5 (Projection theorem of semi-algebraic sets [BPR06, Theorem 2.92])**.** *Let A be a semi-algebraic set of $\mathbb{R}^n$ and define:*

$$B = \{(x_1, \ldots, x_{n-1}) \mid \exists x_n, (x_1, \ldots, x_{n-1}, x_n) \in A\}$$

then B is a semi-algebraic set of $\mathbb{R}^{n-1}$.

Remark 3.2.6. Theorem 3.2.5 is a powerful result. Its proof [BPR06, Section 2.4] (which is constructive) shows a way to express B (in Theorem 3.2.5) by using only the first $n-1$ variables $(x_1, \ldots, x_{n-1})$.

A proof of Theorem 3.2.5 is out of scope of this thesis. Nevertheless, we can illustrate its constructive approach in a minimalist example, as follows:

Example 3.2.7. Consider the set:

$$A := \{(x_2, x_3) \mid \exists x_1, x_1^2 x_2^2 + x_3 = 1\}$$

In fact, A is the projection of the set $B = \{(x_1, x_2, x_3) \mid x_1^2 x_2^2 + x_3 = 1\}$ onto the second and the third coordinates. Thus, by Theorem 3.2.5, it is a semi-algebraic set.

Another way (which is constructive) to prove this is *to cut* A into two sets by considering two cases:

1. If $x_2 = 0$, then $x_3 = 1$.
2. If $x_2 \neq 0$, then to have $x_1^2 x_2^2 + x_3 = 1$ for some x_1, it is necessary and sufficient that $x_3 \leq 1$.

Combining these two cases, we can write A as:

$$A = \{(x_2, x_3) \mid x_3 = 1\} \cup \{(x_2, x_3) \mid x_2^2 > 0 \wedge x_3 < 1\}$$

Thus, we show that A is a semi-algebraic set by definition.

A direct corollary of Theorem 3.2.5 is the following:

Corollary 3.2.8. *Let A be a semi-algebraic set of $\mathbb{R}^{n+m}$ and define:*

$$B = \{x \mid \exists y \in \mathbb{R}^m, (x, y) \in A\} \subseteq \mathbb{R}^n$$

then B is a semi-algebraic set of $\mathbb{R}^n$.

Corollary 3.2.8 is a powerful tool to recognize semi-algebraic sets. We first demonstrate its usefulness by proving that $\mathcal{M}_{m,n}^r$ is a semi-algebraic set. In fact, we can use the Burer Monteiro factorization [BM03] to write

$$\mathcal{M}_{m,n}^r = \{\mathbf{A} \mid \exists \mathbf{X} \in \mathbb{R}^{m\times r}, \mathbf{Y} \in \mathbb{R}^{n\times r}, \|\mathbf{A} - \mathbf{X}\mathbf{Y}^\top\|_F^2 = 0\}.$$

Since $\|\mathbf{A}-\mathbf{X}\mathbf{Y}^\top\|_F^2$ is a polynomial w.r.t. the coefficients of $\mathbf{A}, \mathbf{X}, \mathbf{Y}$, using Corollary 3.2.8 is sufficient to conclude. In the next section, we show another application of Corollary 3.2.8: $\mathcal{E}_{I,J}$ is also a semi-algebraic set for any pair of support constraints (I, J). This result forms the first intuition that the closedness of $\mathcal{E}_{I,J}$ can be resolved by tools of real algebraic geometry.

Next, we introduce the concept of the *language* of an ordered field. From the following paragraph until Definition 3.2.9, we reuse the material of [BPR06, Chapter 2] due to its excellent presentation. However, for simplicity, we will adapt these materials correspondingly to only treat the language of $\mathbb{R}$. When we refer to a subring[2] of $\mathbb{R}$, it is convenient to think of it (and by abuse of notation, we also denote it) as $\mathbb{Z}$ (the set of integers). Note that the introduced result does not apply to fields that are not ordered such as $\mathbb{C}$ (since we cannot compare two arbitrary complex numbers). The notion of semi-algebraic set in Definition 3.2.3 (which involves the inequality $Q(x) > 0$) does not make sense when the underlying field is not ordered.

The central definition of the language of $\mathbb{R}$ is that of a *formula*, an abstraction of semi-algebraic sets. In particular, the definition of formula is recursive: a formula is built from atoms - equalities and inequalities of polynomials whose coefficients are in a subring $\mathbb{Z}$ of $\mathbb{R}$. It can be also formed by combining with logical connectives "and", "or", and "negation" $(\wedge, \vee, \neg)$ and existential/universal quantifiers $(\exists, \forall)$. A formula has variables, which are those of atoms in the formula itself. *Free variables* of a formula are those which are not preceded by a quantifier $(\exists, \forall)$. The definitions of a formula and its free variables are given recursively as follows:

Definition 3.2.9 (Formulas and their free variables)**.** Consider $\mathbb{R}$ and $\mathbb{Z} \subseteq \mathbb{R}$ a subring, a formula Φ and its set of variables $\texttt{Var}(X)$ are defined recursively as:

1. An atom: If $P \in \mathbb{Z}[X]$ (where $\mathbb{Z}[X]$ is the set of (multivariate) polynomials with coefficients in $\mathbb{Z}$) then $\Phi := (P = 0)$ (resp. $\Phi := (P > 0)$) is a formula and its set of free variables is $\texttt{Free}(\Phi) := \{X_1, \ldots, X_n\}$ where n is the number of variables.

2. If Φ_1 and Φ_2 are formulas, then so are $\Phi_1 \vee \Phi_2, \Phi_1 \wedge \Phi_2$ and $\neg\Phi_1$. The set of free variables are defined as:

   (a) $\texttt{Free}(\Phi_1 \vee \Phi_2) := \texttt{Free}(\Phi_1) \cup \texttt{Free}(\Phi_2)$.

   (b) $\texttt{Free}(\Phi_1 \wedge \Phi_2) := \texttt{Free}(\Phi_1) \cup \texttt{Free}(\Phi_2)$.

   (c) $\texttt{Free}(\neg\Phi_1) = \texttt{Free}(\Phi_1)$.

3. If Φ is a formula and $X \in \texttt{Free}(\Phi)$, then $\Phi' = (\exists X)\Phi$ and $\Phi'' = (\forall X)\Phi$ are also formulas and $\texttt{Free}(\Phi') := \texttt{Free}(\Phi) \setminus \{X\}$, and $\texttt{Free}(\Phi'') := \texttt{Free}(\Phi) \setminus \{X\}$.

[2]Formally, a subring $\mathbb{Z}$ of a field $(\mathbb{R}, +, \times, 0, 1)$ is a subset of $\mathbb{R}$ such that $(\mathbb{Z}, +, 0)$ is a group and $(\mathbb{Z}, \times, 1)$ is a monoid. Different from group, a monoid does not require an inverse for each element.

Example 3.2.10. Consider the following formula:

$$\exists x_1, x_1^2 x_2^2 + x_3 = 1$$

This formula has three variables: (x_1, x_2, x_3) and two free variables (x_2, x_3). In fact, this formula is the abstraction of the semi-algebraic set in Example 3.2.7. In fact, by fixing (x_2, x_3), say $(x_2, x_3) = (1, 1)$, the formula becomes $\exists x_1, x_1^2 + 1 = 1$, which is equivalent to a boolean value: either true or false. Thus, the semi-algebraic set in Example 3.2.7 can be viewed as a set of values of free variables of a formula in which the formula is true.

Definition 3.2.11 (Sentence)**.** A sentence is a formula of an ordered field with no free variable.

Example 3.2.12. Consider two formulas:

$$\Phi_1 = \{\exists X_1, X_1^2 + X_2^2 = 0\}$$
$$\Phi_2 = \{\exists X_1, \exists X_2, X_1^2 + X_2^2 = 0\}$$

While both are formulas, Φ_1 is not a sentence and Φ_2 is a sentence. Thus, Φ_2 is either true or false. Here, Φ_2 is true (since $X_1^2 + X_2^2 = 0$ has a root $(0, 0)$). Nevertheless, if one consider $\Phi_2' = \{\exists X_1, \exists X_2, X_1^2 + X_2^2 = -1\}$, then Φ_2' is false.

An algorithm deciding whether a sentence is true or not is very tempting since sentences can be used to express many theorems in the language of $\mathbb{R}$. The proof or disproof will be then given by an algorithm. Such an algorithm does exist. This algorithm is the main tool for our algorithmic answer to Question 3.1.11.

Theorem 3.2.13 (Decision problem [BPR06, Algorithm 11.36])**.** *There exists an algorithm to decide whether a given sentence is correct or not with complexity* $O(sd)^{O(1)^{n-1}}$ *where* s *is a bound on the number of polynomials in* Φ*,* d *is a bound on the degrees of the polynomials in* Φ *and* n *is the number of variables.*

A full description of [BPR06, Algorithm 11.36] (quantifier elimination algorithm) is totally out of this thesis's scope. Nevertheless, we will try to explain it in a concise way. The key observation is Theorem 3.2.5, the central result of real algebraic geometry. As discussed in Remark 3.2.6, its proof implies that one can replace a sentence by another whose number of quantifiers is reduced by one such that both sentences agree (both are true or false). Applying this procedure iteratively will lead to a sentence without any quantifier (and the remaining ones are only coefficients in the subring). We check the correctness of this final sentence by trivially verifying all the equalities/inequalities and obtain the answer for the original one.

3.2.2 A doubly exponential algorithm to detect the ill-posedness of Fixed Support Matrix Factorization

This section is devoted to provide an algorithm to decide on the ill-posedness of (2.4.FSMF). To show how powerful Theorem 3.2.13 is, we will consider a generalized version of Question 3.1.11, which is for multiple factors:

Question 3.2.14. Given $\mathbf{I} = (I_L, \dots, I_1)$, decide whether the set $\mathcal{L}_{\mathbf{I}} := \{\mathbf{X}_L \dots \mathbf{X}_1 \mid \text{supp}(\mathbf{X}_i) \subseteq I_i \subseteq [\![N_i]\!] \times [\![N_{i-1}]\!], \forall i = L, \dots, 1\}$ is closed or not.

Note that $\mathcal{L}_{\mathbf{I}}$ is defined identically to Equation (2.11). In fact, it will be equal to $\mathcal{E}_{I,J}$ if $L = 2, I = I_2, J = I_1$. Therefore, Question 3.2.14 is a generalized version of Question 3.1.11. We consider this generalized version since it will be re-used in Chapter 7. We proceed to the first result, which states that $\mathcal{L}_{\mathbf{I}}$ is a semi-algebraic set.

Lemma 3.2.15. *Given any constraint set $\mathbf{I} = (I_L, I_{L-1}, \dots, I_1)$, the set $\mathcal{L}_{\mathbf{I}}$ is semi-algebraic.*

Proof. Given $\mathbf{I} = (I_L, \dots, I_1)$ and indices (i, j) we define the polynomial:

$$P^{\mathbf{I}}_{i,j}(\mathbf{X}_L, \dots, \mathbf{X}_1) = \sum_{(\ell_{L-1}, \dots, \ell_1) \in E^{\mathbf{I}}_{i,j}} \mathbf{X}_L[i, \ell_{L-1}] \mathbf{X}_{L-1}[\ell_{L-1}, \ell_{L-2}] \dots \mathbf{X}_1[\ell_1, j] \tag{3.3}$$

where $E^{\mathbf{I}}_{i,j} := \{(\ell_{L-1}, \dots, \ell_1) \mid (\ell_k, \ell_{k-1}) \in I_k, \forall 1 \leq k \leq L\}$ with the convention $\ell_L = i, \ell_0 = j$. It is easy to check that a matrix $\mathbf{A}$ satisfies $\mathbf{A} \in \mathcal{L}_{\mathbf{I}}$ if, and only if, there exists matrices $\mathbf{X}_\ell$ such that $\mathbf{A}[i, j] = P^{\mathbf{I}}_{i,j}(\mathbf{X}_1, \dots, \mathbf{X}_L)$ for every i, j. In fact, we do not need to further impose $\text{supp}(\mathbf{X}_\ell) \subseteq I_\ell, \forall \ell = 1, \dots, L$ because the choice of $E^{\mathbf{I}}_{i,j}$ assures that $P^{\mathbf{I}}_{i,j}(\mathbf{X}_L, \dots, \mathbf{X}_1)$ involves only with coefficients of $\mathbf{X}_\ell$ inside the support constraints (I, J). One can thus express $\mathcal{L}_{\mathbf{I}}$ as:

$$\mathcal{L}_{\mathbf{I}} = \{\mathbf{A} \mid \exists (\mathbf{X}_\ell)_{\ell=1}^L, \sum_{(i,j)} (\mathbf{A}[i, j] - P^{\mathbf{I}}_{i,j}(\mathbf{X}_L, \dots, \mathbf{X}_1))^2 = 0\}$$

Using the projection theorem (Theorem 3.2.5), one can conclude that $\mathcal{L}_{\mathbf{I}}$ is semi-algebraic since it is the projection of the semi-algebraic set:

$$\mathcal{A} := \{\mathbf{A}, (\mathbf{X}_\ell)_{\ell=1}^L \mid \sum_{(i,j)} (\mathbf{A}[i, j] - P^{\mathbf{I}}_{i,j}(\mathbf{X}_L, \dots, \mathbf{X}_1))^2 = 0\}. \qquad \square$$

Lemma 3.2.16. *Given $\mathbf{I} = (I_L, \dots, I_1)$, the closedness of $\mathcal{L}_{\mathbf{I}}$ is decidable with an algorithm of complexity $O((sd)^{C^{k-1}})$ where $s = 2$, $d = 2L$, $k = N_L N_0 + 1 + 2\sum_{i=1}^L |L_i|$ and C a universal constant (N_L, N_0 defined as in Question 3.2.14).*

Proof. To decide whether $\mathcal{L}_{\mathbf{I}}$ is closed or not, it is equivalent to decide if the following *sentence* (see Definition 3.2.11) is true or false:

$$\begin{aligned} \exists \mathbf{A}, &(\forall \mathbf{X}_L, \dots, \mathbf{X}_1, P(\mathbf{A}, \mathbf{X}_L, \dots, \mathbf{X}_1) > 0) \wedge \\ &(\forall \epsilon > 0, \exists \mathbf{X}'_L, \dots, \mathbf{X}'_1, P(\mathbf{A}, \mathbf{X}'_L, \dots, \mathbf{X}'_1) - \epsilon < 0) \end{aligned}$$

where $P(\mathbf{A}, \mathbf{X}_1, \dots, \mathbf{X}_L) := \sum_{(i,j)} (\mathbf{A}[i, j] - P^{\mathbf{I}}_{i,j}(\mathbf{X}_L, \dots, \mathbf{X}_1))^2$.

This sentence basically asks whether there exists a matrix $\mathbf{A} \in \overline{\mathcal{L}_{\mathbf{I}}} \setminus \mathcal{L}_{\mathbf{I}}$ or not. It can be proved that this sentence can be decided to be true or false using real algebraic geometry tools (see Theorem 3.2.13), with a complexity $O\left((sd)^{C^{k-1}}\right)$ where C is a universal constant and s, d, k are the number of polynomials, the maximum degree of the polynomials and the number of variables in the sentence, respectively. Applying this to our case, we have $s = 2, d = 2L, k = N_L N_0 + 1 + 2\sum_{\ell=1}^L |I_\ell|$ (remind that $|I_\ell|$ is the total number of unmasked coefficients of $\mathbf{X}_\ell$). $\square$

As the result, we obtain:

Corollary 3.2.17. *There is an algorithm of complexity* $O\left(8^{C^k}\right)$ *where* $k := mn + 1 + 2(|I| + |J|)$ *and* C *is the universal constant in Lemma 3.2.16 that can decide whether a pair of support constraints* $(I, J) \in \{0,1\}^{m\times r} \times \{0,1\}^{n,r}$ *is well-posed or not.*

Proof. This is exactly the setting of Lemma 3.2.16 with $L = 2$. In that case, $m = N_L, n = N_0, I = I_1, J = I_2$. Substituting these values into Lemma 3.2.16 gives the proof. □

While such a doubly exponential algorithms in Lemma 3.2.16 and Corollary 3.2.17 are seemingly impractical in practice, small toy examples (for example, LU factorization of size 2×2) can be verified using Z3prover [3], a software implementing exactly the algorithm in Lemma 3.2.16. Larger factors $(\mathcal{X}_\ell)_{\ell=1}^L$ make Z3prover unable to terminate. We will further discuss about this algorithm in Section 3.4.

Another way to derive an algorithm that decides whether a pair of support constraints (I, J) is well-posed or not is the decide the existence of optimal solutions of (2.4.FSMF) corresponding to the input $(\mathbf{A}, I, J)$ for every possible matrix $\mathbf{A}$.

Lemma 3.2.18. *There exists an algorithm that can decide:*

1. *The ill-posedness/well-posedness of a triple* $(\mathbf{A}, I, J)$ *with complexity* $O\left(8^{C^k}\right)$ *where* $k = 2(|I| + |J|)$.
2. *The ill-posedness/well-posedness of a pair of support constraints* (I, J) *with complexity* $O\left(8^{C^k}\right)$ *where* $k = mn + 2(|I| + |J|)$ *(*m, n *is the size of the target matrix of* (2.4.FSMF)*).*

Proof. Similar to the proof of Lemma 3.2.16, it is sufficient to construct a sentence (cf. Definition 3.2.11) that is equivalent to the existence of an optimal solution of the instance of Equation (2.4.FSMF) corresponding to the input $(\mathbf{A}, I, J)$. Similar to the sentence used in Lemma 3.2.16, consider:

$$\exists \mathbf{X}^\star, \mathbf{Y}^\star, \forall \mathbf{X}, \mathbf{Y}, P(\mathbf{A}, \mathbf{X}^\star, \mathbf{Y}^\star) - P(\mathbf{A}, \mathbf{X}, \mathbf{Y}) \leq 0,$$

where $P(\mathbf{A}, \mathbf{X}, \mathbf{Y}) := \sum_{(i,j)} (\mathbf{A}[i, j] - P_{i,j}^{(I,J)}(\mathbf{X}, \mathbf{Y}^\top))^2$ (which is exactly the polynomial P in Lemma 3.2.16, but for $L = 2$ and $(I_2, I_1) = (I, J)$). It is clear that this sentence is true if and only if an optimal solution exists for the (2.4.FSMF) instance with input $(\mathbf{A}, I, J)$. Deciding this sentence can be done with the quantifier elimination algorithm, with complexity $O\left(8^{C^{2(|I|+|J|)}}\right)$. To decide the well-posedness/ill-posedness of (I, J), it is sufficient to consider the sentence:

$$\forall \mathbf{A} \exists \mathbf{X}^\star, \mathbf{Y}^\star, \forall \mathbf{X}, \mathbf{Y}, P(\mathbf{A}, \mathbf{X}^\star, \mathbf{Y}^\star) - P(\mathbf{A}, \mathbf{X}, \mathbf{Y}) \leq 0.$$

The above sentence basically reads as: for any matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, the input $(\mathbf{A}, I, J)$ make its corresponding instance of (2.4.FSMF) well-posed. Deciding the second sentence takes an algorithm of complexity $O\left(8^{C^{mn+2(|I|+|J|)}}\right)$, hence the results. □

The sentence used in Lemma 3.2.18 allows us to remove a constant 1 in the exponential C^k in Lemma 3.2.16. Nevertheless, the complexity remains doubly exponential.

[3] The package is developed by Microsoft research and can be found at https://github.com/Z3Prover/z3

3.3 Fixed Support Matrix Factorization is NP-hard

In this section, we prove another difficult feature of (2.4.FSMF): it is NP-hard to solve if we allow arbitrary input $(\mathbf{A}, I, J)$. To show that (2.4.FSMF) is NP-hard we use the classical technique to prove NP-hardness: reduction. Our choice of reducible problem is matrix completion with noise [GG10]. While this problem was already introduced in Section 2.3.4, we still provide it formulation in the following for convenience:

Definition 3.3.1 (Matrix completion with noise [GG10])**.** Let $\mathbf{W} \in \{0,1\}^{m\times n}$ be a binary matrix. Given $\mathbf{A} \in \mathbb{R}^{m\times n}, s \in \mathbb{N}$, the matrix completion problem (MCP) is:

$$\underset{\mathbf{X}\in\mathbb{R}^{m\times s},\mathbf{Y}\in\mathbb{R}^{n\times s}}{\textit{Minimize}} \|\mathbf{A} - \mathbf{X}\mathbf{Y}^\top\|_{\mathbf{W}}^2 = \|(\mathbf{A} - \mathbf{X}\mathbf{Y}^\top) \odot \mathbf{W}\|^2. \tag{MCP}$$

This problem is NP-hard even when $s = 1$ [GG10] by its reducibility from Maximum-Edge Biclique Problem, which is NP-complete [Pee00] as introduced in the previous section. The result of its NP-hardness is given in the following theorem

Theorem 3.3.2 (NP-hardness of matrix completion with noise [GG10])**.** *Given a binary weighting matrix $\mathbf{W} \in \{0,1\}^{m\times n}$ and $\mathbf{A} \in [0,1]^{m\times n}$, the optimization problem*

$$\underset{x\in\mathbb{R}^m, y\in\mathbb{R}^n}{\text{Minimize}} \|\mathbf{A} - \mathbf{x}\mathbf{y}^\top\|_{\mathbf{W}}^2. \tag{MCPO}$$

is called rank-one matrix completion problem (MCPO). Denote p^* the infimum of (MCPO) *and let $\epsilon = 2^{-12}(mn)^{-7}$. It is NP-hard to find an approximate solution with objective function accuracy less than ϵ, i.e. with objective value $p \le p^* + \epsilon$.*

The following lemma gives a reduction from (MCPO) to (2.4.FSMF).

Lemma 3.3.3. *For any binary matrix $\mathbf{W} \in \{0,1\}^{m\times n}$, there exist an integer r and two sets I and J such that for all $\mathbf{A} \in \mathbb{R}^{m\times n}$,* (MCPO) *and* (2.4.FSMF) *share the same infimum. I and J can be constructed in polynomial time. Moreover, if one of the problems has a known solution that provides objective function accuracy ϵ, we can find a solution with the same accuracy for the other one in polynomial time.*

Proof sketch. Up to a transposition, we can assume WLOG that $m \ge n$. We will show that with $r = n + 1 = \min(m, n) + 1$, we can find two supports I and J satisfying the conclusion of Lemma 3.3.3.

To create an instance of (2.4.FSMF) (i.e., two supports I, J) that is *equivalent* to (MCPO), we define $I \in \{0,1\}^{m\times(n+1)}$ and $J \in \{0,1\}^{n\times(n+1)}$ as follows:

$$I[i,j] = \begin{cases} 1 - \mathbf{W}[i,j] & \text{if } j \ne n \\ 1 & \text{if } j = n+1 \end{cases}, \; J[i,j] = \begin{cases} 1 & \text{if } j = i \text{ or } j = n+1 \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

Figure 3.1 illustrates an example of support constraints built from $\mathbf{W}$.

We consider the (2.4.FSMF) with the same matrix $\mathbf{A}$ and (I, J) defined as in Equation (3.4). This construction (of I and J) can clearly be made in polynomial time. Consider the coefficients $(\mathbf{X}\mathbf{Y}^\top)[i,j]$:

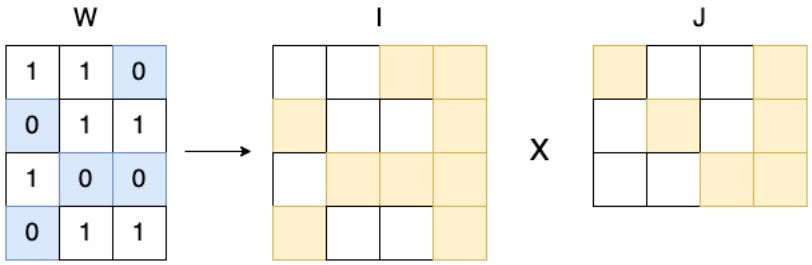


Figure 3.1: Factor supports I and J constructed from the weighted matrix $\mathbf{W} \in \{0,1\}^{4\times 3}$. Colored squares in I and J are positions in the supports.

1) If $\mathbf{W}[i,j] = 0$: $(\mathbf{X}\mathbf{Y}^\top)[i,j] = \sum_{k=1}^{n+1} \mathbf{X}[i,k]\mathbf{Y}[j,k] = \mathbf{X}[i,j]\mathbf{Y}[j,j] + X_{i,n+1}Y_{j,n+1}$ (except for $k = n+1$, only $Y_{j,j}$ can be different from zero due to our choice of J).
2) If $\mathbf{W}[i,j] = 1$: $(\mathbf{X}\mathbf{Y}^\top)[i,j] = \sum_{k=1}^{n+1} \mathbf{X}[i,k]\mathbf{Y}[j,k] = \mathbf{X}[i,n+1]\mathbf{Y}[j,n+1]$ (same reason as in the previous case, in addition to the fact that $I[i,j] = 1 - \mathbf{W}[i,j] = 0$).

Therefore, the following equation holds:

$$(\mathbf{X}\mathbf{Y}^\top) \odot \mathbf{W} = (\mathbf{X}[:,n+1]\mathbf{Y}[:,n+1]^\top) \odot \mathbf{W} \tag{3.5}$$

We will prove that (2.4.FSMF) and (MCPO) share the same infimum[4]. We denote $\mu_1 = \inf_{\mathbf{x}\in\mathbb{R}^m,\mathbf{y}\in\mathbb{R}^n} \|\mathbf{A} - \mathbf{x}\mathbf{y}^\top\|_{\mathbf{W}}^2$ and $\mu_2 = \inf_{\mathrm{supp}(\mathbf{X})\subseteq I,\mathrm{supp}(\mathbf{Y})\subseteq J} \|\mathbf{A} - \mathbf{X}\mathbf{Y}^\top\|^2$. It is clear that $\mu_i \geq 0 > -\infty, i = 1,2$. Our objective is to prove $\mu_1 \leq \mu_2$ and $\mu_2 \leq \mu_1$.

1) Proof of $\mu_1 \leq \mu_2$: By definition of an infimum, for all $\mu > \mu_1$, there exist $\mathbf{x},\mathbf{y}$ such that $\|\mathbf{A} - \mathbf{x}\mathbf{y}^\top\|_{\mathbf{W}}^2 \leq \mu$. We can choose $\mathbf{X}$ and $\mathbf{Y}$ (with $\mathrm{supp}(\mathbf{X}) \subseteq I, \mathrm{supp}(\mathbf{Y}) \subseteq J$) as follows: we take the last columns of $\mathbf{X}$ and $\mathbf{Y}$ equal to $\mathbf{x}$ and $\mathbf{y}$ ($\mathbf{X}[:,n+1] = \mathbf{x}, \mathbf{Y}[n+1,:] = \mathbf{y}$). For the *remaining* columns of $\mathbf{X}$ and $\mathbf{Y}$, we choose:

$$\begin{aligned} \mathbf{X}[i,j] &= \mathbf{A}[i,j] - \mathbf{x}_i\mathbf{y}_j \quad && \text{if } I[i,j] = 1, j \leq n \\ \mathbf{Y}[i,j] &= 1 && \text{if } J[i,j] = 1, j \leq n \end{aligned}$$

This choice of $\mathbf{X}$ and $\mathbf{Y}$ will make $\|\mathbf{A} - \mathbf{X}\mathbf{Y}^\top\|^2 = \|\mathbf{A} - \mathbf{x}\mathbf{y}^\top\|_{\mathbf{W}}^2 \leq \mu$. Indeed, for all (i,j) such that $\mathbf{W}_{i,j} = 0$, we have:

$$\begin{aligned} (\mathbf{A} - \mathbf{X}\mathbf{Y}^\top)[i,j] &= \mathbf{A}[i,j] - \mathbf{X}[i,j]\mathbf{Y}[j,j] - \mathbf{X}[i,n+1]\mathbf{Y}[j,n+1] \\ &= \mathbf{A}[i,j] - \mathbf{A}[i,j] + \mathbf{x}_i\mathbf{y}_j - \mathbf{x}_i\mathbf{y}_j = 0 \end{aligned}$$

Therefore, it is clear that: $(\mathbf{A} - \mathbf{X}\mathbf{Y}^\top) \odot (\mathbf{1} - \mathbf{W}) = \mathbf{0}$.

$$\begin{aligned} \|\mathbf{A} - \mathbf{X}\mathbf{Y}^\top\|^2 &= \|(\mathbf{A} - \mathbf{X}\mathbf{Y}^\top) \odot \mathbf{W}\|^2 + \|(\mathbf{A} - \mathbf{X}\mathbf{Y}^\top) \odot (\mathbf{1} - \mathbf{W})\|^2 \\ &= \|(\mathbf{A} - \mathbf{X}\mathbf{Y}^\top) \odot \mathbf{W}\|^2 \\ &\overset{(3.5)}{=} \|(\mathbf{A} - \mathbf{X}[:,n+1]\mathbf{Y}[:,n+1]^\top) \odot \mathbf{W}\|^2 \\ &= \|(\mathbf{A} - \mathbf{x}\mathbf{y}^\top) \odot W\|^2 \\ &= \|\mathbf{A} - \mathbf{x}\mathbf{y}^\top\|_W^2 \end{aligned}$$

[4]We focus on the infimum instead of minimum since there are cases where the infimum is not attained, as shown in chapter 3

Therefore, $\mu_2 \leq \mu_1$.

2) Proof of $\mu_1 \leq \mu_2$: Inversely, for all $\mu > \mu_2$, there exists $\mathbf{X}, \mathbf{Y}$ satisfying $\text{supp}(\mathbf{X}) \subseteq I, \text{supp}(\mathbf{Y}) \subseteq J$ such that $\|\mathbf{A} - \mathbf{X}\mathbf{Y}^\top\|^2 \leq \mu$. We choose $\mathbf{x} = \mathbf{X}[:, n+1], \mathbf{y} = \mathbf{Y}[:, n+1]$. It is immediate that:

$$\begin{aligned}
\|\mathbf{A} - \mathbf{x}\mathbf{y}^\top\|_{\mathbf{W}}^2 &= \|(\mathbf{A} - \mathbf{x}\mathbf{y}^\top) \odot \mathbf{W}\|^2 \\
&= \|(\mathbf{A} - \mathbf{X}[:, n+1]\mathbf{Y}[:, n+1]^\top) \odot \mathbf{W}\|^2 \\
&\overset{(3.5)}{=} \|(\mathbf{A} - \mathbf{X}\mathbf{Y}^\top) \odot \mathbf{W}\|^2 \\
&\leq \|(\mathbf{A} - \mathbf{X}\mathbf{Y}^\top) \odot \mathbf{W}\|^2 + \|(\mathbf{A} - \mathbf{X}\mathbf{Y}^\top) \odot (\mathbf{1} - \mathbf{W})\|^2 \\
&= \|\mathbf{A} - \mathbf{X}\mathbf{Y}^\top\|^2
\end{aligned}$$

Thus, $\|\mathbf{A} - \mathbf{x}\mathbf{y}^\top\|_{\mathbf{W}}^2 \leq \|\mathbf{A} - \mathbf{X}\mathbf{Y}^\top\|^2 \leq \mu$. We have $\mu_1 \leq \mu_2$.

This shows that $\mu_1 = \mu_2$. Moreover, the proofs of $\mu_1 \leq \mu_2$ and $\mu_2 \leq \mu_1$ also show the procedures to obtain an optimal solution of one problem with a given accuracy ϵ provided that we know an optimal solution of the other with the same accuracy. □

Using lemma 3.3.3, we obtain a result of NP-hardness for (2.4.FSMF) as follows.

Theorem 3.3.4. *When $\mathbf{A} \in [0,1]^{m \times n}$, it is NP-hard to solve (2.4.FSMF) with arbitrary index sets I, J and objective function accuracy less than $\epsilon = 2^{-12}(mn)^{-7}$.*

Proof. Given any instance of (MCPO) (i.e., two matrices $\mathbf{A} \in [0,1]^{m \times n}$ and $\mathbf{W} \in \{0,1\}^{m \times n}$), we can produce an instance of (2.4.FSMF) (the same matrix $\mathbf{A}$ and $I \in \{0,1\}^{m \times r}, J \in \{0,1\}^{n \times r}$) such that both have the same infimum (lemma 3.3.3). Moreover, for any given objective function accuracy, we can use the procedure of lemma 3.3.3 to make sure the solutions of both problems share the same accuracy.

Since all procedures are polynomial, this defines a polynomial reduction from (MCPO) to (2.4.FSMF). Because (MCPO) is NP-hard to obtain a solution with objective function accuracy less than ϵ (theorem 3.3.2), so is (2.4.FSMF). □

We point out that, while the result is interesting on its own, for some applications, such as those arising in machine learning, the accuracy bound $O((mn)^{-7})$ may not be really appealing. We thus keep as an interesting open research direction to determine if some precision threshold exists that make the general problem easy.

3.4 Open questions

In this section, we will discuss several open questions. Many of them originate from two related problems: Tensor decomposition and robust principle component analysis (cf. Section 2.3.4). Reviewing the literature, we found that these problems share the same pathology as (2.4.FSMF): the existence of an optimal solution is not guaranteed for certain instances [SL06, JAS19]. These instances will be called *pathological cases* in the following. Revisiting these problems raises many interesting questions about the well-posedness of (2.4.FSMF) in particular and sparse matrix factorization in general.

Are pathological cases rare? To get the nature of this question, we revisit the problem of tensor decomposition. The ill-posedness of tensor decomposition problem of order larger than 3 is shown in details in [SL06]. The authors showed that there exists a tensor of size $2 \times 2 \times 2$ of rank three, that can be approximated arbitrarily well by the set of rank-two tensors. This phenomenon extends to other tensor decomposition instances under two conditions [SL06]:

1. The order of the tensor has to be at least three.

2. The approximation rank is strictly bigger than one.

Note that the case of matrices can be included in the study of tensors, because matrices are tensors of order two. Notably, the first assumption does not apply in the case of LRMA that is indeed well-posed, as shown in previous section.

Moreover, the authors of [SL06] proves a stronger result for the tensor decomposition in $\mathbb{R}^{2\times2\times2}$. Note that the maximum rank of a $\mathbb{R}^{2\times2\times2}$ is three and the set of rank-three tensors of size $2 \times 2 \times 2$ contains an open set [SL06, Table 7.1, Theorem 7.1]. The stronger result for the ill-posedness of the tensor decomposition states as:

Theorem 3.4.1 (Theorem 8.1, [SL06])**.** *All tensors of rank* 3 *of* $\mathbb{R}^{2\times2\times2}$ *do not have an optimal rank-*2 *approximation (with respect to the Frobenius norm). In particular, the pathological cases of approximating a tensor in* $\mathbb{R}^{2\times2\times2}$ *by a rank-two tensor comprise an open set with positive volume.*

In words, Theorem 3.4.1 proves that the pathological cases are *not* rare since they have positive Lebesgue measure. Therefore, if one samples a random tensor of size $2 \times 2 \times 2$ from a standard Gaussian distribution, the probability of getting a pathological tensor is strictly larger than zero.

Interestingly, if we consider complex tensors (i.e., the coefficients are complex values), Theorem 3.4.1 is no longer correct. This result, shown in [QML20], states that:

Theorem 3.4.2 (Adapted from Corollary 7.5(i), [QML20])**.** *The set of complex tensors which do not admit a best rank r approximation is of Lebesgue measure zero.*[5]

Therefore, whether the pathological cases are rare or not depends on the underlying fields of the tensor. Translating these results to the setting of (2.4.FSMF), we define a matrix $\mathbf{A}$ is pathological w.r.t. an ill-posed support constraints (I, J) if $(\mathbf{A}, I, J)$ is ill-posed. In the case of (2.4.FSMF) (whose field is $\mathbb{R}$), we have not yet identified any support constraint (I, J) where an equivalent version for (2.4.FSMF) of Theorem 3.4.1 holds (i.e., the set of pathological matrices $\mathbf{A}$ for an ill-posed (I, J) has strictly positive Lebesgue measure). To illustrate this point, let's revisit the **LU** decomposition problem, which is shown to be ill-posed in Section 3.1. We will show that the set of pathological matrices are of measure Lebesgue zero. This is an immediate result of the following corollary:

Corollary 3.4.3 (Lebesgue measure of the set of matrices non admitting an **LU** decomposition)**.** *The set of matrices that do not have an* ***LU*** *decomposition has Lebesgue measure zero.*

[5] For a more complete statement of this theorem, we invite readers to consider [QML20, Section 5].

Proof. The proof comes straightforward from the fact that the set of singular square matrices has Lebesgue measure zero. Applying this fact with Theorem 3.1.9 proves Corollary 3.4.3. □

By excluding the set of matrices which do not have an (exact) **LU** decomposition, the remaining ones admit an **LU** factorization and their corresponding instances of (2.4.FSMF) with **LU** support constraints attain the infimum zero. This example motivates us to propose the following conjecture:

Conjecture 3.4.4. *For any support constraints (I, J), the set of matrices $\mathbf{A}$ such that $(\mathbf{A}, I, J)$ is ill-posed is of Lebesgue measure zero.*

Again, we emphasize that if Conjecture 3.4.4 holds, then in practice, the ill-posedness caused by (2.4.FSMF) is *not* too serious since the pathological cases rarely happen.

Does ill-posedness remain if one allows for more than just one fixed pair of support constraints? Throughout this chapter, we have seen that if we fix the support constraint (I, J), the problem can become ill-posed. However, in many situations, there are many possible supports that $(\mathbf{X}, \mathbf{Y})$ can admit, for example, in the general sparse matrix factorization problem (2.3.SMF). The last paragraph of this chapter discusses briefly the possible ill-posedness of (2.3.SMF). However, instead of constructing our own toy examples, we re-use the ill-posedness of another well-known problem of significant interest: Robust Principal Component Analysis (RPCA) [JAS19] or Section 2.3.4. In particular, we prove that RPCA can be seen as an instance of (2.3.SMF) with multiple possible supports. Combining this result with the ones in [JAS19], we arrive at the ill-posedness of the general problem (2.3.SMF).

For a detailed introduction of RPCA, we refer readers to Section 2.3.4. In this part, we only remind its formulation, which is:

$$\begin{aligned} &\underset{L,S\in\mathbb{R}^{m\times n}}{\text{Minimize}} && \|\mathbf{A}-\mathbf{S}-\mathbf{L}\|^2 \\ &\text{Subject to:} && \|\mathbf{S}\|_0 \leq s, \operatorname{rank}(\mathbf{L}) \leq r. \end{aligned} \tag{RPCA}$$

This problem is shown to suffer from the same ill-posedness phenomenon as: the set $\mathrm{LS}(s,r) := \{\mathbf{S}+\mathbf{L} \mid \|\mathbf{S}\|_0 \leq s, \operatorname{rank}(\mathbf{L}) \leq r\} \subseteq \mathbb{R}^{m\times n}$ is not closed. This result is exhibited in [JAS19]:

Theorem 3.4.5 (Ill-posedness of (RPCA) [JAS19, Theorem 2.1, 2.2])**.** *There exist (m, n, r, s) such that $LS(s, r)$ is not closed.*

In the remaining, we show that (RPCA) is a special case in the framework of sparse matrix factorization. To be more specific, we will build a family of support constraints for the problem of two-factor sparse matrix factorization (2.3.SMF) which is equivalent to (RPCA).

We consider the following two-factor (2.3.SMF):

$$\begin{aligned} &\underset{X\in\mathbb{R}^{m\times(s+r)},Y\in\mathbb{R}^{n\times(s+r)}}{\text{Minimize}} && \|\mathbf{A}-\mathbf{X}\mathbf{Y}^\top\|^2 \\ &\text{Subject to:} && \mathbf{X}\in\mathcal{E}_X \text{and } \mathbf{Y}\in\mathcal{E}_Y, \end{aligned} \tag{3.6}$$

where $\mathcal{E}_X = \{\mathbf{X} \mid \|\mathbf{X}[:,i]\|_0 \leq 1, i = 1,\ldots,s\}$ and $\mathcal{E}_Y = \{\mathbf{Y} \mid \|\mathbf{Y}[:,i]\|_0 \leq 1, i = 1,\ldots,s\}$.

The equivalence of the two formulations (RPCA) and (3.6) is shown by proving the following lemma:

Lemma 3.4.6. *For every matrix pair $(\mathbf{S},\mathbf{L})$ in the setting of (RPCA), there exists a pair of $(\mathbf{X},\mathbf{Y}) \in I \times J$ yielding the same approximation matrix (i.e., $\mathbf{S}+\mathbf{L} = \mathbf{X}\mathbf{Y}^\top$) and vice versa.*

Proof. Consider the product $\mathbf{X}\mathbf{Y}^\top$:

$$\mathbf{X}\mathbf{Y}^\top = \sum_{i=1}^{r+s} \mathbf{X}[:,i]\mathbf{Y}[:,i]^\top = \underbrace{\left(\sum_{i=1}^{s} \mathbf{X}[:,i]\mathbf{Y}[:,i]^\top\right)}_{:=\mathbf{S}'} + \underbrace{\left(\sum_{i=s+1}^{s+r} \mathbf{X}[:,i]\mathbf{Y}[:,i]^\top\right)}_{:=\mathbf{L}'} = \mathbf{S}' + \mathbf{L}'. \tag{3.7}$$

Due to the constraint imposed by $(\mathcal{E}_X,\mathcal{E}_Y)$, we have $\|\mathbf{X}[:,i]\mathbf{Y}[:,i]^\top\|_0 \leq 1, \forall 1 \leq i \leq s$. Therefore, $\|\mathbf{S}'\|_0 = \|\sum_{i=1}^{s} \mathbf{X}[:,i]\mathbf{Y}[:,i]^\top\|_0 \leq s$. Moreover, $\operatorname{rank}(\mathbf{L}') \leq r$ since $\mathbf{L}' = \sum_{i=s+1}^{s+r} \mathbf{X}[:,i]\mathbf{Y}[:,i]^\top$. Thus, for any pair of $(\mathbf{X},\mathbf{Y})$, we can choose $(\mathbf{S},\mathbf{L}) = (\mathbf{S}',\mathbf{L}')$.

Vice-versa, given any $(\mathbf{S},\mathbf{L})$ satisfying the constraints in (RPCA), we can choose $(\mathbf{X},\mathbf{Y})$ such that $\sum_{i=1}^{s} \mathbf{X}[:,i]\mathbf{Y}[:,i]^\top = \mathbf{S}$ and $\sum_{i=s+1}^{s+r} \mathbf{X}[:,i]\mathbf{Y}[:,i]^\top = \mathbf{L}$. This completes the other direction of the lemma. □

By Lemma 3.4.6, it is easy to see that (RPCA) and (3.6) share the same infimum. Moreover, knowing an optimal solution (if it exists) of one problem, one can construct an optimal solution of the other (in polynomial time). Otherwise, both problems do not admit an minimizer (i.e., they are both ill-posed). This showcases that even if we allow several possible support constraints on the factors, (2.3.SMF) is not safe from ill-posedness.

Knowing that the constraint $(\mathcal{E}_X,\mathcal{E}_Y)$ of (3.6) is quite specific, curious readers might wonder whether ill-posedness might happen for other types of support constraints. Following this natural question, we raise the following question but we left it open for future work:

Question 3.4.7. Determine if (2.3.SMF) is ill-posed or not if we impose either of the following family of support constraints:

1. k-sparse matrices: $\mathcal{M}_k^{\text{total}} = \{\mathbf{X} \mid |\operatorname{supp}(\mathbf{X})| \leq k\}$.
2. Sparsity by row/ column: $\mathcal{M}_k^{\text{row}} = \{\mathbf{X} \mid |\operatorname{supp}(\mathbf{X}[i,:])| \leq k, \forall i\}$ or $\mathcal{M}_k^{\text{col}} := \{\mathbf{X} \mid |\operatorname{supp}(\mathbf{X}[:,i])| \leq k, \forall i\}$.

To the best our knowledge, none of these questions have been addressed in the literature. An answer for them might shed the light for many improvement for (2.4.FSMF) in particular and (2.3.SMF) in general.

Is there a better algorithm to detect ill-posed instances? As seen in Section 3.2, there exists an algorithm (quantifier elimination) to decide the closedness of $\mathcal{E}_{I,J}$. However, this approach suffers from two main drawbacks:

1. Complexity: The complexity of *elimination quantifier algorithm* is doubly exponential [BPR06, Chapter 14] (for example, 2^{2^n}). It is impractical in practice.

2. Lack of insight knowledge: in fact, *elimination quantifier algorithm* acts as a black box, which does its job (determine the closedness), but reveals no more than that.

Therefore, a faster algorithm to decide the closedness is well desired. We conjecture that such an algorithm must exploit better the structure of (I, J), and thus, provide more compact *description/characterization* of the closedness of $\mathcal{E}_{I,J}$.

Can the NP-hardness result of (2.4.FSMF) be improved? Lemma 3.3.3 constructs a hard instance where $(I, J) \in \{0,1\}^{m\times r} \times \{0,1\}^{n\times r}$ and $r = \min(m, n) + 1$. It is also interesting to investigate the hardness of (2.4.FSMF) given a fixed r. When $r = 1$, the problem is polynomially tractable since this case is covered by Theorem 5.2.3 (TODO: add cref). On the other hand, when $r \geq 2$, the question becomes complicated due to the non-closedness/ ill-posedness explained in previous sections. Therefore, it is unclear whether (2.4.FSMF) is tractable if r is fixed (or equivalently, fixed-parameter tractable). We leave the proof or disproof of this question open for future researches.

Is there a link between ill-posedness and NP-hardness? So far, we consider three problems: (2.4.FSMF), tensor decomposition and robust principle component analysis. There are similarities among these three: all of them have ill-posed instances (as discuss previously). In addition, they are all NP-hard to solve (see the NP-hardness of tensor decomposition [Has90], RPCA [GV18] and (2.4.FSMF) in Section 3.3). It is thus natural to consider the relation between the two properties. One of our intuitions is that to approximate the infimum of an ill-posed instance up to an additive error ϵ, then the coefficients of the factors might diverge (exponentially) quickly to infinity when $\epsilon \to \infty$. This intuition is partly supported by Claim 3.1.1. It leads to our following conjecture:

Conjecture 3.4.8. *There exists $n_0 \in \mathbb{N}, c > 0$ such that for any ill-posed instance $(\mathbf{A}, I, J), \mathbf{A} \in \mathbb{R}^{m\times n}, |I|, |J| \geq n_0$ of (2.4.FSMF) , then for any $\epsilon > 0$, for any feasible $(\mathbf{X}_\epsilon, \mathbf{Y}_\epsilon)$ (i.e., those satisfy the support constraints (I, J)), such that:*

$$L(\mathbf{X}_\epsilon, \mathbf{Y}_\epsilon) \leq \inf_{\mathbf{X},\mathbf{Y}} L(\mathbf{X}, \mathbf{Y}) + \epsilon,$$

we have $\max(\|\mathbf{X}_\epsilon\|_F, \|\mathbf{Y}_\epsilon\|_F) \geq c \exp(1/\epsilon)$.

In words, Conjecture 3.4.8 implies that to have a pair of feasible solution that approximate the infimum of $L(\mathbf{X}, \mathbf{Y})$ up to an additive error ϵ, any algorithm needs to use at least $1/\epsilon$ bits to represent the approximate such a solution, which is not polynomial w.r.t. $\log(1/\epsilon)$ (the number of minimum bits to represent ϵ in the binary representation). If this conjecture holds, it implies there does not exist any polynomial algorithm for (2.4.FSMF)[6].

Note that if the conjecture holds, then $n_0 \geq 2$. Indeed, we consider Example 3.1.4 where $m = n = 2$, this conjecture does not hold because the instance of (2.4.FSMF) is indeed ill-posed but with $\epsilon = 1/k^4$, we only needs $\|\mathbf{X}_\epsilon\|_F, \|\mathbf{Y}_\epsilon\|_F$ to be of order $O(k)$, or equivalently $O(\sqrt[4]{1/\epsilon})$. This example does not reject Conjecture 3.4.8 because things might behave differently for growing size of the support constraints (in Example 3.1.4, all factors are of size 2×2 and all support constraints are of size 3). That is why there is

[6]Note that there might still exist algorithms which are *fixed-parameter tractable* in the sense that if we fix ϵ, then their complexity is polynomial w.r.t. other input arguments.

the quantifier $\exists n_0$ in the statement of Conjecture 3.4.8. This question is, however, left for future work.

Conclusion

In this section, we witness two difficulties when it comes to (2.4.FSMF): some instances do not even have optimal solution in Section 3.1 and solving *all* instances (2.4.FSMF) approximately up to an additive error is NP-hard in Theorem 3.3.4. Still, this is not the end of the story because certain problems such as training neural networks is proved to be NP-hard [BR92] (or even ER-complete[7] [AKM21, BHJ+22]), but they can still be solved fairly efficiently in practice. Recent study on optimization landscape reveals that training NNs admits such efficient algorithms because its objective function has a benign landscape. We will find out whether it is the case for (2.4.FSMF) in the next chapter.

[7]ER is a larger and believed to strictly contain the complexity class NP. Therefore, a problem being ER-complete is conceptually more difficult than ones being NP-hard

Chapter 4

General optimization landscape

In this chapter, we study the *landscape* of the objective function $L(\mathbf{X}, \mathbf{Y})$ of (2.4.FSMF), with arbitrary (I, J). We focus on two notions: *spurious local minima* and *spurious local valleys* (shortened to *spurious objects*), which are main challenges of non-convex optimization. We show that in general, the landscape of (2.4.FSMF) suffers from *spurious objects*. This is in sharp contrast with established work on the landscape of low rank matrix factorization and other related optimization problems (matrix completion, linear neural network training). This might explain the NP-hardness in the general case of (2.4.FSMF), which we proved in Section 3.3. This section re-tells mostly the story written in [LRG22, Section 4].

4.1 Preliminary: notions of optimization landscape

We start by recalling the classical definitions of global and local minima of a real-valued function.

Definition 4.1.1 (Spurious local minimum [ZSEW18, NW06])**.** Consider $L : \mathbb{R}^d \to \mathbb{R}$. A vector $x^* \in \mathbb{R}^d$ is a:

- **global minimum** (of L) if $L(x^*) \leq L(x), \forall x$.
- **local minimum** if there is a neighborhood $\mathcal{N}$ of x^* such that $L(x^*) \leq L(x), \forall x \in \mathcal{N}$.
- **strict local minimum** if there is a neighborhood $\mathcal{N}$ of x^* such that $L(x^*) < L(x), \forall x \in \mathcal{N}, x \neq x^*$.
- **(strict) spurious local minimum** if x^* is a (strict) local minimum but it is not a global minimum.

The presence of spurious local minima is undesirable because local optimization methods can get stuck in one of them and never reach the global optimum.

Remark 4.1.2. With the loss functions $L(\mathbf{X}, \mathbf{Y})$ of (2.4.FSMF), strict local minima do not exist since for every invertible diagonal matrix $\mathbf{D}$, possibly arbitrarily close to the identity, we have $L(\mathbf{XD}, \mathbf{YD}^{-1}) = L(\mathbf{X}, \mathbf{Y})$.

However, this is not the only undesirable landscape in an optimization problem: spurious local valleys, as defined next, are also challenging.

Definition 4.1.3 (Sublevel Set [BV04])**.** Consider $L : \mathbb{R}^d \to \mathbb{R}$. For every $\alpha \in \mathbb{R}$, the α-level set of L is the set $E_\alpha = \{x \in \mathbb{R}^d \mid L(x) \leq \alpha\}$.

Definition 4.1.4 (Path-Connected Set and Path-Connected Component)**.** A subset $S \subseteq \mathbb{R}^d$ is path-connected if for every $x, y \in S$, there is a continuous function $r : [0, 1] \to S$ such that $r(0) = x, r(1) = y$. A path-connected component of $E \subseteq \mathbb{R}^d$ is a maximal path-connected subset: $S \subseteq E$ is path-connected, and if $S' \subseteq E$ is path-connected with $S \subseteq S'$ then $S = S'$.

Definition 4.1.5 (Spurious Local Valley [VBB20, Ngu19])**.** Consider $L : \mathbb{R}^d \to \mathbb{R}$ and a set $S \subset \mathbb{R}^d$.

- S is a **local valley** of L if it is a non-empty path-connected component of some sublevel set.
- S is a **spurious local valley** of L if it is a local valley of L and does not contain a global minimum.

The notion of spurious local valley is inspired by the definition of a *strict* spurious local minimum. If x^* is a strict spurious local minimum, then $\{x^*\}$ is a spurious local valley. However, the notion of spurious local valley has a wider meaning than just a neighborhood of a strict spurious local minimum. fig. 4.1 illustrates some other scenarios: as shown on

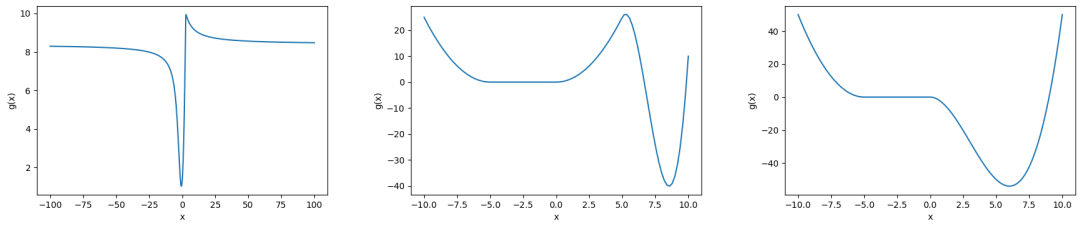


Figure 4.1: Examples of functions with spurious objects.

section 4.1, the segment (approximately) $[10, +\infty)$ creates a spurious local valley, and this function has only one local (and global) minimizer, at zero; in section 4.1, there are spurious local minima that are not strict, but form a spurious local valley anyway. It is worth noticing that the concept of a spurious local valley does *not* cover that of a spurious local minimum. Functions can have spurious (non-strict) local minima even if they do not possess any spurious local valley (section 4.1). Therefore, in this paper, we treat the existence of spurious local valleys and spurious local minima independently. The common point is that if the landscape possesses either of them, local optimization methods need to have proper initialization to have guarantees of convergence to a global minimum.

In the literature, one common technique to prove that a critical point is not a spurious local minimum is to use the notion of *strict saddle points*:

Definition 4.1.6 (Strict saddle points [ZSEW18, Definition 2])**.** A critical point x (of a twice continuously differentiable function L) is a strict saddle point if the Hessian of L evaluated at x is not positive semi-definite, or equivalently has at least one –*strictly* – negative eigenvalue).

A strict saddle point cannot be a local minimum because it does not satisfy the second-order necessary condition for a local minimum [NW06, Theorem 12.5]. Consequently, it cannot be a spurious local minimum either. To prove a function do not have spurious local minimum, it is sufficient to prove that it has the following *strict saddle property*:

Definition 4.1.7 (Strict saddle property [ZSEW18, Definition 3])**.** Consider a twice differentiable function $L : \mathbb{R}^d \to \mathbb{R}$. If each critical point of L is either a global minimum or a *strict* saddle point then L is said to have the strict saddle property. When this property holds, L has no spurious local minimum.

Even if L has the strict saddle property, it may have no global minimum, consider e.g. the function $L(x) = -\|x\|_2^2$.

On the other hand, to prove the non-existence of spurious local valleys, the following lemma was employed in previous works [VBB20, Ngu19]:

Lemma 4.1.8 (Sufficient condition for the non-existence of any spurious local valley [VBB20, Lemma 2])**.** *Consider a continuous function $L : \mathbb{R}^d \to \mathbb{R}$. Assume that, for any initial parameter $\tilde{x} \in \mathbb{R}^d$, there exists a continuous path $f : t \in [0, 1] \to \mathbb{R}^d$ such that:*

a) $f(0) = \tilde{x}$.

b) $f(1) \in \operatorname{argmin}_{x \in \mathbb{R}^d} L(x)$.

c) The function $L \circ f : t \in [0, 1] \to \mathbb{R}$ is non-increasing.

Then there is no spurious local valley in the landscape of function L.

The result is intuitive and a formal proof can be found in [VBB20]. The theorem claims that given any initial point, if one can find a continuous path connecting the initial point to a global minimizer and the loss function is non-increasing on the path, then there does not exist any spurious local valley. We remark that although (2.4.FSMF) is a constrained optimization problem, lemma 4.1.8 is still applicable because one can think of the objective function as defined on a subspace: $L : \mathbb{R}^{|I|+|J|} \to \mathbb{R}$. In our case, to apply lemma 4.1.8, the constructed function f has to be a *feasible path*, defined as:

Definition 4.1.9 (Feasible path)**.** A feasible path w.r.t the support constraints (I, J) (or simply a feasible path) is a continuous function $f(t) = (X_f(t), Y_f(t)) : [0, 1] \to \mathbb{R}^{m \times r} \times \mathbb{R}^{n \times r}$ satisfying $\operatorname{supp}(X_f(t)) \subseteq I, \operatorname{supp}(Y_f(t)) \subseteq J, \forall t \in [0, 1]$.

Conversely, we generalize and formalize an idea from [VBB20] into the following lemma, which gives a sufficient condition for the existence of a spurious local valley:

Lemma 4.1.10 (Sufficient condition for the existence of a spurious local valley)**.** *Consider a continuous function $L : \mathbb{R}^d \to \mathbb{R}$ whose global minimum is attained. Assume we know three subsets $S_1, S_2, S_3 \subset \mathbb{R}^d$ such that:*

1) The global minima of L are in S_1.

2) Every continuous path from S_3 to S_1 passes through S_2.

3) $\inf_{x\in S_2} L(x) > \inf_{x\in S_3} L(x) > \inf_{x\in S_1} L(x)$.

Then L has a spurious local valley. Moreover, any $x \in S_3$ such that $L(x) < \inf_{x\in S_2} L(x)$ is a point inside a spurious local valley.

Proof. Denote $\Sigma = \{x \mid L(x) = \inf_{x\in\mathbb{R}^d} L(\theta)\}$ the set of global minimizers of L. Σ is not empty due to the assumption that the global minimum is attained, and $\Sigma \subseteq S_1$ by the first assumption.

Since $\inf_{x\in S_2} L(x) > \inf_{x\in S_3} L(x)$, there exists $\tau \in S_3, L(\tau) < \inf_{x\in S_2} L(x)$. Consider Φ the path-connected component of the sublevel set $\{x \mid L(x) \leq L(\tau)\}$ that contains τ. Since Φ is a non-empty path-connected component of a level set, it is a local valley. It is thus sufficient to prove that $\Phi \cap \Sigma = \emptyset$ to obtain that it matches the very definition of a spurious local valley.

Indeed, by contradiction, let's assume that there exists $\tau' \in \Phi \cap \Sigma$. Since $\tau, \tau' \in \Phi$ and Φ is path-connected, by definition of path-connectedness there exists a continuous function $f : [0, 1] \to \Phi$ such that $f(0) = \tau \in S_3, f(1) = \tau' \in \Sigma \subseteq S_1$. Due to the assumption that every continuous path from S_3 to S_1 has to pass through a point in S_2, there must exist $t \in (0, 1)$ such that $f(t) \in S_2 \cap \Phi$. Therefore, $L(f(t)) \leq L(\tau)$ (since $f(t) \in \Phi$) and $L(f(t)) > L(\tau)$ (since $f(t) \in S_2$), which is a contradiction. □

4.2 Existing works on landscape of related matrix factorization problems

This section is devoted to review several works on the landscape of optimization function of matrix factorization related problems. A comparison between these results and what we are going to establish demonstrates another challenging aspect of (2.4.FSMF): using iterative methods (such as gradient descent and its variants) to optimize $L(\mathbf{X}, \mathbf{Y})$ might be also difficult to yield the optimal solution. Moreover, we introduce in detail two results that are directly related to specific instances of Equation (2.4.FSMF). They will be re-used later in Chapter 5 to prove the benign landscape of (2.4.FSMF) given *structured* support constraints (I, J).

The global landscape of the loss functions for matrix decomposition related problems (matrix sensing [BNS16, LZT18], phase retrieval [SQW16], matrix completion [GLM16, GJZ17, CL19]) and neural network training (either with linear [ZSEW18, Kaw16, VBB20] or non-linear activation functions [Ngu19, NH17]) has been a popular subject of study recently. These works have direct link to ours since matrix factorization *without any support constraint* (i.e., low rank matrix approximation) can be seen either as a matrix decomposition problem or as a specific case of neural network (with two layers, no bias and linear activation function). Notably it has been proved [ZSEW18] that for linear neural networks, every local minimum is a global minimum and if the network is shallow (i.e., there is only one hidden layer), critical points are either global minima or strict saddle points (i.e., their Hessian have at least one –*strictly*– negative eigenvalue). However, there

is still a *tricky* type of landscape that could represent a challenge for local optimization methods and has not been covered until recently: spurious local valleys [Ngu19, VBB20]. In particular, the combination of these results shows the benign landscape for LMRA, a particular instance of (2.4.FSMF).

Another line of research associated to matrix decomposition problems combines tools from probability/statistics to produce results in the following spirit: under certain random models (on the inputs), the objective function does not have spurious objects. Many existing works can be found belong to this line [BNS16, LZT18, SQW16, GLM16, GJZ17, CL19, CLC19].

However, to the best of our knowledge, existing analyses of landscape are only proposed for neural network training in general and matrix factorization problem in particular *without support constraints*, cf. [ZSEW18, VBB20, Kaw16], while the study of the landscape of (2.4.FSMF) remains untouched in the literature and our work can be considered as a generalization of such previous results.

Throughout this thesis, we will show results of two types corresponding to the question of landscape:

1. Negative results: Under certain conditions of (I, J), there exists $\mathbf{A}$ such that $L(\mathbf{X}, \mathbf{Y})$ has spurious objects. It is shown in Theorem 4.3.1, Chapter 4.

2. Positive result: Under certain conditions of (I, J), $L(\mathbf{X}, \mathbf{Y})$ does not have spurious objects for any matrix $\mathbf{A}$. They are shown in Chapter 5.

To finish this section, we introduce two *positive* results (i.r., there is no spurious objects) in the landscape of linear neural network training.

Theorem 4.2.1 (No spurious local valleys in linear networks [VBB20, Theorem 11])**.** *Consider linear neural networks of any depth $K \geq 1$ and of any layer widths $p_k \geq 1$ and any input - output dimension $n, m \geq 1$ with the following form: $\Phi(b, \theta) = \mathbf{W}_K \dots \mathbf{W}_1 b$ where $\theta = (\mathbf{W}_i)_{i=1}^K$, and $b \in \mathbb{R}^n$ is a training input sample. With the squared loss function, there is no spurious local valley. More specifically, the function $L(\theta) = \|\mathbf{A} - \Phi(\mathbf{B}, \theta)\|^2$ satisfies the condition of lemma 4.1.8 for any matrices $\mathbf{A} \in \mathbb{R}^{m \times N}$ and $\mathbf{B} \in \mathbb{R}^{n \times N}$ ($\mathbf{A}$ and $\mathbf{B}$ are the whole sets of training output and input respectively).*

Theorem 4.2.2 (No spurious local minima in shallow linear networks [ZSEW18, Theorem 3])**.** *Let $B \in \mathbb{R}^{d_0 \times N}, A \in \mathbb{R}^{d_2 \times N}$ be input and output training examples. Consider the problem:*

$$\underset{\mathbf{X} \in \mathbb{R}^{d_0 \times d_1}, \mathbf{Y} \in \mathbb{R}^{d_1 \times d_2}}{\text{Minimize}} \quad L(\mathbf{X}, \mathbf{Y}) = \|\mathbf{A} - \mathbf{XYB}\|^2$$

If $\mathbf{B}$ is full row rank, L has the strict saddle property (see Definition 4.1.7) hence L has no spurious local minimum.

Both theorems are valid for a particular case of matrix factorization with fixed support: full support matrix factorization. Indeed, given a factorized matrix $A \in \mathbb{R}^{m \times n}$, in Theorem 4.2.1, if $K = 2, B = \mathbf{I}_n$ $(n = N)$, then the considered function is $L = \|A - W_2 W_1\|^2$. This is (2.4.FSMF) without support constraints I and J (and without a transpose on W_1, which does not change the nature of the problem). Theorem 4.2.1 guarantees that L satisfies the conditions of Lemma 4.1.8, thus has no spurious local valley.

Similarly, in Theorem 4.2.2, if $B = \mathbf{I}_{d_0}$ ($d_0 = N$, therefore B is full row rank), we return to the same situation of Theorem 4.2.1. In general, Theorem 4.2.2 claims that the landscape of the full support matrix factorization problem has the strict saddle property and thus, does not have spurious local minima.

In the next section, we will show a negative results: For some (I, J), such a benign landscape is not guaranteed. This shows that the nice property in the full support case does not generally carry to others of (2.4.FSMF).

4.3 Spurious objects in the general landscape of Fixed Support Matrix Factorization

First, we establish a sufficient condition for the *existence* of a spurious local valley in (2.4.FSMF).

Theorem 4.3.1. *Consider function $L(\mathbf{X}, \mathbf{Y}) = \|\mathbf{A} - \mathbf{X}\mathbf{Y}^\top\|^2$ in (2.4.FSMF). Given two support constraints $I \in \{0,1\}^{m\times r}$, $J \in \{0,1\}^{n\times r}$, if there exist $i_1 \neq i_2 \in [\![m]\!], j_1 \neq j_2 \in [\![n]\!]$ and $k \in [\![r]\!]$ such that (i_2, j_2) belongs to at least 2 rank-one supports, one of which is $\mathcal{S}_k$, and if $(i_1, j_1), (i_2, j_1), (i_1, j_2)$ belong only to $\mathcal{S}_k$, then:*

1) There exists $\mathbf{A}$ such that: $L(\mathbf{X}, \mathbf{Y})$ has a spurious local valley.

2) There exists $\mathbf{A}$ such that: $L(\mathbf{X}, \mathbf{Y})$ has a spurious local minimum.

In both cases, $\mathbf{A}$ can be chosen so that the global minimum of $L(X, Y)$ under the considered support constraints is achieved and is zero.

Proof. Let $l \neq k$ be another rank-one contribution support $\mathcal{S}_l$ that contains (i_1, j_1). Without loss of generality, we can assume $i_1 = j_1 = 1, i_2 = j_2 = 2$ and $k = 1, l = 2$. In particular, let $I' = J' := \{(1,1), (2,1), (2,2)\}$, then $I' \subseteq I, J' \subseteq J$. When $m = n = 2$, these are the support constraints for the **LU** decomposition.

1) We define the matrix $\mathbf{A}$ by block matrices as:

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}' & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}, \text{ where } \mathbf{A}' = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \in \mathbb{R}^{2\times 2}. \tag{4.1}$$

The minimum of $L(\mathbf{X}, \mathbf{Y}) := \|\mathbf{A} - \mathbf{X}\mathbf{Y}^\top\|^2$ over feasible pairs is zero and it is attained at $\mathbf{X} = \left[\begin{smallmatrix} \mathbf{X}' & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{smallmatrix}\right], \mathbf{Y} = \left[\begin{smallmatrix} \mathbf{Y}' & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{smallmatrix}\right]$ where $\mathbf{X}' = \left[\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right], \mathbf{Y}' = \left[\begin{smallmatrix} 1 & 0 \\ 1 & -1 \end{smallmatrix}\right]$. $(\mathbf{X}, \mathbf{Y})$ is feasible since $\text{supp}(\mathbf{X}) = \text{supp}(\mathbf{X}') = I' \subseteq I, \text{supp}(\mathbf{Y}) = \text{supp}(\mathbf{Y}') = J' \subseteq J$. Moreover,

$$\mathbf{X}\mathbf{Y}^\top = \begin{pmatrix} \mathbf{X}'\mathbf{Y}'^\top & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} = \begin{pmatrix} \mathbf{A}' & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} = \mathbf{A}. \tag{4.2}$$

Using lemma 4.1.10 we now prove that this matrix $\mathbf{A}$ produces a spurious local valley for $L(\mathbf{X}, \mathbf{Y})$ with the considered support constraints (I, J). In fact, since $(1,1), (1,2), (2,1)$ are only in $\mathcal{S}_1$ and in no other support $\mathcal{S}_\ell$, $\ell \neq 1$, one can easily check that for every feasible pair $(\mathbf{X}, \mathbf{Y})$ we have:

$$(\mathbf{X}\mathbf{Y}^\top)[i, j] = \mathbf{X}[i, 1]\mathbf{Y}[j, 1], \quad \forall (i, j) \in \{(1,1), (1,2), (2,1)\}. \tag{4.3}$$

Thus, every feasible pair $(\mathbf{X}^\star, \mathbf{Y}^\star)$ reaching the global optimum $\|\mathbf{A}-\mathbf{X}^\star(\mathbf{Y}^\star)^\top\| = 0$ must satisfy:

$$\mathbf{X}^\star[1,1]\mathbf{Y}^\star[1,1] = \mathbf{X}^\star[2,1]\mathbf{Y}^\star[1,1] = \mathbf{X}^\star[1,1]\mathbf{Y}^\star[2,1] = 1$$

This implies:

$$\mathbf{X}^\star[2,1]\mathbf{Y}^\star[2,1] = \frac{(\mathbf{X}^\star_{2,1}\mathbf{Y}^\star_{1,1})(\mathbf{X}^\star_{1,1}\mathbf{Y}^\star_{2,1})}{\mathbf{X}^\star_{1,1}\mathbf{Y}^\star_{1,1}} = 1$$

Moreover, such an optimum feasible pair also satisfies $0 = \mathbf{A}[2,2] = (\mathbf{X}^\star(\mathbf{Y}^\star)^\top)[2,2] = \sum_p \mathbf{X}^\star[2,p]\mathbf{Y}^\star[2,p]$, hence $\sum_{p\neq 1} \mathbf{X}^\star[2,p]\mathbf{Y}^\star[2,p] = -\mathbf{X}^\star[2,1]\mathbf{Y}^\star[2,1] = -1$.

To show the existence of a spurious local valley we use Lemma 4.1.10 and consider the set $\tilde{S}_\sigma = \{(\mathbf{X},\mathbf{Y}) \mid \text{supp}(\mathbf{X}) \subseteq I, \text{supp}(\mathbf{Y}) \subseteq J, \sum_{p\neq 1} \mathbf{X}[2,p]\mathbf{Y}[2,p] = \sigma\}$. We will show that $S_1 := \tilde{S}_{-1}, S_2 := \tilde{S}_1, S_3 := \tilde{S}_5$ satisfy the assumptions of lemma 4.1.10.

To compute $\inf_{(\mathbf{X},\mathbf{Y})\in S_i} L(\mathbf{X},\mathbf{Y})$, we study $g(\sigma) := \inf_{(\mathbf{X},\mathbf{Y})\in \tilde{S}_\sigma} L(\mathbf{X},\mathbf{Y})$. Denoting $Z = \left[\begin{smallmatrix} \mathbf{1}_{2\times 2} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{smallmatrix}\right] \in \{0,1\}^{m\times n}$ we have:

$$\begin{aligned}
g(\sigma) &= \inf_{(\mathbf{X},\mathbf{Y})\in\tilde{S}_\sigma} \|\mathbf{A}-\mathbf{X}\mathbf{Y}^\top\|^2 \\
&\geq \inf_{(\mathbf{X},\mathbf{Y})\in\tilde{S}_\sigma} \|(\mathbf{A}-\mathbf{X}\mathbf{Y}^\top)\odot\mathbf{Z}\|^2 \\
&\overset{(4.3)}{=} \inf_{(\mathbf{X},\mathbf{Y})\in\tilde{S}_\sigma} \left\| \begin{pmatrix} \mathbf{A}[1,1]-\mathbf{X}[1,1]\mathbf{Y}[1,1] & \mathbf{A}[1,2]-\mathbf{X}[1,1]\mathbf{Y}[2,1] \\ \mathbf{A}[2,1]-\mathbf{X}[2,1]\mathbf{Y}[1,1] & \mathbf{A}[2,2]-\sigma-\mathbf{X}[2,1]\mathbf{Y}[2,1] \end{pmatrix} \right\|^2 \\
&= \inf_{\mathbf{X}[1,1],\mathbf{X}[2,1],\mathbf{Y}[1,1],\mathbf{Y}[2,1]} \left\| \begin{pmatrix} 1-\mathbf{X}[1,1]\mathbf{Y}[1,1] & 1-\mathbf{X}[1,1]\mathbf{Y}[2,1] \\ 1-\mathbf{X}[2,1]\mathbf{Y}[1,1] & -\sigma-\mathbf{X}[2,1]\mathbf{Y}[2,1] \end{pmatrix} \right\|^2
\end{aligned}$$

Besides (4.3), the third equality exploits the fact that $(\mathbf{X}\mathbf{Y}^\top)[2,2] = \sum_p \mathbf{X}[2,p]\mathbf{Y}[2,p] = \mathbf{X}[2,1]\mathbf{Y}[2,1] + \sigma$. The last quantity is the loss of the best rank-one approximation of $\tilde{\mathbf{A}} = \left[\begin{smallmatrix} 1 & 1 \\ 1 & -\sigma \end{smallmatrix}\right] \in \mathbb{R}^{2\times 2}$. Since this is a 2×2 symmetric matrix, its eigenvalues can be computed as the solutions of a second degree polynomial, leading to an analytic expression of this last quantity as: $\frac{2(\sigma+1)^2}{(\sigma^2+3)+\sqrt{(\sigma^2+3)^2-4(\sigma+1)^2}}$.

Moreover, this infimum can be attained if $\big[\mathbf{X}[1,1],\mathbf{X}[2,1]\big]$ and $\big[\mathbf{Y}[1,1],\mathbf{Y}[2,1]\big]$ is the first eigenvectors of $\tilde{\mathbf{A}}$ and the other coefficients of $\mathbf{X},\mathbf{Y}$ are set to zero. Therefore,

$$g(\sigma) = \frac{2(\sigma+1)^2}{(\sigma^2+3)+\sqrt{(\sigma^2+3)^2-4(\sigma+1)^2}}. \tag{4.4}$$

We can now verify that S_1, S_2, S_3 satisfy all the conditions of lemma 4.1.10.

1) The minimum value of L is zero. As shown above, it is only attained with $\sum_{p\neq 1} \mathbf{X}^\star[2,p]\mathbf{Y}^\star[2,p] = -1$ as shown. Thus, the global minima belong to $S_1 = \tilde{S}_{-1}$.
2) For any feasible path $r : [0,1] \to \mathbb{R}^{m\times r} \times \mathbb{R}^{n\times r} : t \to (X(t), Y(t))$ we have $\sigma_r(t) = \sum_{p\neq 1} X(t)[2,p]Y(t)[2,p]$ is also continuous. If $(X(0),Y(0)) \in S_3 = \tilde{S}_5$ and $(X(1),Y(1)) \in S_1 = \tilde{S}_{-1}$ then $\sigma_r(0) = 5$ and $\sigma_r(1) = -1$), hence by the

Mean Value Theorem, there must exist $t \in (0,1)$ such that $\sigma_r(t) = 1$, which means $(X(t), Y(t)) \in S_2 = \tilde{S}_1$.

3) Since one can check numerically that $g(1) > g(5) > g(-1)$, we have

$$\inf_{(\mathbf{X},\mathbf{Y})\in S_2} L(\mathbf{X},\mathbf{Y}) > \inf_{(\mathbf{X},\mathbf{Y})\in S_3} L(\mathbf{X},\mathbf{Y}) > \inf_{(\mathbf{X},\mathbf{Y})\in S_1} L(\mathbf{X},\mathbf{Y}).$$

The proof is concluded with the application of lemma 4.1.10. In addition, any point $(\mathbf{X}, \mathbf{Y})$ satisfying $\sigma = 5$ and $L(\mathbf{X}, \mathbf{Y}) < g(1) = 2$ is inside a spurious local valley. For example, one of such a point is $\mathbf{X} = \left[\begin{smallmatrix} \mathbf{X}' & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{smallmatrix}\right], Y = \left[\begin{smallmatrix} Y' & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{smallmatrix}\right]$ where $\mathbf{X}' = \left[\begin{smallmatrix} 1 & 0 \\ -5 & 1 \end{smallmatrix}\right], \mathbf{Y}' = \left[\begin{smallmatrix} -1/5 & 0 \\ 1 & 5 \end{smallmatrix}\right]$.

2) We define the matrix $\mathbf{A}$ by block matrices as:

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}' & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}, \text{ where } \mathbf{A}' = \begin{pmatrix} b & 0 \\ 0 & a \end{pmatrix} \in \mathbb{R}^{2\times 2}. \tag{4.5}$$

where $a > b > 0$. It is again evident that the infimum of $\|\mathbf{A} - \mathbf{X}\mathbf{Y}^\top\|^2$ under the considered support constraints is zero, and is achieved (taking $\mathbf{X} = \left[\begin{smallmatrix} \mathbf{X}' & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{smallmatrix}\right], \mathbf{Y} = \left[\begin{smallmatrix} \mathbf{Y}' & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{smallmatrix}\right]$ where $\mathbf{X}' = \left[\begin{smallmatrix} b & 0 \\ 0 & a \end{smallmatrix}\right], \mathbf{Y}' = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right]$ and with the same proof as in Equation (4.2), we have $\mathbf{X}\mathbf{Y}^\top = \mathbf{A}$.)

Now, we will consider $\tilde{\mathbf{X}} = \left[\begin{smallmatrix} \mathbf{X}' & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{smallmatrix}\right], \tilde{\mathbf{Y}} = \left[\begin{smallmatrix} \mathbf{Y}' & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{smallmatrix}\right]$ where $\mathbf{X}' = \left[\begin{smallmatrix} 0 & 0 \\ a & 0 \end{smallmatrix}\right], \mathbf{Y}' = \left[\begin{smallmatrix} 0 & 0 \\ 1 & 0 \end{smallmatrix}\right]$. Since $L(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}) = b^2 > 0$ it cannot be a global minimum. We will show that $(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$ is indeed a local minimum, which will thus imply that $(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$ is a spurious local minimum. For each feasible pair $(\mathbf{X}, \mathbf{Y})$ we have:

$$\begin{aligned}
&\|\mathbf{A} - \mathbf{X}\mathbf{Y}^\top\|^2 = \sum_{i,j} (\mathbf{A} - \mathbf{X}\mathbf{Y}^\top)[i,j]^2 \\
&\geq (\mathbf{A} - \mathbf{X}\mathbf{Y}^\top)[1,1]^2 + (\mathbf{A} - \mathbf{X}\mathbf{Y}^\top)[1,2]^2 + (\mathbf{A} - \mathbf{X}\mathbf{Y}^\top)[2,1]^2 \\
&\overset{(4.3)}{=} (b - \mathbf{X}[1,1]\mathbf{Y}[1,1])^2 + (\mathbf{X}[2,1]\mathbf{Y}[1,1])^2 + (\mathbf{X}[1,1]\mathbf{Y}[2,1])^2 \\
&\geq (\mathbf{X}[1,1]\mathbf{Y}[1,1])^2 - 2b\mathbf{X}[1,1]\mathbf{Y}[1,1] + b^2 + 2(\mathbf{X}[2,1]\mathbf{Y}[2,1])|\mathbf{X}[1,1]\mathbf{Y}[1,1]| \\
&\geq 2(\mathbf{X}[2,1]\mathbf{Y}[2,1] - b)|\mathbf{X}[1,1]\mathbf{Y}[1,1]| + b^2.
\end{aligned}$$

where in the third line we used that for $u = |\mathbf{X}[2,1]|\mathbf{Y}[1,1]|$, $v = \mathbf{X}[1,1]|\mathbf{Y}[2,1]|$, since $(u - v)^2 \geq 0$ we have $u^2 + v^2 \geq 2uv$. Since $\tilde{\mathbf{X}}[2,1]\tilde{\mathbf{Y}}[2,1] = a > b$, there exists a neighborhood of $(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$ such that $\mathbf{X}[2,1]\mathbf{Y}[2,1] - b > 0$ for all $(\mathbf{X}, \mathbf{Y})$ in that neighbourhood. Since $|\mathbf{X}[1,1]\mathbf{Y}[1,1]| \geq 0$ in this neighborhood, it follows that $\|\mathbf{A} - \mathbf{X}\mathbf{Y}^\top\|^2 \geq b^2 = L(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}) > 0$ in that neighborhood. This concludes the proof.

□

Remark 4.3.2. Theorem 4.3.1 is constructed based on the **LU** structure. We elaborate our intuition on the technical proof of Theorem 4.3.1 as follows: Consider the **LU** decomposition problem of size 2×2 (i.e., $I = J = \{(1,1),(2,1),(2,2)\}$). It is obvious that such (I, J) satisfies the assumptions of Theorem 4.3.1 (for $i_1 = j_1 = 1, i_2 = j_2 = 2$). We consider three matrices of size 2×2:

$$\mathbf{A}_1 = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}, \ \mathbf{A}_2 = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}, \ \mathbf{A}_3 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

$\mathbf{A}_1$ (resp. $\mathbf{A}_2$) is simply the matrix $\mathbf{A}'$ in (4.1) (resp. in (4.5), with $a = 2, b = 1$) in the proof of Theorem 4.3.1. $\mathbf{A}_3$ is a matrix which does not admit an $\mathbf{LU}$ decomposition. We plot the graphs of $g_i(\sigma) = \inf_{X_{2,2}Y_{2,2}=\sigma} \|\mathbf{A}_i - \mathbf{X}\mathbf{Y}^\top\|$ (this is exactly $g(\sigma)$ introduced in the proof of Theorem 4.3.1) in Figure 4.2.

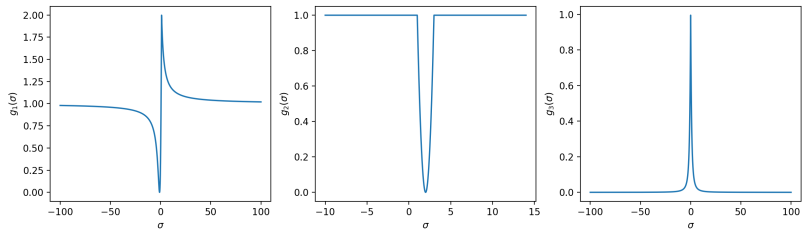


Figure 4.2: Illustration of the functions $g_i(\sigma), i = 1, 2, 3$ from left to right.

In particular, the spurious local valley constructed in the proof of Theorem 4.3.1 with $\mathbf{A}_1$ is a spurious local valley extending to infinity. With $\mathbf{A}_2$, one can see that $g_2(\sigma)$ has a plateau with value $1 = b^2$. The local minimum that we consider in the proof of Theorem 4.3.1 is simply a point in this plateau (where $\sigma = 0$). Lastly, since the matrix $\mathbf{A}_3$ does not admit an $\mathbf{LU}$ decomposition, there is no optimal solution. Nevertheless, the infimum zero can be approximated with arbitrary precision when σ tends to infinity (two valleys extending to $\pm\infty$).

For the cases with the matrices $\mathbf{A}_1$ and $\mathbf{A}_3$, once initialized inside the valleys of their landscapes, any sequence $(\mathbf{X}_k, \mathbf{Y}_k)$ with sufficiently small steps associated to a decreasing loss $L(\mathbf{X}_k, \mathbf{Y}_k)$ will have the corresponding parameter σ converging to infinity. As a consequence, at least one parameter of either $\mathbf{X}_k$ or $\mathbf{Y}_k$ has to diverge. This is thus a setting in which PALM (and other optimization algorithms which seek to locally decrease their objective function in a monotone way) can diverge.

The existence of spurious local valleys shown in theorem 4.3.1 highlights the importance of initialization: if an initial point is already inside a spurious valley, first-order methods cannot escape this suboptimal area. An optimist may wonder if there nevertheless exist a smart initialization that avoids all spurious local valleys initially. The answer is positive, as shown in the following theorem.

Theorem 4.3.3. *Given any $(I, J), \mathbf{A}$ such that the infimum of (2.4.FSMF) is attained, every initialization $(\mathbf{X}, \mathbf{0}), \mathrm{supp}(\mathbf{X}) \subseteq I$ (or symmetrically $(\mathbf{0}, \mathbf{Y}), \mathrm{supp}(\mathbf{Y}) \subseteq J$) is not in any spurious local valley. In particular, $(\mathbf{0}, \mathbf{0})$ is never in any spurious local valley.*

Proof. Let $(\mathbf{X}^*, \mathbf{Y}^*)$ be a minimizer of (2.4.FSMF), which exists due to our assumptions. We only prove the result for the initialization $(\mathbf{X}, \mathbf{0}), \mathrm{supp}(\mathbf{X}) \subseteq I$. The case of the initialization $(\mathbf{0}, \mathbf{Y}), \mathrm{supp}(\mathbf{Y}) \subseteq J$ can be dealt with similarly.

To prove the theorem, it is sufficient to construct $f(t) = (X_f(t), Y_f(t)) : [0, 1] \to \mathbb{R}^{m\times r} \times \mathbb{R}^{n\times r}$ as a feasible path such that:

1) $f(0) = (\mathbf{X}, \mathbf{0})$.

2) $f(1) = (\mathbf{X}^*, \mathbf{Y}^*)$.

3) $L \circ f$ is non-increasing w.r.t t.

Indeed, if such f exists, the sublevel set corresponding to $L(\mathbf{X}, \mathbf{0})$ has both $(\mathbf{X}, \mathbf{0})$ and $(\mathbf{X}^*, \mathbf{Y}^*)$ in the same path-connected components (since $L \circ f$ is non-increasing).

We will construct such a function feasible path f as a concatenation of two functions feasible paths $f_1 : [0, 1/2] \to \mathbb{R}^{m\times r} \times \mathbb{R}^{n\times r}$, $f_2 : [1/2, 1] \to \mathbb{R}^{m\times r} \times \mathbb{R}^{n\times r}$, defined as follows:

1) $f_1(t) = ((1 - 2t)\mathbf{X} + 2t\mathbf{X}^*, \mathbf{0})$.

2) $f_2(t) = (\mathbf{X}^*, (2t - 1)\mathbf{Y}^*)$.

It is obvious that $f(0) = f_1(0) = (\mathbf{X}, \mathbf{0})$ and $f(1) = f_2(1) = (\mathbf{X}^*, \mathbf{Y}^*)$. Moreover f is continuous since $f_1(1/2) = f_2(1/2) = (\mathbf{X}^*, \mathbf{0})$. Also, $L \circ f$ is non-increasing on $[0, 1]$ since:

1) $L(f_1(t)) = \|\mathbf{A} - ((1 - 2t)\mathbf{X} + 2t\mathbf{X}^*)\mathbf{0}^\top\|^2 = \|\mathbf{A}\|^2$ is constant for $t \in [0, 1/2]$.

2) $L(f_2(t)) = \|\mathbf{A} - (2t - 1)\mathbf{X}^*\mathbf{Y}^*\|^2$ is convex w.r.t t. Moreover, it attains a global minimum at $t = 1$ (since we assume that $(\mathbf{X}^*, \mathbf{Y}^*)$ is a global minimizer of (2.4.FSMF)). As a result, $t \mapsto L(f_2(t))$ is non-increasing on $[1/2, 1]$.

□

Yet, such an initialization does not guarantee that first-order methods converge to a global minimum. Indeed, while in the proof of this result we do show that there exists a feasible path joining this "smart" initialization to an optimal solution without increasing the loss function, the value of the objective function is "flat" in the first part of this feasible path. Thus, even if such initialization is completely outside any spurious local valley, it is not clear whether local information at the initialization allows to "guide" optimization algorithms towards the global optimum to blindly find such a path. In fact, first-order methods are not bound to follow our constructive continuous path.

Conclusion

This chapter concludes the first part of this thesis: the challenges of (2.4.FSMF). So far, we can see that (2.4.FSMF) is challenging because if one allows *arbitrary support constraints* (I, J), then:

1. Optimal solutions of instances of (2.4.FSMF) are not bound to exist (cf. Section 3.1).
2. Finding the optimal values of (2.4.FSMF) up to an additive value ϵ is NP-hard (cf. Section 3.3).
3. The general landscape of Equation (2.4.FSMF) can have spurious local valleys and spurious local minima. This is a real challenge for optimization methods based on first-order information (cf. Section 4.3).

This is definitely not the end of the story (otherwise, I would not have enough material to defend my Ph.D.). So far, many questions related to (2.4.FSMF) that we studied are challenging because we consider them with arbitrary support constraints. One might wonder if things become more "manageable" if the pair of support constraints (I, J) admits certain structures. This is exactly the setting that we are going to deal with in the next chapter.

Part II

Fixed support matrix factorization with structures: polynomial solvability and applications

Chapter 5

Polynomially solvable structures and their benign landscape

In this chapter, our main results focus on the tractability of (2.4.FSMF). More precisely, despite its NP-hardness proved Section 3.3 in the general setting, there exists many families of support constraints (I, J) with practical interest in which (2.4.FSMF) can be solved optimally with a polynomial algorithm. This chapter will start with such a polynomially solvable instance of (I, J) - low rank matrix approximation (or full support matrix factorization). This fact was quickly mentioned in Section 2.3.4, but it will be elaborated in this chapter. It turns out that the support constraints of full support matrix factorization is not the only one making (2.4.FSMF) polynomially solvable. In Section 5.2, we describe a more generalized family of tractable instances of (2.4.FSMF). More surprisingly, despite the general landscape that might possess spurious local valleys and spurious local minima Chapter 4, under the family of structured support constraints introduced in Section 5.2, these spurious objects are guaranteed to disappear. The main contents of this chapter is taken from [LRG22].

5.1 Preliminary: Low Rank Matrix Approximation and Singular Value Decomposition

We first remind the readers of the problem of low rank matrix approximation/full support matrix factorization, which is:

$$\underset{\mathbf{X}\in\mathbb{R}^{m\times r},\mathbf{Y}\in\mathbb{R}^{n\times r}}{\text{Minimize}} \quad \|\mathbf{A}-\mathbf{X}\mathbf{Y}^\top\|_F^2. \tag{5.1}$$

As discussed in Section 2.3.4, this is a particular case of (2.4.FSMF) where $I = \mathbf{1}_{m\times r}, J = \mathbf{1}_{n\times r}$. Since the set $\{\mathbf{X}\mathbf{Y}^\top \mid \mathbf{X}\in\mathbb{R}^{m\times r}, \mathbf{Y}\in\mathbb{R}^{n\times r}\}$ is exactly the set of matrices of rank at most r and of size $m\times n$, we can rewrite Equation (5.1) in the following form:

$$\underset{\mathbf{B}\in\mathbb{R}^{m\times n},\text{rank}(\mathbf{B})\leq r}{\text{Minimize}} \quad \|\mathbf{A}-\mathbf{B}\|_F^2. \tag{5.2}$$

As also discussed in Section 2.3.4, finding the closest rank-r matrix to a given matrix $\mathbf{A}$ is polynomially tractable via Singular Value Decomposition (SVD). Indeed, assume that

the SVD of $\mathbf{A}$ is:

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$$

where $\mathbf{U} \in \mathbb{R}^{m\times m}$ and $\mathbf{V} \in \mathbb{R}^{n\times n}$ are orthogonal matrices, $\mathbf{D} \in \mathbb{R}^{m\times n}$ is a generalized diagonal matrix with positive and decreasingly sorted diagonal elements (i.e., $D[1,1] \geq D[2,2] \geq \ldots \geq \mathrm{D}[\min(m,n), \min(m,n)] \geq 0$), then the best approximated rank-r matrix of the matrix $\mathbf{A}$ (or equivalently, an optimal solution of (5.2))is given by:

$$\mathbf{B}^* = \mathbf{U}[:,T]\mathbf{D}[T,T]\mathbf{V}[:,T]^\top$$

where $T = \{1,\ldots,\min(m,n,r)\}$. *Translating* this into (5.1), we obtain an optimal pair of factors $(\mathbf{X}^*, \mathbf{Y}^*)$:

$$\mathbf{X} = \mathbf{U}[:,T]\mathbf{D}[T,T], \qquad \mathbf{Y} = \mathbf{V}[:,T]$$

Note that this pair of optimal factors is not unique. As long as $\mathbf{X}\mathbf{Y}^\top = \mathbf{B}^*$, $(\mathbf{X},\mathbf{Y})$ yields an optimal solution of (5.1).

Since the Singular Value Decomposition can be calculated to machine precision in $O(mn^2)$ [KKS16] (and also [TBI97, Lecture 31, Page 236]), it is convenient to think of LRMA as polynomially solvable/tractable. Applying SVD for general support constraints (I,J) is not possible because one generally cannot control the supports of $\mathbf{U},\mathbf{V}$. Thus, the solution obtained by SVD might be not feasible. Also, due to the NP-hardness proved in Section 3.3, it is unlikely to this approach to work for arbitrary support constraints (otherwise, this thesis provides a proof for P vs NP millennium problem). Thus, the question that we are interested in this chapter is restricted to:

Question 5.1.1. Under which conditions of (I,J), there exists a polynomial algorithm (possibly based on SVD) that can solve (2.4.FSMF) to optimality.

In this chapter, we provide two sufficient conditions for (I,J) in Question 5.1.1. In fact, the instances corresponding to those "tractable" (I,J) can be dealt with by algorithms using (multiple) block-wise SVD. These algorithms will be introduced in Section 5.2. In addition, we also investigate the landscape associated to these instances, showing the absence of spurious local valleys and spurious local minima, in contrast to what have been demonstrated in Chapter 4.

5.2 Tractable instances of Fixed Support Matrix Factorization

Motivated by the tractability of the full support case $I = [\![m]\!] \times [\![r]\!], J = [\![n]\!] \times [\![r]\!]$, we devote this section to enlarge the family of supports for which (2.4.FSMF) can be solved by an effective direct algorithm based on blockwise SVDs. We start with an important definition:

Definition 5.2.1 (Support of rank-one contribution)**.** Given two support constraints $I \in \{0,1\}^{m\times r}$ and $J \in \{0,1\}^{n\times r}$ of (2.4.FSMF) and $k \in [\![r]\!]$, we define the kth rank-one contribution support $\mathcal{S}_k(I,J)$ (or in short, $\mathcal{S}_k$) as: $\mathcal{S}_k(I,J) = I[:,k]J[:,k]^\top$. This can be seen either as: a tensor product: $\mathcal{S}_k \in \{0,1\}^{m\times n}$ is a binary matrix or a Cartesian product: $\mathcal{S}_k$ is a set of matrix indices defined as $\mathrm{supp}(I[:,k]) \times \mathrm{supp}(J[:,k])$.

Given a pair of support constraints (I, J), if $\mathtt{supp}(\mathbf{X}) \subseteq I, \mathtt{supp}(\mathbf{Y}) \subseteq J$, we have: $\mathtt{supp}(\mathbf{X}[:,k]\mathbf{Y}[:,k]^\top) \subseteq \mathcal{S}_k, \ \forall k \in [\![r]\!]$. Since $\mathbf{X}\mathbf{Y}^\top = \sum_{k=1}^r \mathbf{X}[:,k]\mathbf{Y}[:,k]^\top$ the notion of contribution support $\mathcal{S}_k$ captures the constraint on the support of the k^{th} *rank-one contribution*, $\mathbf{X}[:,k]\mathbf{Y}[:,k]^\top$, of the matrix product $\mathbf{X}\mathbf{Y}^\top$ (illustrated in Figure 5.1).

In the case of full supports ($\mathcal{S}_k = \mathbf{1}_{m\times n}$ for each $k \in [\![r]\!]$), the optimal solution can be obtained in a greedy manner: indeed, it is well known that Algorithm 1 computes factors achieving the best rank-r approximation to A (notice that here the algorithm also works for complex-valued matrices):

Algorithm 1 Generic Greedy Algorithm

Require: $\mathbf{A} \in \mathbb{R}^{m\times n}$ or $\mathbb{C}^{m\times n}$; $\{\mathcal{S}_k\}_{k\in[\![r]\!]}$ rank-one supports
1: **for** $i \in [\![r]\!]$ **do**
2: $\quad(\mathbf{X}[:,i], \mathbf{Y}[:,i]) = (u, v)$ where $uv^\top$ is any best rank-one approximation to $\mathbf{A} \odot \mathcal{S}_i$
3: $\quad\mathbf{A} = \mathbf{A} - \mathbf{X}[:,i], \mathbf{Y}[:,i]^\top$
4: **end for**
5: **return** $(\mathbf{X}, \mathbf{Y})$

Even beyond the full support case, the output of Algorithm 1 always satisfies the support constraints due to line 2, however it may not always be the optimal solution of (2.4.FSMF). Our analysis of the polynomial tractability conducted below will allow us to show that, under appropriate assumptions on I, J, one can compute in polynomial time an optimal solution of (2.4.FSMF) using variants of Algorithm 1. The definition of these variants will involve a partition of $[\![r]\!]$ in terms of equivalence classes of rank-one supports:

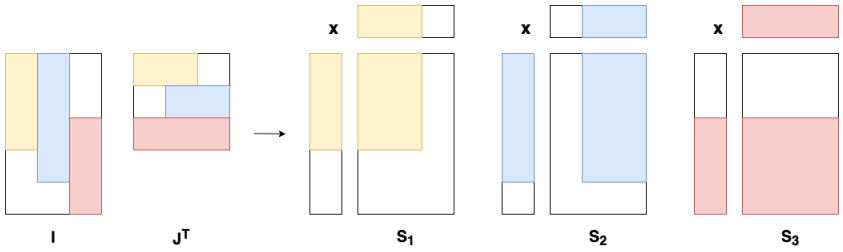


Figure 5.1: Illustration the idea of support of rank-one contribution. Colored rectangles indicate the support constraints (I, J) and the support constraints $\mathcal{S}_k$ on each component matrix $\mathbf{X}[:,k]\mathbf{Y}[:,k]^\top$.

Definition 5.2.2 (Equivalence classes of rank-one supports, representative rank-one supports)**.** Given $I \in \{0,1\}^{m\times r}$, $J \in \{0,1\}^{n\times r}$, define an equivalence relation on $[\![r]\!]$ as: $i \sim j$ if and only if $\mathcal{S}_i = \mathcal{S}_j$ (or equivalently $(I[:,i], J[:,i]) = (I_{:,j}, J_{:,j})$). This yields a partition of $[\![r]\!]$ into equivalence classes.

Denote $\mathcal{P}$ the collection of equivalence classes. For each class $P \in \mathcal{P}$ denote $\mathcal{S}_P$ a representative rank-one support, $R_P \subseteq [\![m]\!]$ and $C_P \subseteq [\![n]\!]$ the supports of rows and columns in $\mathcal{S}_P$, respectively. For every $k \in P$ we have $\mathcal{S}_k = \mathcal{S}_P$ and $\mathrm{supp}(I[:,k]) = R_P$, $\mathrm{supp}(J[:,k]) = C_P$.

For every $\mathcal{P}' \subseteq \mathcal{P}$ denote $\mathcal{S}_{\mathcal{P}'} = \cup_{P\in\mathcal{P}'}\mathcal{S}_P \subseteq [\![m]\!] \times [\![n]\!]$ and $\bar{\mathcal{S}}_{\mathcal{P}'} = ([\![m]\!] \times [\![n]\!])\backslash\mathcal{S}_{\mathcal{P}'}$.

For instance, in the example in Figure 5.1 we have three distinct equivalence classes. With the introduction of equivalence classes, one can modify Algorithm 1 to make it more efficient, as in Algorithm 2: Instead of computing the SVD r times, one can simply compute it only $|\mathcal{P}|$ times. For the full support case, we have $\mathcal{P} = \{[\![r]\!]\}$, thus Algorithm 2 is identical to the classical SVD.

Algorithm 2 Alternative Generic Greedy Algorithm

Require: $\mathbf{A} \in \mathbb{R}^{m\times n}$ or $\mathbb{C}^{m\times n}$; $\{\mathcal{S}_P\}_{P\in\mathcal{P}}$ representative rank-one supports

1: **for** $P \in \mathcal{P}$ **do**
2: $\quad(\mathbf{X}[:,P], \mathbf{Y}[:,P]) = (\mathbf{U}, \mathbf{V})$ where $\mathbf{U}\mathbf{V}^\top$ is any best rank-$|P|$ approximation to $\mathbf{A} \odot \mathcal{S}_P$
3: $\quad\mathbf{A} = \mathbf{A} - \mathbf{X}[:,P], \mathbf{Y}[:,P]^\top$
4: **end for**
5: **return** $(\mathbf{X}, \mathbf{Y})$

A first simple sufficient condition ensuring the tractability of an instance of (2.4.FSMF) is stated in the following theorem.

Theorem 5.2.3. *Consider $I \in \{0,1\}^{m\times r}$, $J \in \{0,1\}^{n\times r}$, and $\mathcal{P}$ the collection of equivalence classes of Definition 5.2.2. If the representative rank-one supports are pairwise disjoint, i.e., $\mathcal{S}_P \cap \mathcal{S}_{P'} = \emptyset$ for each distinct $P, P' \in \mathcal{P}$, then matrix factorization with fixed support is tractable for any $\mathbf{A} \in \mathbb{R}^{m\times n}$.*

Proof. In this proof, for each equivalent class $P \in \mathcal{P}$ (Definition 5.2.2) we use $\mathbf{X}_P := \mathbf{X}[:,P] \in \mathbb{R}^{m\times r}, \mathbf{Y}_P := \mathbf{Y}[:,P] \in \mathbb{R}^{n\times r}$ ($\mathbf{X}[:,P]$ is introduced in Notation) to shorten the notations. We also use the notations R_P, C_P (Definition 5.2.2). For each equivalent class P, we have:

$$(\mathbf{X}_P\mathbf{Y}_P^\top)[R_P, C_P] = \mathbf{X}[R_P, P]\mathbf{Y}[C_P, P]^\top \tag{5.3}$$

and the product $\mathbf{X}\mathbf{Y}^\top$ can be decomposed as: $\mathbf{X}\mathbf{Y}^\top = \sum_{P\in\mathcal{P}} \mathbf{X}_P\mathbf{Y}_P^\top$. Due to the hypothesis of this theorem, with $P, P' \in \mathcal{P}, P' \neq P$, we further have:

$$\mathbf{X}_{P'}\mathbf{Y}_{P'}^\top \odot \mathcal{S}_P = \mathbf{0} \tag{5.4}$$

The objective function $L(\mathbf{X}, \mathbf{Y})$ is:

$$\begin{aligned}
\|\mathbf{A} - \mathbf{X}\mathbf{Y}^\top\|^2 &= \left(\sum_{P\in\mathcal{P}} \|(\mathbf{A} - \mathbf{X}\mathbf{Y}^\top) \odot \mathcal{S}_P\|^2\right) + \|(\mathbf{A} - \mathbf{X}\mathbf{Y}^\top) \odot \bar{\mathcal{S}}_\mathcal{P}\|^2 \\
&= \left(\sum_{P\in\mathcal{P}} \|(\mathbf{A} - \sum_{P'\in\mathcal{P}} \mathbf{X}_{P'}\mathbf{Y}_{P'}^\top) \odot \mathcal{S}_P\|^2\right) + \|(\mathbf{A} - \sum_{P'\in\mathcal{P}} \mathbf{X}_{P'}\mathbf{Y}_{P'}^\top) \odot \bar{\mathcal{S}}_\mathcal{P}\|^2 \\
&\overset{(5.4)}{=} \left(\sum_{P\in\mathcal{P}} \|(\mathbf{A} - \mathbf{X}_P\mathbf{Y}_P^\top) \odot \mathcal{S}_P\|^2\right) + \|\mathbf{A} \odot \bar{\mathcal{S}}_\mathcal{P}\|^2 \\
&= \left(\sum_{P\in\mathcal{P}} \|\mathbf{A}[R_P, C_P] - (\mathbf{X}_P\mathbf{Y}_P^\top)[R_P, C_P]\|^2\right) + \|\mathbf{A} \odot \bar{\mathcal{S}}_\mathcal{P}\|^2 \\
&\overset{(5.3)}{=} \left(\sum_{P\in\mathcal{P}} \|\mathbf{A}[R_P, C_P] - \mathbf{X}[R_P, P]\mathbf{Y}[C_P, P]^\top\|^2\right) + \|\mathbf{A} \odot \bar{\mathcal{S}}_\mathcal{P}\|^2
\end{aligned} \tag{5.5}$$

Therefore, if we ignore the constant term $\|\mathbf{A} \odot \bar{\mathcal{S}}_\mathcal{P}\|^2$, the function $L(\mathbf{X}, \mathbf{Y})$ is decomposed into a sum of functions $\|\mathbf{A}[R_P, C_P] - \mathbf{X}[R_P, P]\mathbf{Y}[C_P, P]^\top\|^2$, which are LRMA instances. Since all the optimized parameters are $\{(\mathbf{X}[R_P, P], \mathbf{Y}[C_P, P])\}_{P\in\mathcal{P}}$, an optimal solution of L is $\{(\mathbf{X}^\star[R_P, P], \mathbf{Y}^\star[C_P, P])\}_{P\in\mathcal{P}}$, where $(\mathbf{X}^\star[R_P, P], \mathbf{Y}^\star[C_P, P])$ is a minimizer of $\|\mathbf{A}_{R_P,C_P} - \mathbf{X}[R_P, P]\mathbf{Y}[C_P, P]^\top\|^2$ which is computed efficiently using a truncated SVD. Since the blocks associated to distinct P are disjoint, these SVDs can be performed blockwise, in any order, and even in parallel. □

Figure 5.2 illustrates the tractable condition described in Theorem 5.2.3. For these easy instances, we can therefore recover the factors in polynomial time with the procedure described in Algorithm 3. Given a target matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ and support constraints $I \in \{0,1\}^{m\times r}$, $J \in \{0,1\}^{n\times r}$ satisfying the condition in Theorem 5.2.3, Algorithm 3 returns two factors $(\mathbf{X}, \mathbf{Y})$ solution of (2.4.FSMF).

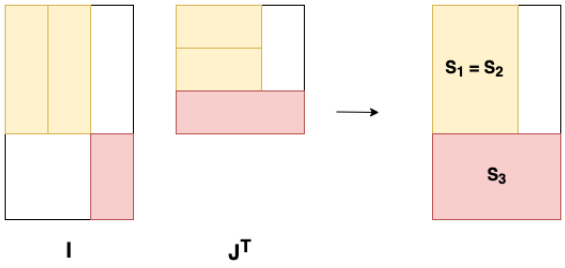


Figure 5.2: An instance of support constraints (I, J) satisfying Theorem 5.2.3. We use colored rectangles to indicate the support constraints (I, J). The indices belonging to the same equivalence class share the same color.

Algorithm 3 Fixed support matrix factorization (under Theorem 5.2.3 assumptions)

procedure SVD_FSMF($\mathbf{A} \in \mathbb{R}^{m\times n}, I \in \{0,1\}^{m\times r}, J \in \{0,1\}^{n\times r}$)
  Partition $[\![r]\!]$ into $\mathcal{P}$ (Definition 5.2.2) to get $\{\mathcal{S}_P\}_{P\in\mathcal{P}}$
  return $(\mathbf{X}, \mathbf{Y})$ using Algorithm 2 with input $\mathbf{A}$, $\{\mathcal{S}_P\}_{P\in\mathcal{P}}$
end procedure

As simple as this condition is, it is satisfied in some important cases, for instance for a class of Hierarchical matrices (HODLR, cf. Definition 2.4.1), or for the so-called *butterfly supports*: in the latter case, the condition is used in Chapter 6 to design an efficient hierarchical factorization method, which is shown to outperform first-order optimization approaches commonly used in this context, in terms both of computational time and accuracy. For the former, we show that finding the projection of a target matrix $\mathbf{A}$ onto the set of HODLR matrices (Definition 2.4.1) is equivalent to solve an instance of (2.4.FSMF) that satisfies the conditions of Theorem 5.2.3 in the following.

Lemma 5.2.4. *For each $N \geq 1$ there exists $I, J \in \{0,1\}^{2^N\times(3\times 2^N-2)}$ support constraints such that for any HODLR matrix $\mathbf{H} \in \mathbb{R}^{2^N\times 2^N}$, we have:*

1) $\mathbf{H}$ *admits a factorization* $\mathbf{X}\mathbf{Y}^\top$ *and* $\mathrm{supp}(\mathbf{X}) \subseteq I, \mathrm{supp}(\mathbf{Y}) \subseteq J$.

2) $|I| = |J| = O(n \log n)$ *($n = 2^N$).*

3) (I, J) *satisfies the assumption of Theorem 5.2.3.*

Proof. The proof can be found in Section 5.5.1. □

What if we allow partial intersection between two representative rank-one contribution supports? In the next result, we explore the tractability of (2.4.FSMF) beyond the conditions in Theorem 5.2.9.

Definition 5.2.5 (Complete equivalence classes of rank-one supports - CEC)**.** $P \in \mathcal{P}$ is a *complete equivalence class* (or *CEC*) if $|P| \geq \min\{|C_P|, |R_P|\}$ with C_P, R_P as in Definition 5.2.2. Denote $\mathcal{P}^\star \subseteq \mathcal{P}$ the family of all complete equivalence classes, $T = \cup_{P\in\mathcal{P}^\star} P \subseteq [\![r]\!]$, $\bar{T} = [\![r]\!]\backslash T$, and the shorthand $\mathcal{S}_T = \mathcal{S}_{\mathcal{P}^\star}$.

The interest of complete equivalence classes is that their expressivity is powerful enough to represent any matrix whose support is included in $\mathcal{S}_T$, as illustrated by the following lemma. Its proof will be deferred to Section 5.5.2.

Lemma 5.2.6. *Given $I \in \{0,1\}^{m\times r}$, $J \in \{0,1\}^{n\times r}$, consider T, $\mathcal{S}_T$ as in Definition 5.2.5. For any matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ such that $\mathrm{supp}(\mathbf{A}) \subseteq \mathcal{S}_T$, there exist $\mathbf{X} \in \mathbb{R}^{m\times r}, \mathbf{Y} \in \mathbb{R}^{n\times r}$ such that $\mathbf{A} = \mathbf{X}\mathbf{Y}^\top$ and $\mathrm{supp}(\mathbf{X}) \subseteq I_T$, $\mathrm{supp}(\mathbf{Y}) \subseteq J_T$. Such a pair can be computed using Algorithm 3: $(\mathbf{X}, \mathbf{Y}) = $ SVD_FSMF$(\mathbf{A}, I_T, J_T)$.*

The next definition introduces the key properties that the indices $k \in [\![r]\!]$ which are not in any CEC need to satisfy in order to make (2.4.FSMF) overall tractable.

Definition 5.2.7 (Rectangular support outside CECs of rank-one supports)**.** Given $I \in \{0,1\}^{m\times r}$, $J \in \{0,1\}^{n\times r}$, consider T and $\mathcal{S}_T$ as in Definition 5.2.5 and $\bar{T} = [\![r]\!] \setminus T$. For $k \in \bar{T}$ define the support outside CECs of the k^{th} rank-one support. as: $\mathcal{S}'_k = \mathcal{S}_k \setminus \mathcal{S}_T$. If $\mathcal{S}'_k = R_k \times C_k$ for some $R_k \subseteq [\![m]\!], C_k \subseteq [\![n]\!]$, (or equivalently $\mathcal{S}'_k$ is of rank at most one), we say the support outside CECs of the k^{th} rank-one support $\mathcal{S}'_k$ is *rectangular*.

To state our tractability result, we further categorize the indices in I and J as follows:

Definition 5.2.8 (Taxonomy of indices of I and J)**.** With the notations of Definition 5.2.7, assume that $\mathcal{S}'_k$ is rectangular for all $k \in \bar{T}$. We decompose the indices of I (resp J) into three sets as follows:

	Classification for I	Classification for J
1	$I_T = \{(i,k) \mid k \in T, i \in [\![m]\!]\} \cap I$	$J_T = \{(j,k) \mid k \in T, j \in [\![n]\!]\} \cap J$
2	$I^1_{\bar{T}} = \{(i,k) \mid k \notin T, i \in R_k\} \cap I$	$J^1_{\bar{T}} = \{(j,k) \mid k \notin T, j \in C_k\} \cap J$
3	$I^2_{\bar{T}} = \{(i,k) \mid k \notin T, i \notin R_k\} \cap I$	$J^2_{\bar{T}} = \{(j,k) \mid k \notin T, j \notin C_k\} \cap J$

The following theorem generalizes Theorem 5.2.3.

Theorem 5.2.9. *Consider $I \in \{0,1\}^{m\times r}$, $J \in \{0,1\}^{n\times r}$. Assume that for all $k \in \bar{T}$, $\mathcal{S}'_k$ is rectangular and that for all $k, l \in \bar{T}$ we have $\mathcal{S}'_k = \mathcal{S}'_l$ or $\mathcal{S}'_k \cap \mathcal{S}'_l = \emptyset$. Then, $(I^1_{\bar{T}}, J^1_{\bar{T}})$ satisfy the assumptions of Theorem 5.2.3. Moreover, for any matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, two instances of (2.4.FSMF) with data $(\mathbf{A}, I, J)$ and $(\mathbf{A} \odot \bar{\mathcal{S}}_T, I^1_{\bar{T}}, J^1_{\bar{T}})$ respectively, share the same infimum. Given an optimal solution of one instance, we can construct the optimal solution of the other in polynomial time. In other word, (2.4.FSMF) with $(\mathbf{A}, I, J)$ is polynomially tractable.*

The proof of Theorem 5.2.9 can be found in Section 5.5.3. Theorem 5.2.9 implies that solving the problem with support constraints (I, J) can be achieved by reducing to another problem, with support constraints satisfying the assumptions of Theorem 5.2.3. The latter problem can thus be efficiently solved by Algorithm 3. In particular, Theorem 5.2.3 is a special case of Theorem 5.2.9 when all the equivalent classes (including CECs) have disjoint representative rank-one supports.

Figure 5.3 shows an instance of (I, J) satisfying the assumptions of Theorem 5.2.9. The extension in Theorem 5.2.9 is not directly motivated by concrete examples, but it is rather introduced as a first step to show that the family of polynomially tractable supports (I, J) can be enlarged, as it is not restricted to just the family introduced in Theorem 5.2.3.

An algorithm for instances satisfying the assumptions of Theorem 5.2.9 is given in Algorithm 4. The correctness of Algorithm 4 is followed by Corollary 5.5.3 and Remark 5.5.4. More specifically, in Algorithm 4, we choose $(\mathbf{X}^2_{\bar{T}}, \mathbf{Y}^2_{\bar{T}}) = (\mathbf{0}, \mathbf{0})$ implicitly as well as $\mathbf{X}_T\mathbf{Y}_T^\top = \mathbf{A} \odot \mathcal{S}_T$ explicitly (Line 3, using Lemma 5.2.6) as in Remark 5.5.4, and compute $(\mathbf{X}^1_{\bar{T}}, \mathbf{Y}^1_{\bar{T}})$ to be equal to an optimal solution of (2.4.FSMF) with input $(\mathbf{A} \odot \bar{\mathcal{S}}_T, I^1_{\bar{T}}, J^1_{\bar{T}})$ (Line 4).

In Algorithm 4, two calls to Algorithm 3 are made, they can be done in any order (Line 3 and Line 4 can be switched without changing the result).

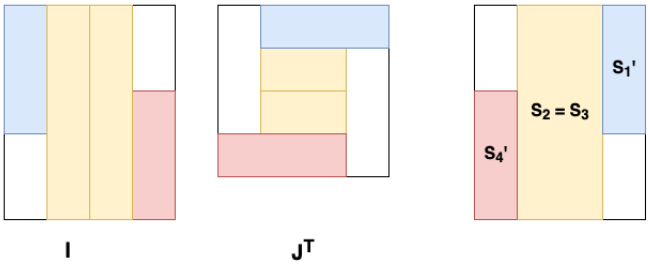


Figure 5.3: An instance of support constraints (I, J) satisfying the assumptions of Theorem 5.2.9. We have $T = \{2, 3\}$. The supports outside CEC $\mathcal{S}'_1$ and $\mathcal{S}'_4$ are disjoint.

Algorithm 4 Fixed support matrix factorization (under Theorem 5.2.9's assumptions)

procedure SVD_FSMF2($\mathbf{A} \in \mathbb{R}^{m\times n}, I \in \{0,1\}^{m\times r}, J \in \{0,1\}^{n\times r}$)
  Partition the indices of I, J into $I_T, I^1_{\bar{T}}, I^2_{\bar{T}}$ (and $J_T, J^1_{\bar{T}}, J^2_{\bar{T}}$) (Definition 5.2.7).
  $(\mathbf{X}_T, \mathbf{Y}_T) = \text{SVD\_FSMF}(\mathbf{A} \odot \mathcal{S}_T, I_T, J_T)$ ($T, \mathcal{S}_T$ as in Definition 5.2.5).
  $(\mathbf{X}^1_{\bar{T}}, \mathbf{Y}^1_{\bar{T}}) = \text{SVD\_FSMF}(\mathbf{A} \odot \bar{\mathcal{S}_T}, I^1_{\bar{T}}, J^1_{\bar{T}})$
  return $(\mathbf{X}_T + \mathbf{X}^1_{\bar{T}}, \mathbf{Y}_T + \mathbf{Y}^1_{\bar{T}})$
end procedure

5.3 Benign optimization landscape of tractable Fixed Support Matrix Factorization

We start with the first result on the landscape in the simple setting of Theorem 5.2.3.

Theorem 5.3.1. *Under the assumption of Theorem 5.2.3, the function $L(\mathbf{X}, \mathbf{Y})$ in (2.4.FSMF) does not admit any spurious local valley for any matrix $\mathbf{A}$. In addition, L has the strict saddle property.*

Proof. Recall that under the assumption of Theorem 5.2.3, all the variables to be optimized are decoupled into "blocks" $\{(\mathbf{X}[R_P, P], \mathbf{Y}[C_P, P])\}_{P\in\mathcal{P}}$ ($P, \mathcal{P}$ are defined in Definition 5.2.2). We denote $\mathcal{P} = \{P_1, P_2, \ldots, P_\ell\}$, $P_i \subseteq [\![r]\!]$, $1 \le i \le \ell$. From Equation (5.5), we have:

$$\|\mathbf{A} - \mathbf{X}\mathbf{Y}^\top\|^2 = \left(\sum_{P\in\mathcal{P}} \|\mathbf{A}_{R_P,C_P} - \mathbf{X}[R_P, P]\mathbf{Y}[C_P, P]^\top\|^2\right) + \|\mathbf{A} \odot \bar{\mathcal{S}_\mathcal{P}}\|^2 \tag{5.6}$$

Therefore, the function $L(\mathbf{X}, \mathbf{Y})$ is a sum of functions $L_P(\mathbf{X}[R_P, P], \mathbf{Y}[C_P, P])$ defined as $\|\mathbf{A}[R_P, C_P] - \mathbf{X}[R_P, P]\mathbf{Y}[C_P, P]^\top\|^2$, which do *not* share parameters and are instances of the full support matrix factorization problem restricted to the corresponding blocks in $\mathbf{A}$. The global minimizers of L are $\{(\mathbf{X}^\star[R_P, P], \mathbf{Y}^\star[C_P, P])\}_{P\in\mathcal{P}}$, where for each $P \in \mathcal{P}$ the pair $(\mathbf{X}^\star[R_P, P], \mathbf{Y}^\star[C_P, P])$ is any global minimizer of $\|\mathbf{A}[R_P, C_P] - \mathbf{X}[R_P, P]\mathbf{Y}[C_P, P]^\top\|^2$.

1) **Non-existence of any spurious local valley**: By Theorem 4.2.1, from any initial point $(\mathbf{X}^0[R_P, P], \mathbf{Y}^0[C_P, P])$, there exists a continuous function $f_P(t) = (\tilde{\mathbf{X}}_P(t), \tilde{\mathbf{Y}}_P(t)) : [0, 1] \mapsto \mathbb{R}^{|R_P|\times|P|} \times \mathbb{R}^{|C_P|\times|P|}$ satisfying the conditions in Lemma 4.1.8, which are:

i) $f_P(0) = (\mathbf{X}^0[R_P, P], \mathbf{Y}^0[C_P, P])$.

ii) $f_P(1) = (\mathbf{X}^\star[R_P, P], \mathbf{Y}^\star[C_P, P])$.

iii) $L_P \circ f_P : [0,1] \to \mathbb{R}$ is non-increasing.

Consider a feasible path (Definition 4.1.9) $f(t) = (\tilde{\mathbf{X}}(t), \tilde{\mathbf{Y}}(t)) : [0,1] \mapsto \mathbb{R}^{m\times r} \times \mathbb{R}^{r\times n}$ defined in such a way that $\tilde{\mathbf{X}}(t)[R_P, P] = \tilde{\mathbf{X}}_P(t)$ for each $P \in \mathcal{P}$ and similarly for $\tilde{\mathbf{Y}}(t)$. Since $L \circ f = \sum_{P\in\mathcal{P}} L_P \circ f_P + \|\mathbf{A} \odot \bar{\mathcal{S}_\mathcal{P}}\|^2$, f satisfies the assumptions of Lemma 4.1.8, which shows the non-existence of any spurious local valley.

2) **Non-existence of any spurious local minimum**: Due to the decomposition in Equation (5.6), the gradient and Hessian of $L(\mathbf{X}, \mathbf{Y})$ have the following form:

$$\frac{\partial L}{\partial \mathbf{X}[R_P, P]} = \frac{\partial L_P}{\partial \mathbf{X}[R_P, P]}, \qquad \frac{\partial L}{\partial \mathbf{Y}[C_P, P]} = \frac{\partial L_P}{\partial \mathbf{Y}[C_P, P]}, \ \forall P \in \mathcal{P}$$

$$H(L)_{|(\mathbf{X},\mathbf{Y})} \begin{pmatrix} H(L_{P_1})_{|(\mathbf{X}[R_{P_1},P_1],\mathbf{Y}[C_{P_1},P_1]))} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & H(L_{P_\ell})_{|(\mathbf{X}[R_{P_\ell},P_\ell],\mathbf{Y}[C_{P_\ell},P_\ell]))} \end{pmatrix}$$

Consider a critical point $(\mathbf{X}, \mathbf{Y})$ of $L(\mathbf{X}, \mathbf{Y})$ that is not a global minimizer. Since $(\mathbf{X}, \mathbf{Y})$ is a critical point of $L(\mathbf{X}, \mathbf{Y})$, $(\mathbf{X}_{R_P,P}, \mathbf{Y}_{C_P P})$ is a critical point of the function L_P for all $P \in \mathcal{P}$. Since $(\mathbf{X}, \mathbf{Y})$ is not a global minimizer of $L(\mathbf{X}, \mathbf{Y})$, there exists $P \in \mathcal{P}$ such that $(\mathbf{X}[R_P, P], \mathbf{Y}[C_P, P])$ is not a global minimizer of L_P. By Theorem 4.2.2, $H(L_P)_{|(\mathbf{X}[R_P,P],\mathbf{Y}[C_P,P])}$ is not positive semi-definite. Hence, $H(L)_{|(\mathbf{X},\mathbf{Y})}$ is not positive semi-definite either (since $H(L)_{|(\mathbf{X},\mathbf{Y})}$ has block diagonal form). This implies that $(\mathbf{X}, \mathbf{Y})$ it is a strict saddle point as well (hence, not a spurious local minimum).

□

For spurious local valleys, we have the same results for the setting in Theorem 5.2.9. The proof is, however, less straightforward.

Theorem 5.3.2. *If I, J satisfy the assumptions of Theorem 5.2.9, then for each matrix $\mathbf{A}$ the landscape of $L(\mathbf{X}, \mathbf{Y})$ in (2.4.FSMF) has no spurious local valley.*

The following is a concept which will be convenient for the proof of Theorem 5.3.2.

Definition 5.3.3 (CEC-full-rank)**.** A feasible point $(\mathbf{X}, \mathbf{Y})$ is said to be *CEC-full-rank* if $\forall P \in \mathcal{P}^\star$, either $\mathbf{X}[R_P, P]$ or $\mathbf{Y}[C_P, P]$ is full row rank.

We need three following lemmas to prove Theorem 5.3.2:

Lemma 5.3.4. *Given $I \in \{0,1\}^{m\times r}$, $J \in \{0,1\}^{n\times r}$, consider T and $\mathcal{S}_T$ as in Definition 5.2.2 and a feasible point $(\mathbf{X}, \mathbf{Y})$. There exists a feasible path $f : [0,1] \to \mathbb{R}^{m\times r} \times \mathbb{R}^{n\times r} : f(t) = (\mathbf{X}_f(t), \mathbf{Y}_f(t))$ such that:*

1) f connects $(\mathbf{X}, \mathbf{Y})$ with a CEC-full-rank point: $f(0) = (\mathbf{X}, \mathbf{Y})$, and $f(1)$ is CEC-full-rank.

2) $\mathbf{X}_f(t)(\mathbf{Y}_f(t))^\top = \mathbf{X}\mathbf{Y}^\top, \forall t \in [0,1]$.

Lemma 5.3.5. *Under the assumption of Theorem 5.2.9, for any CEC-full-rank feasible point* $(\mathbf{X}, \mathbf{Y})$*, there exists feasible path* $f : [0,1] \to \mathbb{R}^{m\times r} \times \mathbb{R}^{n\times r} : f(t) = (\mathbf{X}_f(t), \mathbf{Y}_f(t))$ *such that:*

1) $f(0) = (\mathbf{X}, \mathbf{Y})$.

2) $L \circ f$ *is non-increasing.*

3) $(\mathbf{A} - \mathbf{X}_f(1)(\mathbf{Y}_f(1))^\top) \odot \mathcal{S}_T = \mathbf{0}$.

Lemma 5.3.6. *Under the assumption of Theorem 5.2.9, for any CEC-full-rank feasible point* $(\mathbf{X}, \mathbf{Y})$ *satisfying:* $(\mathbf{A} - \mathbf{X}\mathbf{Y}^\top) \odot \mathcal{S}_T = \mathbf{0}$*, there exists a feasible path* $f : [0,1] \to \mathbb{R}^{m\times r} \times \mathbb{R}^{n\times r} : f(t) = (\mathbf{X}_f(t), \mathbf{Y}_f(t))$ *such that:*

1) $f(0) = (\mathbf{X}, \mathbf{Y})$.

2) $L \circ f$ *is non-increasing.*

3) $f(1)$ *is an optimal solution of* L.

The proofs of Lemma 5.3.4, Lemma 5.3.5 and Lemma 5.3.6 can be found in Section 5.6.2, Section 5.6.3 and Section 5.6.4 respectively.

Proof of Theorem 5.3.2. Given any initial point $(\mathbf{X}^0, \mathbf{Y}^0)$, Lemma 5.3.4 shows the existence of a continuous path along which the product of $\mathbf{X}\mathbf{Y}^\top = \mathbf{X}^0(\mathbf{Y}^0)^\top$ does not change (thus, $L(\mathbf{X}, \mathbf{Y})$ is constant) and ending at a CEC-full-rank point. Therefore it is sufficient to prove the theorem under the additional assumption that $(\mathbf{X}^0, \mathbf{Y}^0)$ is CEC-full-rank. With this additional assumption, one can employ Lemma 5.3.5 to build a continuous path $f_1(t) = (\mathbf{X}_1(t), \mathbf{Y}_1(t))$, such that $t \mapsto L(\mathbf{X}_1(t), \mathbf{Y}_1(t))$ is non-increasing, that connects $(\mathbf{X}^0, \mathbf{Y}^0)$ to a point $(\mathbf{X}^1, \mathbf{Y}^1)$ satisfying:

$$(\mathbf{A} - \mathbf{X}^1(\mathbf{Y}^1)^\top) \odot \mathcal{S}_T = \mathbf{0}.$$

Again, one can assume that $(\mathbf{X}^1, \mathbf{Y}^1)$ is CEC-full-rank (one can invoke Lemma 5.3.4 one more time). Therefore, $(\mathbf{X}^1, Y^1)$ satisfies the conditions of Lemma 5.3.6 . Hence, there exists a continuous path $f_2(t) = (\mathbf{X}_2(t), \mathbf{Y}_2(t))$ that makes $L(\mathbf{X}_2(t), \mathbf{Y}_2(t))$ non-increasing and that connects $(\mathbf{X}^1, \mathbf{Y}^1)$ to $(\mathbf{X}^*, \mathbf{Y}^*)$, a global minimizer.

Finally, since the concatenation of f_1 and f_2 satisfies the assumptions of Lemma 4.1.8, we can conclude that there is no spurious local valley in the landscape of $\|\mathbf{A} - \mathbf{X}\mathbf{Y}^\top\|^2$. □

The next natural question is whether spurious local minima exist in the setting of Theorem 5.2.9. While in the setting of Theorem 5.2.3, all critical points which are not global minima are saddle points, the setting of Theorem 5.2.9 allows second order critical points (point whose gradient is zero and Hessian is positive semi-definite), which are not global minima.

Example 5.3.7. Consider the following pair of support contraints I, J and factorized matrix $I = \left[\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}\right]$, $J = \left[\begin{smallmatrix} 1 & 1 \\ 1 & 1 \end{smallmatrix}\right]$, $\mathbf{A} = \left[\begin{smallmatrix} 10 & 0 \\ 0 & 1 \end{smallmatrix}\right]$. With the notations of Definition 5.2.5 we have $T = \{1\}$ and one can check that this choice of I and J satisfies the assumptions of Theorem 5.2.9. The infimum of $L(\mathbf{X}, \mathbf{Y}) = \|\mathbf{A} - \mathbf{X}\mathbf{Y}^\top\|^2$ is zero, and attained, for example

at $\mathbf{X}^* = I_2, \mathbf{Y}^* = A$. Consider the following feasible point $(\mathbf{X}_0, \mathbf{Y}_0)$: $\mathbf{X}_0 = \left[\begin{smallmatrix} 0 & 1 \\ 0 & 0 \end{smallmatrix}\right]$, $\mathbf{Y}_0 = \left[\begin{smallmatrix} 0 & 10 \\ 0 & 0 \end{smallmatrix}\right]$. Since $\mathbf{X}_0\mathbf{Y}_0^\top = \left[\begin{smallmatrix} 10 & 0 \\ 0 & 0 \end{smallmatrix}\right] \neq \mathbf{A}$, $(\mathbf{X}_0, \mathbf{Y}_0)$ is not a global optimal solution. Calculating the gradient of L verifes that $(\mathbf{X}_0, \mathbf{Y}_0)$ is a critical point:

$$\nabla L(\mathbf{X}_0, \mathbf{Y}_0) = ((\mathbf{A} - \mathbf{X}_0\mathbf{Y}_0^\top)\mathbf{Y}_0, (\mathbf{A}^\top - \mathbf{Y}_0\mathbf{X}_0^\top)\mathbf{X}_0) = (\mathbf{0}, \mathbf{0})$$

Nevertheless, the Hessian of the function L at $(\mathbf{X}_0, \mathbf{Y}_0)$ is positive semi-definite. Direct calculation of the Hessian of L at point $(\mathbf{X}_0, \mathbf{Y}_0)$ is given by:

$$H(L)_{|(\mathbf{X}_0, \mathbf{Y}_0)} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 100 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

which is indeed positive semi-definite.

This example shows that if we want to prove the non-existence of spurious local minima in the new setting, one cannot rely on the Hessian. This is challenging since the second order derivatives computation is already tedious. Nevertheless, with Definition 5.3.3, we can still say something about spurious local minima in the new setting.

Theorem 5.3.8. *Under the assumptions of Theorem 5.2.9, if a feasible point $(\mathbf{X}, \mathbf{Y})$ is CEC-full-rank, then $(\mathbf{X}, \mathbf{Y})$ is not a spurious local minimum of (2.4.FSMF). Otherwise there is a feasible path, along which $L(\cdot, \cdot)$ is constant, that joins $(\mathbf{X}, \mathbf{Y})$ to some $(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$ which is not a spurious local minimum.*

When $(\mathbf{X}, \mathbf{Y})$ is *not* CEC-full-rank, the theorem guarantees that it is not a strict local minimum, since there is path starting from $(\mathbf{X}, \mathbf{Y})$ with constant loss. This should however not be a surprise in light of Remark 4.1.2: indeed, the considered loss function admits no strict local minimum at all. Yet, the path with "flat" loss constructed in the theorem is fundamentally different from the ones naturally due to scale invariances of the problem and captured by Remark 4.1.2. Further work would be needed to investigate whether this can be used to get a stronger result.

Before proving Theorem 5.3.8, we provide a sketch of our proof. This allows readers to better understand our techniques in the actual proof.

Proof sketch. To prove this theorem, we proceed through two main steps:

1) First, we show that any local minimum satisfies:

$$(\mathbf{A} - \mathbf{X}\mathbf{Y}^\top) \odot \mathcal{S}_T = \mathbf{0} \tag{5.7}$$

2) Second, we show that if a point $(\mathbf{X}, \mathbf{Y})$ is CEC-full-rank and satisfies Equation (5.7), it cannot be a spurious local minimum.

Combining the above to steps, we obtain as claimed that if a feasible pair $(\mathbf{X}, \mathbf{Y})$ is CEC-full-rank, then it is not a spurious local minimum. Finally, if a feasible pair $(\mathbf{X}, \mathbf{Y})$ is not CEC-full-rank, Lemma 5.3.4 yields a feasible path along which L is constant that joins $(\mathbf{X}, \mathbf{Y})$ to some feasible $(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$ which is CEC-full-rank, hence (as we have just shown) not a spurious local mimimum. □

A formal proof of Theorem 5.3.8 in Section 5.6.5. Although Theorem 5.3.8 does not exclude completely the existence of spurious local minima, together with Theorem 5.3.1, we eliminate a large number of such points.

5.4 Numerical illustration: landscape and behaviour of gradient descent

As a numerical illustration of the practical impact of our results, we compare the performance of Algorithm 4 to other popular first-order methods on problem (2.4.FSMF).

We consider two types of instances of (2.4.FSMF): $I_1 = \mathbf{1}_{2^a \times 2^a} \otimes \mathbf{I}_{2^b \times 2^b}, J_1 = \mathbf{I}_{2^a \times 2^a} \otimes \mathbf{1}_{2^b \times 2^b}$ where $\otimes$ denotes the Kronecker product, $a = \lceil N/2 \rceil, b = \lfloor N/2 \rfloor$ (hence $a + b = N$) and $I_2 = \mathbf{1}_{2 \times 2} \otimes \mathbf{I}_{2^{N-1}}, J_2 = \mathbf{I}_2 \otimes \mathbf{1}_{2^{N-1} \times 2^{N-1}}$. These supports are interesting because they are those taken at the first two steps of the hierarchical algorithm that we will explain later in Chapter 6. These support constraints will be explained in detail (also in Chapter 6). The first pair of support constraints (I_1, J_1) is also equivalent to the so-called Monarch parameterization [DCS+22b]. Both pairs (I_1, J_1) and (I_2, J_2) are proved to satisfy Theorem 5.2.3 [ZGR21, Lemma 3.15].

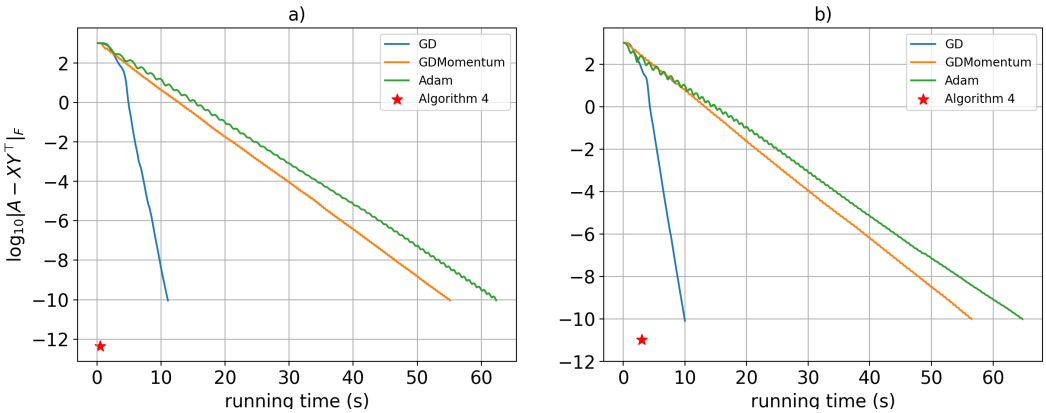


Figure 5.4: Evolution of $\log_{10} \|\mathbf{A} - \mathbf{X}\mathbf{Y}^\top\|_F$ for three variants of gradient descent and Algorithm 4 with support constraints (I_1, J_1) (left) and (I_2, J_2) (right) for $N = 10$.

We consider $\mathbf{A}$ as the Hadamard matrix of size $2^N \times 2^N$, which is known to admit an exact factorization with each of the considered support constraints, and we employ Algorithm 4 to factorize A in these two settings. We compare Algorithm 4 to three variants of gradient descent: vanilla gradient descent (GD), gradient descent with momentum (GDMomentum) and ADAM [GBC16, Chapter 8]. We use the efficient implementation of these iterative algorithms available in Pytorch 1.11.

For each matrix size 2^N, learning rates for iterative methods are tuned by grid search: we run all the factorizations with all learning rates in $\{5 \times 10^{-k}, 10^{-k} \mid k = 1, \dots, 4\}$. Matrix $\mathbf{X}$ (resp. $\mathbf{Y}$) is initialized with i.i.d. random coefficients inside its support I (resp. J) drawn according to the law $\mathcal{N}(0, 1/R_I)$ (resp. $\mathcal{N}(0, 1/R_J)$) where R_I, R_J are respectively the number of elements in each column of I and of J. All these experiments are run on an Intel Core i7 CPU 2,3 GHz. In the interest of reproducible research, our implementation is available in open source [LGR22].

Since $\mathbf{A}$ admits an exact factorization with both the supports (I_1, J_1) and (I_2, J_2), we set a threshold $\epsilon = 10^{-10}$ for these iterative algorithms (i.e if $\log_{10}(\|\mathbf{A} - \mathbf{X}\mathbf{Y}^\top\|_F) \leq -10$, the algorithm is terminated and considered to have found an optimal solution). This determines the running time for a given iterative algorithm for a given dimension 2^N and a given learning rate. For each dimension 2^N we report the best running time over all learning rates. The reported running times do not include the time required for hyperparameters tuning. The experiments illustrated in Figure 5.4 for $N = 10$ confirm our results

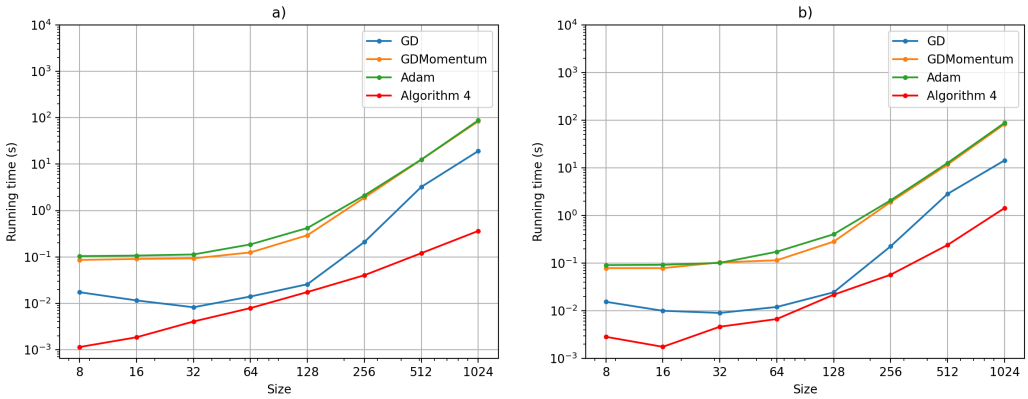


Figure 5.5: Running time (in logarithmic scale, contrary to Figure 5.4) of three variants of gradient descent and Algorithm 4 to reach a precision $\log_{10}(\|\mathbf{A} - \mathbf{X}\mathbf{Y}^\top\|_F) \leq -10$; $N \in \{3, \dots, 10\}$ with support constraints (I_1, J_1) (left) and (I_2, J_2) (right).

on the landscape presented in Section 5.3: the assumptions of theorem theorem 5.2.3 are satisfied so the landscape is benign and all variants of gradient descent are able to find a good factorization for A from a random initialization.

Figure 5.4 also shows that Algorithm 4 is consistently better than the considered iterative methods in terms of running time, regardless of the size of $\mathbf{A}$, cf. Figure 5.5. A crucial advantage of Algorithm 4 over gradient methods is also that it is free of hyperparameter tuning, which is critical for iterative methods to perform well, and may be quite time consuming (we recall that the time required for hyperparameters tuning of these iterative methods is *not* considered in Figure 5.5). In addition, Algorithm 4 can be further accelerated since its main steps (cf Algorithm 2) rely on block SVDs that can be computed in parallel (in these experiments, our implementation of Algorithm 4 is not parallelized yet).

5.5 Proof for technical results in Section 5.2

5.5.1 Proof for Lemma 5.2.4

Proof. The proof is carried out by induction.

1) For $N = 1$, one can consider $(I, J) \in \{0,1\}^{2\times 4} \times \{0,1\}^{2\times 4}$ defined (in the binary matrix form) as follows:

$$I = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}, \; J = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}.$$

Any $(\mathbf{X}, \mathbf{Y})$ constrained to (I, J) will have the following form:

$$\mathbf{X} = \begin{pmatrix} x_1 & 0 & x_3 & 0 \\ 0 & x_2 & 0 & x_4 \end{pmatrix}, \; \mathbf{Y} = \begin{pmatrix} 0 & y_2 & y_3 & 0 \\ y_1 & 0 & 0 & y_4 \end{pmatrix}, \; \mathbf{X}\mathbf{Y}^\top = \begin{pmatrix} x_3y_3 & x_1y_1 \\ x_2y_2 & x_4y_4 \end{pmatrix}.$$

Given any matrix $\mathbf{H} \in \mathbb{R}^{2\times 2}$ (and in particular, given any HODLR matrix in this dimension) it is easy to see that $\mathbf{H}$ can be represented as $\mathbf{X}\mathbf{Y}^\top$ such that $\text{supp}(\mathbf{X}) \subseteq I, \text{supp}(\mathbf{Y}) \subseteq J$ (take e.g. $x_3 = a_{11}$, $x_1 = a_{12}$, $x_2 = a_{21}$, $x_4 = a_{22}$ and all $y_i = 1$). It is also easy to verify that this choice of (I, J) makes all the supports of the rank-one contributions pairwise disjoint, so that the assumptions of Theorem 5.2.3 are fulfilled. Finally, we observe that $|I_N| = |J_N| = 4$.

2) Suppose that our hypothesis is correct for $N-1$, we need to prove its correctness for N. Let (I_{N-1}, J_{N-1}) be the pair of supports for $N-1$, we construct (I_N, J_N) (still in binary matrix form) as follows:

$$I_N = \begin{pmatrix} \mathbf{1}_{n/2\times 1} & \mathbf{0}_{n/2\times 1} & I_{N-1} & \mathbf{0}_{n/2\times(3n/2-2)} \\ \mathbf{0}_{n/2\times 1} & \mathbf{1}_{n/2\times 1} & \mathbf{0}_{n/2\times(3n/2-2)} & I_{N-1} \end{pmatrix}$$
$$J_N = \begin{pmatrix} \mathbf{0}_{n/2\times 1} & \mathbf{1}_{n/2\times 1} & J_{N-1} & \mathbf{0}_{n/2\times(3n/2-2)} \\ \mathbf{1}_{n/2\times 1} & \mathbf{0}_{n/2\times 1} & \mathbf{0}_{n/2\times(3n/2-2)} & J_{N-1} \end{pmatrix}$$

where $n = 2^N$ and $\mathbf{1}_{p\times q}$ (resp. $\mathbf{0}_{p\times q}$) is the matrix of size $p\times q$ full of ones (resp. of zeros). Since I_{N-1} and J_{N-1} are both of dimension $2^{N-1} \times (3 \times 2^{N-1} - 2) = (n/2)(3n/2 - 2)$, the dimensions of I_N and J_N are both equal to $(n, 2 \times (3n/2 - 2) + 2) = (n, 3n - 2)$. Moreover, the cardinalities of I_N and J_N satisfy the following recursive formula:

$$|I_N| = n + 2|I_{N-1}|, \qquad |J_N| = n + 2|J_{N-1}|,$$

which justifies the fact that $|I_N| = |J_N| = O(n \log n)$. Finally, any pair of factors $(\mathbf{X}, \mathbf{Y})$ respecting the support constraints (I_N, J_N) need to have the following form:

$$\mathbf{X} = \begin{pmatrix} \mathbf{X}_1 & \mathbf{0}_{n/2\times 1} & \mathbf{X}_3 & \mathbf{0}_{n/2\times(3n/2-2)} \\ \mathbf{0}_{n/2\times 1} & \mathbf{X}_2 & \mathbf{0}_{n/2\times(3n/2-2)} & \mathbf{X}_4 \end{pmatrix}$$
$$\mathbf{Y} = \begin{pmatrix} \mathbf{0}_{n/2\times 1} & \mathbf{Y}_2 & \mathbf{Y}_3 & \mathbf{0}_{n/2\times(3n/2-2)} \\ \mathbf{Y}_1 & \mathbf{0}_{n/2\times 1} & \mathbf{0}_{n/2\times(3n/2-2)} & \mathbf{Y}_4 \end{pmatrix}$$

where $\mathbf{X}_i, \mathbf{Y}_i \in \mathbb{R}^{n/2}, 1 \le i \le 2$, and for $3 \le j \le 4$ we have $\mathbf{X}_j, \mathbf{Y}_j \in \mathbb{R}^{n/2\times(3n/2-2)}$, $\text{supp}(\mathbf{X}_j) \subseteq I_{N-1}, \text{supp}(\mathbf{Y}_j) \subseteq J_{N-1}$. Their product yields:

$$\mathbf{X}\mathbf{Y}^\top = \begin{pmatrix} \mathbf{X}_3\mathbf{Y}_3^\top & \mathbf{X}_1\mathbf{Y}_1^\top \\ \mathbf{X}_2\mathbf{Y}_2^\top & \mathbf{X}_4\mathbf{Y}_4^\top \end{pmatrix}.$$

Given an HODLR matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, since $\mathbf{A}_{12}, \mathbf{A}_{21} \in \mathbb{R}^{n/2\times n/2}$ are of rank at most one, one can find $\mathbf{X}_i, \mathbf{Y}_i \in \mathbb{R}^{n/2}, 1 \leq i \leq 2$ such that $\mathbf{A}_{12} = \mathbf{X}_1\mathbf{Y}_1^\top, \mathbf{A}_{21} = \mathbf{X}_2\mathbf{Y}_2^\top$. Since $\mathbf{A}_{11}, \mathbf{A}_{22} \in \mathbb{R}^{n/2\times n/2}$ are HODLR, by the induction hypothesis, one can also find $\mathbf{X}_i, \mathbf{Y}_i \in \mathbb{R}^{n/2\times(3n/2-2)}$, $3 \leq i \leq 4$ such that $\mathrm{supp}(\mathbf{X}_i) \subseteq I_{N-1}$, $\mathrm{supp}(\mathbf{Y}_i) \subseteq I_{N-1}$ and $\mathbf{A}_{11} = \mathbf{X}_3\mathbf{Y}_3^\top, \mathbf{A}_{22} = \mathbf{X}_4\mathbf{Y}_4^\top$. Finally, this construction also makes all the supports of the rank-one contributions pairwise disjoint: the first two rank-one supports are $\mathcal{S}_1 = \{n/2+1,\dots,n\} \times [\![n/2]\!], \mathcal{S}_2 = [\![n/2]\!] \times \{n/2+1,\dots,n\}$, and the remaining ones are inside $[\![n/2]\!] \times [\![n/2]\!]$ and $\{n/2+1,\dots,n\} \times \{n/2+1,\dots,n\}$ which are disjoint by the induction hypothesis. □

5.5.2 Proof for Lemma 5.2.6

Proof. Denote $\mathcal{P}$ the partition of $[\![r]\!]$ into equivalence classes defined by the rank-one supports associated to (I, J), and $\mathcal{P}^\star \subseteq \mathcal{P}$ the corresponding CECs. Remind that we will use the shorthand $\mathbf{W}_P := \mathbf{W}[:, P] \in \mathbb{R}^{m\times n}$ for any matrix $\mathbf{W} \in \mathbb{R}^{m\times n}$ (see definition of Notation) and for any $P \subseteq [\![n]\!]$.

Since $T \subseteq [\![r]\!]$ is precisely the set of indices of CECs, and since $I_T := I[:, T]$ (resp. $J_T := J[:, T]$) is the restriction of I (resp. of J) to columns indexed by T, the partition of $[\![r]\!]$ into equivalence classes *w.r.t* (I_T, J_T) is precisely $\mathcal{P}^\star$, and for $P \in \mathcal{P}\backslash\mathcal{P}^*$, we have $\mathcal{S}_P = \emptyset$. WLOG, we assume $\mathcal{P}^\star = \{P_i \mid 1 \leq i \leq \ell\}$. Denote $\mathcal{P}_k = \{P_1, \dots, P_k\}$, $\mathcal{S}_{\mathcal{P}_k} = \cup_{1\leq i\leq k}\mathcal{S}_{P_i}$ for $1 \leq k \leq \ell$ and $\mathcal{S}_{P_0} = \emptyset$. We prove below that $(\mathbf{X}, \mathbf{Y}) = \text{SVD\_FSMF}(\mathbf{A}, I_T, J_T)$ satisfies:

$$\mathbf{X}_{P_k}\mathbf{Y}_{P_k}^\top = \mathbf{A} \odot (\mathcal{S}_{\mathcal{P}_k} \setminus \mathcal{S}_{\mathcal{P}_{k-1}}), \forall\ 1 \leq k \leq \ell, \tag{5.8}$$

which implies: $\mathbf{X}\mathbf{Y}^\top = \sum_{P\in\mathcal{P}^\star}\mathbf{X}_P\mathbf{Y}_P^\top = \sum_{k=1}^{\ell}\mathbf{A}\odot(\mathcal{S}_{\mathcal{P}_k}\setminus\mathcal{S}_{\mathcal{P}_{k-1}}) = \mathbf{A}\odot\mathcal{S}_\ell = \mathbf{A}\odot\mathcal{S}_T = \mathbf{A}$ (since we assume $\mathrm{supp}(\mathbf{A}) = \mathcal{S}_T$). This yields the conclusion since $\mathrm{supp}(\mathbf{X}) \subseteq I_T$ and $\mathrm{supp}(\mathbf{Y}) \subseteq J_T$ by definition of SVD_FSMF$(\cdot)$.

We prove Equation (5.8) by induction on ℓ. To ease the reading, in this proof, we denote C_{P_k}, R_{P_k} (Definition 5.2.5) by C_k, R_k respectively.

For $\ell = 1$ it is sufficient to consider $k = 1$: we have $\mathcal{S}_{\mathcal{P}_1} \setminus \mathcal{S}_{\mathcal{P}_0} = C_1 \times R_1$. Since $\min(|R_1|, |C_1|) \leq |P_1|$ (Definition 5.2.5), taking the best rank-$|P_1|$ approximation of $A \odot (R_1 \times C_1)$ (whose rank is at most $\min(|R_1|, |C_1|)$) yields $\mathbf{X}_{P_1}\mathbf{Y}_{P_1}^\top = \mathbf{A} \odot (R_1 \times C_1) = \mathbf{A} \odot (\mathcal{S}_{\mathcal{P}_1} \setminus \mathcal{S}_{\mathcal{P}_0})$.

Assume that Equation (5.8) holds for $\ell - 1$. We prove its correctness for ℓ. Consider: $\mathbf{A}' := \mathbf{A} - \sum_{k<\ell}\mathbf{X}_{P_k}\mathbf{Y}_{P_k}^\top = \mathbf{A} - \mathbf{A} \odot \mathcal{S}_{\mathcal{P}_{\ell-1}} = \mathbf{A} \odot \bar{\mathcal{S}}_{\mathcal{P}_{\ell-1}}$. Therefore, $\mathbf{A}' \odot \mathcal{S}_{P_\ell} = \mathbf{A} \odot (\mathcal{S}_{\mathcal{P}_\ell} \setminus \mathcal{S}_{\mathcal{P}_{\ell-1}})$. Again, since $\min(|R_\ell|, |C_\ell|) \leq |P_\ell|$ (Definition 5.2.5), taking the best rank-$|P_\ell|$ approximation of $\mathbf{A}'\odot\mathcal{S}_{P_\ell} = \mathbf{A}'\odot(R_\ell\times C_\ell)$ (whose rank is at most $\min(|R_\ell|, |C_\ell|)$) yields $\mathbf{X}_{P_\ell}\mathbf{Y}_{P_\ell}^\top = \mathbf{A}'\odot(R_\ell\times C_\ell) = \mathbf{A}\odot(\mathcal{S}_{P_\ell}\setminus\mathcal{S}_{P_{\ell-1}})$. That implies Equation (5.8) is correct for all ℓ. □

5.5.3 Proof for Theorem 5.2.9

Before presenting the proof of Theorem 5.2.9, we introduce a definition and a technical lemma. The following definition introduces a decomposition of the factors $\mathbf{X}$ and $\mathbf{Y}$ using the taxonomy of indices from Definition 5.2.8.

Definition 5.5.1. Given I_T, J_T and $I_T^i, J_T^i, i = 1, 2$ as in Definition 5.2.8, consider (X, Y) a feasible point of (2.4.FSMF), we denote:

1) $\mathbf{X}_T = \mathbf{X} \odot I_T, \mathbf{X}^i_{\bar{T}} = \mathbf{X} \odot I^i_{\bar{T}}$, for $i = 1, 2$.

2) $\mathbf{Y}_T = \mathbf{Y} \odot I_T, \mathbf{Y}^i_{\bar{T}} = \mathbf{Y} \odot I^i_{\bar{T}}$, for $i = 1, 2$.

with $\odot$ the Hadamard product between a matrix and a binary matrix representing support constraint.

The following is a technical result.

Lemma 5.5.2. *Given I, J support constraints of (2.4.FSMF), consider $T, \mathcal{S}_T, \mathcal{S}_\mathcal{P}$ as in Definition 5.2.2, $\mathbf{X}_T, \mathbf{X}^i_{\bar{T}}, \mathbf{Y}_T, \mathbf{Y}^i_{\bar{T}}$ as in Definition 5.2.7 and assume that for all $k \in \bar{T}$, $\mathcal{S}'_k$ is rectangular. It holds:*

C1 $\operatorname{supp}(\mathbf{X}_T \mathbf{Y}_T^\top) \subseteq \mathcal{S}_T$.

C2 $\operatorname{supp}(\mathbf{X}^1_{\bar{T}} (\mathbf{Y}^1_{\bar{T}})^\top) \subseteq \mathcal{S}_\mathcal{P} \setminus \mathcal{S}_T$.

C3 $\operatorname{supp}(\mathbf{X}^i_{\bar{T}} (\mathbf{Y}^j_{\bar{T}})^\top) \subseteq \mathcal{S}_T, \forall 1 \leq i, j \leq 2, (i, j) \neq (1, 1)$.

Proof. We justify (**C1**)-(**C3**) as follow:

- **C1**: Since $\mathbf{X}_T \mathbf{Y}_T^\top = \sum_{i \in T} \mathbf{X}[:, i] \mathbf{Y}[:, i]^\top$, $\operatorname{supp}(\mathbf{X}_T \mathbf{Y}_T^\top) \subseteq \cup_{i \in T} \mathcal{S}_k = \mathcal{S}_T$.
- **C2**: Consider the coefficient (i, j) of $(\mathbf{X}^1_{\bar{T}})(\mathbf{Y}^1_{\bar{T}})^\top$

$$((\mathbf{X}^1_{\bar{T}})(\mathbf{Y}^1_{\bar{T}})^\top)_{i,j} = \sum_k (\mathbf{X}^1_{\bar{T}})_{i,k} (\mathbf{Y}^1_{\bar{T}})_{j,k} = \sum_{(i,k) \in I^1_{\bar{T}}, (j,k) \in J^1_{\bar{T}}} \mathbf{X}_{i,k} \mathbf{Y}_{j,k}$$

  By the definition of $I^1_{\bar{T}}, J^1_{\bar{T}}$, $(\mathbf{X}^1_{\bar{T}})(\mathbf{Y}^1_{\bar{T}})^\top_{i,j} \neq 0$ iff $(i, j) \in \cup_{\ell \in \bar{T}} R_\ell \times C_\ell = \mathcal{S}_\mathcal{P} \setminus \mathcal{S}_T$.

- **C3**: We prove for the case of $(\mathbf{X}^1_{\bar{T}})(\mathbf{Y}^2_{\bar{T}})^\top$. Others can be proved similarly.

$$((\mathbf{X}^1_{\bar{T}})(\mathbf{Y}^2_{\bar{T}})^\top)_{i,j} = \sum_k (\mathbf{X}^1_{\bar{T}})_{i,k} (\mathbf{Y}^2_{\bar{T}})_{j,k} = \sum_{(i,k) \in I^1_{\bar{T}}, (j,k) \in J^2_{\bar{T}}} \mathbf{X}_{i,k} \mathbf{Y}_{j,k} \tag{5.9}$$

  Since $\forall \ell \in \bar{T}, \mathcal{S}'_\ell$ is rectangular, $\mathcal{S}_\mathcal{P} \setminus \mathcal{S}_T = \cup_{\ell \in \bar{T}} \mathcal{S}'_\ell = \cup_{\ell \in \bar{T}} R_\ell \times C_\ell$. If $(i, j) \in \mathcal{S}_\mathcal{P} \setminus \mathcal{S}_T$, Equation (5.9) shows that $((\mathbf{X}^1_{\bar{T}})(\mathbf{Y}^2_{\bar{T}})^\top)_{i,j} = 0$ since there is no k such that $(i, k) \in I^2_{\bar{T}}, (j, k) \in J^2_{\bar{T}}$ due to the definition of $I^1_{\bar{T}}, J^2_{\bar{T}}$). Moreover, $\operatorname{supp}((\mathbf{X}^1_{\bar{T}})(\mathbf{Y}^2_{\bar{T}})^\top) \subseteq \mathcal{S}_\mathcal{P}$ (since $\operatorname{supp}(\mathbf{X}^1_{\bar{T}}) \subseteq I, \operatorname{supp}(\mathbf{Y}^2_{\bar{T}}) \subseteq J$). Thus, it shows that $\operatorname{supp}((\mathbf{X}^1_{\bar{T}})(\mathbf{Y}^2_{\bar{T}})^\top) \subseteq \mathcal{S}_\mathcal{P} \setminus (\mathcal{S}_\mathcal{P} \setminus \mathcal{S}_T) = \mathcal{S}_T$. □

Here, we present the proof of Theorem 5.2.9.

Proof of Theorem 5.2.9. Given $\mathbf{X}, \mathbf{Y}$ feasible point of the input $(\mathbf{A}, I, J)$, consider $\mathbf{X}_T, \mathbf{Y}_T$ and $\mathbf{X}^i_{\bar{T}}, \mathbf{Y}^i_{\bar{T}}, i = 1, 2$ defined as in Definition 5.5.1. Let μ_1 and μ_2 be the infimum value of (2.4.FSMF) with $(\mathbf{A}, I, J)$ and with $(\mathbf{A}', I^1_{\bar{T}}, J^1_{\bar{T}})$ ($\mathbf{A}' = \mathbf{A} \odot \bar{\mathcal{S}}_T$) respectively.

First, we remark that $I^1_{\bar{T}}$ and $J^1_{\bar{T}}$ satisfy the assumptions of Theorem 5.2.3. Indeed, it holds $\mathcal{S}_k(I^1_{\bar{T}}, J^1_{\bar{T}}) = \mathcal{S}_k(I, J) \setminus \mathcal{S}_T = \mathcal{S}'_k$ by construction. For any two indices $k, l \in \bar{T}$, the representative rank-one supports are either equal ($\mathcal{S}'_k = \mathcal{S}'_l$) or disjoint ($\mathcal{S}'_k \cap \mathcal{S}'_l = \emptyset$) by assumption. That shows why $I^1_{\bar{T}}$ and $J^1_{\bar{T}}$ satisfy the assumptions of Theorem 5.2.3.

Next, we prove that $\mu_1 = \mu_2$. Since $(\mathcal{S}_T, \mathcal{S}_\mathcal{P} \setminus \mathcal{S}_T, \bar{\mathcal{S}}_\mathcal{P})$ form a partition of $[\![m]\!] \times [\![n]\!]$, we have $\mathbf{C} \odot \mathbf{D} = \mathbf{0}$, $\mathbf{C} \neq \mathbf{D}, \mathbf{C}, \mathbf{D} \in \{\mathcal{S}_T, \mathcal{S}_\mathcal{P} \setminus \mathcal{S}_T, \bar{\mathcal{S}}_\mathcal{P}\}$ ($\mathbf{C}, \mathbf{D}$ are represented in binary matrix forms). From the definition of $\mathbf{A}'$ it holds $\mathbf{A}' \odot \bar{\mathcal{S}}_\mathcal{P} = \mathbf{A} \odot \bar{\mathcal{S}}_\mathcal{P}$ and $\mathbf{A}' \odot \mathcal{S}_T = \mathbf{0}$. Moreover, it holds $(\mathbf{X}^1_{\bar{T}})(\mathbf{Y}^1_{\bar{T}})^\top \odot \mathcal{S}_T \cup \bar{\mathcal{S}}_\mathcal{P} = \mathbf{0}$ due to **C2**.

Since $\operatorname{supp}(\mathbf{X}_T) \subseteq I_T, \operatorname{supp}(\mathbf{X}^i_{\bar{T}}) \subseteq I_{\bar{T}}, \operatorname{supp}(\mathbf{Y}_T) \subseteq J_T, \operatorname{supp}(\mathbf{Y}^i_{\bar{T}}) \subseteq J_{\bar{T}}, i = 1, 2$, the product $\mathbf{X}\mathbf{Y}^\top$ can be decomposed as:

$$\mathbf{X}\mathbf{Y}^\top = \mathbf{X}_T\mathbf{Y}_T^\top + \sum_{1 \leq i,j \leq 2} (\mathbf{X}^i_{\bar{T}})(\mathbf{Y}^j_{\bar{T}})^\top. \tag{5.10}$$

Consider the loss function of (2.4.FSMF) with input $(\mathbf{A}', I^1_{\bar{T}}, J^1_{\bar{T}})$ and solution $(\mathbf{X}^1_{\bar{T}}, \mathbf{Y}^1_{\bar{T}})$:

$$\begin{aligned}
&\|\mathbf{A}' - \mathbf{X}^1_{\bar{T}}(\mathbf{Y}^1_{\bar{T}})^\top\|^2 \\
&= \|(\mathbf{A}' - \mathbf{X}^1_{\bar{T}}(\mathbf{Y}^1_{\bar{T}})^\top) \odot \mathcal{S}_T\|^2 + \|(\mathbf{A}' - \mathbf{X}^1_{\bar{T}}(\mathbf{Y}^1_{\bar{T}})^\top) \odot (\mathcal{S}_\mathcal{P} \setminus \mathcal{S}_T)\|^2 \\
&\qquad + \|(\mathbf{A}' - \mathbf{X}^1_{\bar{T}}(\mathbf{Y}^1_{\bar{T}})^\top) \odot \bar{\mathcal{S}}_\mathcal{P}\|^2 \\
&\overset{\text{C2}}{=} \|(\mathbf{A}' - (\mathbf{X}^1_{\bar{T}})(\mathbf{Y}^1_{\bar{T}})^\top) \odot \mathcal{S}_\mathcal{P} \setminus \mathcal{S}_T\|^2 + \|\mathbf{A}' \odot \bar{\mathcal{S}}_\mathcal{P}\|^2 \\
&\overset{\text{C1+C3}}{=} \|(\mathbf{A} - \mathbf{X}_T\mathbf{Y}_T^\top - \sum_{1 \leq i,j \leq 2} (\mathbf{X}^i_{\bar{T}})(\mathbf{Y}^j_{\bar{T}})^\top) \odot (\mathcal{S}_\mathcal{P} \setminus \mathcal{S}_T)\|^2 + \|\mathbf{A} \odot \bar{\mathcal{S}}_\mathcal{P}\|^2 \\
&\overset{(5.10)}{=} \|(\mathbf{A} - \mathbf{X}\mathbf{Y}^\top) \odot (\mathcal{S}_\mathcal{P} \setminus \mathcal{S}_T)\|^2 + \|\mathbf{A} \odot \bar{\mathcal{S}}_\mathcal{P}\|^2
\end{aligned} \tag{5.11}$$

Perform the same calculation with $(\mathbf{A}, I, J)$ and solution $(\mathbf{X}, \mathbf{Y})$:

$$\begin{aligned}
&\|(\mathbf{A} - \mathbf{X}\mathbf{Y}^\top)\|^2 \\
&= \|(\mathbf{A} - \mathbf{X}\mathbf{Y}^\top) \odot \mathcal{S}_T\|^2 + \|(\mathbf{A} - \mathbf{X}\mathbf{Y}^\top) \odot (\mathcal{S}_\mathcal{P} \setminus \mathcal{S}_T)\|^2 + \|(\mathbf{A} - \mathbf{X}\mathbf{Y}^\top) \odot \bar{\mathcal{S}}_\mathcal{P}\|^2 \\
&= \|(\mathbf{A} - \mathbf{X}\mathbf{Y}^\top) \odot \mathcal{S}_T\|^2 + \|(\mathbf{A} - \mathbf{X}\mathbf{Y}^\top) \odot (\mathcal{S}_\mathcal{P} \setminus \mathcal{S}_T)\|^2 + \|\mathbf{A} \odot \bar{\mathcal{S}}_\mathcal{P}\|^2
\end{aligned} \tag{5.12}$$

where the last equality holds since $\operatorname{supp}(\mathbf{X}\mathbf{Y}^\top) \subseteq \mathcal{S}_\mathcal{P}$. Therefore, for any feasible point $(\mathbf{X}, \mathbf{Y})$ of instance $(\mathbf{A}, I, J)$, we can choose $\tilde{\mathbf{X}} = \mathbf{X}^1_{\bar{T}}, \tilde{\mathbf{Y}} = \mathbf{Y}^1_{\bar{T}}$ feasible point of $(\mathbf{A}', I^1_{\bar{T}}, J^1_{\bar{T}})$ such that $\|\mathbf{A} - \mathbf{X}\mathbf{Y}^\top\| \geq \|\mathbf{A}' - \tilde{\mathbf{X}}\tilde{\mathbf{Y}}^\top\|$ (Equation (5.11) and Equation (5.12)). This shows $\mu_1 \geq \mu_2$.

On the other hand, given any feasible point $(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$ of instance $(\mathbf{A}', I^1_{\bar{T}}, J^1_{\bar{T}})$, we can construct a feasible point $(\mathbf{X}, \mathbf{Y})$ for instance $(\mathbf{A}, I, J)$ such that $\|\mathbf{A} - \mathbf{X}\mathbf{Y}^\top\|^2 = \|\mathbf{A}' - \mathbf{X}'\mathbf{Y}'^\top\|^2$. We construct $(\mathbf{X}, \mathbf{Y}) = (\mathbf{X}_T + \mathbf{X}^1_{\bar{T}} + \mathbf{X}^2_{\bar{T}}, \mathbf{Y}_T + \mathbf{Y}^1_{\bar{T}} + \mathbf{Y}^2_{\bar{T}})$ where:

1) $\mathbf{X}^1_{\bar{T}} = \tilde{\mathbf{X}}, \mathbf{Y}^1_{\bar{T}} = \tilde{\mathbf{Y}}$,
2) $\mathbf{X}^2_{\bar{T}}, \mathbf{Y}^2_{\bar{T}}$ can be chosen arbitrarily such that $\operatorname{supp}(\mathbf{X}^2_{\bar{T}}) \subseteq I^2_{\bar{T}}, \operatorname{supp}(\mathbf{Y}^2_{\bar{T}}) \subseteq J^2_{\bar{T}}$
3) $\mathbf{X}_T$ and $\mathbf{Y}_T$ such that $\operatorname{supp}(\mathbf{X}_T) \subseteq I_T, \operatorname{supp}(\mathbf{Y}_T) \subseteq J_T$ and:

$$\mathbf{X}_T\mathbf{Y}_T^\top = (\mathbf{A} - (\mathbf{X}^1_{\bar{T}} + \mathbf{X}^2_{\bar{T}})(\mathbf{Y}^1_{\bar{T}} + \mathbf{Y}^2_{\bar{T}})^\top) \odot \mathcal{S}_T$$

$(\mathbf{X}_T, \mathbf{Y}_T)$ exists due to Lemma 5.2.6. By Lemma 5.5.2, with this choice we have:

$$\begin{aligned}
(\mathbf{A} - \mathbf{X}\mathbf{Y}^\top) \odot \mathcal{S}_T &\overset{(5.10)}{=} (\mathbf{A} - (\mathbf{X}^1_{\bar{T}} + \mathbf{X}^2_{\bar{T}})(\mathbf{Y}^1_{\bar{T}} + \mathbf{Y}^2_{\bar{T}})^\top - \mathbf{X}_T\mathbf{Y}_T^\top) \odot \mathcal{S}_T \\
&\overset{\text{C1}}{=} (\mathbf{A} - (\mathbf{X}^1_{\bar{T}} + \mathbf{X}^2_{\bar{T}})(\mathbf{Y}^1_{\bar{T}} + \mathbf{Y}^2_{\bar{T}})^\top) \odot \mathcal{S}_T) - \mathbf{X}_T\mathbf{Y}_T^\top = \mathbf{0}
\end{aligned} \tag{5.13}$$

Therefore $\|\mathbf{A} - \mathbf{X}\mathbf{Y}^\top\|^2 = \|\mathbf{A}' - \tilde{\mathbf{X}}\tilde{\mathbf{Y}}^\top\|^2$ (Equation (5.11) and Equation (5.12)). Thus, $\mu_2 \geq \mu_1$. We obtain $\mu_1 = \mu_2$. In addition, given $(\mathbf{X}, \mathbf{Y})$ an optimal solution of (2.4.FSMF) with instance $(\mathbf{A}, I, J)$, we have shown how to construct an optimal solution $(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$ with instance $(\mathbf{A} \odot \bar{\mathcal{S}}_T, I^1_{\bar{T}}, J^1_{\bar{T}})$ and vice versa. That completes our proof. □

The following Corollary is a direct consequence of the proof of Theorem 5.2.9.

Corollary 5.5.3. *With the same assumptions and notations as in Theorem 5.2.9, a feasible point* $(\mathbf{X}, \mathbf{Y})$ *(i.e., such that* $\text{supp}(\mathbf{X}) \subseteq I, \text{supp}(\mathbf{Y}) \subseteq J$*) is an optimal solution of* (2.4.FSMF) *if and only if:*

1) $(\mathbf{X} \odot I^1_{\bar{T}}, \mathbf{Y} \odot J^1_{\bar{T}})$ *is an optimal solution of* (2.4.FSMF) *with* $(\mathbf{A} \odot \bar{\mathcal{S}}_T, I^1_{\bar{T}}, J^1_{\bar{T}})$.

2) The following equation holds: $(\mathbf{A} - \mathbf{X}\mathbf{Y}^\top) \odot \mathcal{S}_T = \mathbf{0}$

Remark 5.5.4. In the proof of Theorem 5.2.9, for an optimal solution, one can choose $\mathbf{X}^2_{\bar{T}}, \mathbf{Y}^2_{\bar{T}}$ arbitrarily. If we choose $\mathbf{X}^2_{\bar{T}} = \mathbf{0}, \mathbf{Y}^2_{\bar{T}} = \mathbf{0}$, thanks to Equation (5.13), $\mathbf{X}_T$ and $\mathbf{Y}_T$ has to satisfy:

$$\mathbf{X}_T\mathbf{Y}_T^\top = (\mathbf{A} - (\mathbf{X}^1_{\bar{T}} + \mathbf{X}^2_{\bar{T}})(\mathbf{Y}^1_{\bar{T}} + \mathbf{Y}^2_{\bar{T}})^\top) \odot \mathcal{S}_T = (\mathbf{A} - \mathbf{X}^1_{\bar{T}}(\mathbf{Y}^1_{\bar{T}})^\top) \odot \mathcal{S}_T \overset{\text{C2}}{=} \mathbf{A} \odot \mathcal{S}_T$$

5.6 Proof for technical results in Section 5.3

In this section, provide proofs for several technical lemmas in Section 5.3, namely Lemmas 5.3.4 to 5.3.6 and Corollary 5.6.6.

5.6.1 Proofs for a key lemma

In this section, we will introduce an important technical lemma. It is used extensively for the proof of other technical lemmas of the tractability and the landscape of (2.4.FSMF) under the assumptions of Theorem 5.2.9.

Lemma 5.6.1. *Consider* I, J *support constraints of* (2.4.FSMF) *such that* $\mathcal{P}^\star = \mathcal{P}$*. For any CEC-full-rank feasible point* $(\mathbf{X}, \mathbf{Y})$ *and continuous function* $g : [0,1] \to \mathbb{R}^{m\times n}$ *satisfying* $\text{supp}(g(t)) \subseteq \mathcal{S}_T$ *(Definition 5.2.5) and* $g(0) = \mathbf{X}\mathbf{Y}^\top$*, there exists a feasible continuous function* $f : [0,1] \to \mathbb{R}^{m\times r} \times \mathbb{R}^{n\times r} : f(t) = (\mathbf{X}_f(t), \mathbf{Y}_f(t))$ *such that:*

A1 $f(0) = (\mathbf{X}_T, \mathbf{Y}_T)$.

A2 $g(t) = \mathbf{X}_f(t)\mathbf{Y}_f(t)^\top, \forall t \in [0,1]$.

A3 $\|f(z) - f(t)\|^2 \leq \mathcal{C}\|g(z) - g(t)\|^2, \forall t, z \in [0,1]$.

where $\mathcal{C} = \max_{P\in\mathcal{P}^\star}\left(\max\left(\left|\left|\left|\mathbf{X}^\dagger_{R_P,P}\right|\right|\right|^2, \left|\left|\left|\mathbf{Y}^\dagger_{C_P,P}\right|\right|\right|^2\right)\right)$ *(*$\mathbf{D}^\dagger$ *and* $|||\mathbf{D}|||$ *denote the pseudo-inverse and operator norm of a matrix* $\mathbf{D}$ *respectively).*

Lemma 5.6.1 consider the case where $\mathcal{P}$ only contains CECs. Later in other proofs, we will control the factors $(\mathbf{X}, \mathbf{Y})$ by decomposing $\mathbf{X} = \mathbf{X}_T + \mathbf{X}_{\bar{T}}$ (and $\mathbf{Y} = \mathbf{Y}_T + \mathbf{Y}_{\bar{T}}$) ($T, \bar{T}$ defined in Definition 5.2.5 and $\mathbf{X}_{\bar{T}} := \mathbf{X}[:, \bar{T}], \mathbf{Y}_{\bar{T}} := \mathbf{Y}[:, \bar{T}]$) and manipulate $(\mathbf{X}_T, \mathbf{Y}_T)$ and $(\mathbf{X}_{\bar{T}}, \mathbf{Y}_{\bar{T}})$ separately. Since the supports of $(\mathbf{X}_T, \mathbf{Y}_T)$ satisfy Lemma 5.6.1, it provides us a tool to work with $(\mathbf{X}_T, \mathbf{Y}_T)$.

The proof of Lemma 5.6.1 is carried out by induction. We firstly introduce and prove two other lemmas: Lemma 5.6.2 and Lemma 5.6.3. While Lemma 5.6.2 is Lemma 5.6.1 without support constraints, Lemma 5.6.3 is Lemma 5.6.1 where $|\mathcal{P}^\star| = 1$.

Lemma 5.6.2. *Let $\mathbf{X} \in \mathbb{R}^{m\times r}, \mathbf{Y} \in \mathbb{R}^{n\times r}, \min(m, n) \leq r$ and assume that $\mathbf{X}$ or $\mathbf{Y}$ has full row rank. Given any continuous function $g : [0, 1] \to \mathbb{R}^{m\times n}$ in which $g(0) = \mathbf{X}\mathbf{Y}^\top$, there exists a continuous function $f : [0, 1] \to \mathbb{R}^{m\times r} \times \mathbb{R}^{n\times r} : f(t) = (\mathbf{X}_f(t), \mathbf{Y}_f(t))$ such that:*

1) $f(0) = (\mathbf{X}, \mathbf{Y})$.

2) $g(t) = \mathbf{X}_f(t)\mathbf{Y}_f(t)^\top, \forall t \in [0, 1]$.

3) $\|f(z) - f(t)\|^2 \leq \mathcal{C}\|g(z) - g(t)\|^2, \forall t, z \in [0, 1]$.

where $\mathcal{C} = \max\left(\left|\left|\left|\mathbf{X}^\dagger\right|\right|\right|^2, \left|\left|\left|\mathbf{Y}^\dagger\right|\right|\right|^2\right)$.

Proof. WLOG, we can assume that X has full row rank. We define f as:

$$\begin{aligned} \mathbf{X}_f(t) &= \mathbf{X} \\ \mathbf{Y}_f(t) &= \mathbf{Y} + (g(t) - g(0))^\top(\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X} = \mathbf{Y} + (\mathbf{X}^\dagger(g(t) - g(0)))^\top \end{aligned} \tag{5.14}$$

where $\mathbf{X}^\dagger = \mathbf{X}^\top(\mathbf{X}\mathbf{X}^\top)^{-1}$ the pseudo-inverse of $\mathbf{X}$. The function $\mathbf{Y}_f$ is well-defined due to the assumption of $\mathbf{X}$ being full row rank. It is immediate for the first two constraints. Since $\|f(z) - f(t)\|^2 = \|\mathbf{Y}_f(z) - \mathbf{Y}_f(t)\|^2 = \|\mathbf{X}^\dagger(g(z) - g(t))\|^2$, the third one is also satisfied as:

$$\|f(z) - f(t)\|^2 = \|\mathbf{X}^\dagger(g(z) - g(t))\|^2 \leq \left|\left|\left|\mathbf{X}^\dagger\right|\right|\right|^2 \|g(z) - g(t)\|^2 \leq \mathcal{C}\|g(z) - g(t)\|^2$$

□

Lemma 5.6.3. *Consider I, J support constraints of (2.4.FSMF) where $\mathcal{P}^\star = \mathcal{P} = \{P\}$, for any feasible CEC-full-rank point $(\mathbf{X}, \mathbf{Y})$ and continuous function $g : [0, 1] \to \mathbb{R}^{m\times n}$ satisfying $\operatorname{supp}(g(t)) \subseteq \mathcal{S}_P$ (Definition 5.2.2) and $g(0) = \mathbf{X}\mathbf{Y}^\top$, there exists a feasible continuous function $f : [0, 1] \to \mathbb{R}^{m\times r} \times \mathbb{R}^{n\times r} : f(t) = (\mathbf{X}_f(t), \mathbf{Y}_f(t))$ such that:*

B1 $f(0) = (\mathbf{X}, \mathbf{Y})$.

B2 $g(t) = \mathbf{X}_f(t)\mathbf{Y}_f(t)^\top, \forall t \in [0, 1]$.

B3 $\|f(z) - f(t)\|^2 \leq \mathcal{C}\|g(z) - g(t)\|^2$.

where $\mathcal{C} = \max\left(\left|\left|\left|\mathbf{X}[R_P, P]^\dagger\right|\right|\right|^2, \left|\left|\left|\mathbf{Y}[C_P, P]^\dagger\right|\right|\right|^2\right)$.

Proof. WLOG, we assume that $P = [\![|P|]\!], R_P = [\![|R_P|]\!], C_P = [\![|C_P|]\!]$. Furthermore, we can assume $|P| \geq |R_P|$ and $\mathbf{X}[R_P, P]$ is full row rank (due to the hypothesis and the fact that P is complete).

Since $\mathcal{P}^\star = \mathcal{P} = \{P\}$, a continuous feasible function $f(t)$ must have the form: $\mathbf{X}_f(t) = \left[\begin{smallmatrix} \tilde{X}_f(t) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{smallmatrix}\right]$ and $\mathbf{Y}_f(t) = \left[\begin{smallmatrix} \tilde{Y}_f & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{smallmatrix}\right]$ where $\tilde{\mathbf{X}}_f : [0,1] \to \mathbb{R}^{|R_P| \times |P|}, \tilde{\mathbf{Y}}_f : [0,1] \to \mathbb{R}^{|C_P| \times |P|}$ are continuous functions. f is fully determined by $(\tilde{\mathbf{X}}_f(t), \tilde{\mathbf{Y}}_f(t))$.

Moreover, if $g : [0,1] \to \mathbb{R}^{m \times n}$ satisfying $\text{supp}(g(t)) \subseteq \mathcal{S}_T$, then g has to have the form: $g(t) = \left[\begin{smallmatrix} \tilde{g} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{smallmatrix}\right]$ where $\tilde{g} : [0,1] \to \mathbb{R}^{|R_P| \times |C_P|}$ is a continuous function.

Since $g(0) = \mathbf{X}\mathbf{Y}^\top$, $\tilde{g}(0) = (\mathbf{X}[R_P, P])(\mathbf{Y}[C_P, P])^\top$. Thus, to satisfy each constraint **B1**-**B3**, it is sufficient to find $\tilde{\mathbf{X}}_f$ and $\tilde{\mathbf{Y}}_f$ such that:

B1: $\tilde{\mathbf{X}}_f(0) = \mathbf{X}_{R_P,P}, \tilde{\mathbf{Y}}_f(0) = \mathbf{Y}_{C_P,P}$.

B2: $\tilde{g}(t) = \tilde{\mathbf{X}}_f(t)\tilde{\mathbf{Y}}_f(t)^\top, \forall t \in [0,1]$ because:

$$\mathbf{X}_f(t)\mathbf{Y}_f(t)^\top = \begin{pmatrix} \tilde{\mathbf{X}}_f(t)\tilde{\mathbf{Y}}_f(t)^\top & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} = \begin{pmatrix} \tilde{g}(t) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} = g(t)$$

B3: $\|\mathbf{X}'(z) - \mathbf{X}'(t)\|^2 + \|\mathbf{Y}'(z) - \mathbf{Y}'(t)\|^2 \leq \mathcal{C}\|\mathbf{A}'(z) - \mathbf{A}'(t)\|^2$ since $\|\mathbf{X}'_f(z) - \mathbf{X}_f(t)\|^2 + \|\mathbf{Y}'_f(z) - \mathbf{Y}_f(t)\|^2 = \|f(z) - f(t)\|^2$ and $\|\mathbf{A}'(z) - \mathbf{A}'(t)\|^= \|g(z) - g(t)\|^2$.

Such function exists thanks Lemma 5.6.2 (since we assume $\mathbf{X}[R_P, P]$ has full rank). □

Proof of Lemma 5.6.1. In this proof, we will use the shorthand $\mathbf{W}_P := \mathbf{W}[:, P] \in \mathbb{R}^{m \times n}$ for any matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$ (see definition of Notation) and for any $P \subseteq [\![n]\!]$.

We prove by induction on the size $\mathcal{P}$. By Lemma 5.6.3 the result is true if $|\mathcal{P}| = 1$. Assume the result is true if $|\mathcal{P}| \leq p$. We consider the case where $|\mathcal{P}| = p + 1$. Let $P \in \mathcal{P}$ and partition $\mathcal{P}$ into $\mathcal{P}' = \mathcal{P} \setminus \{P\}$ and $\{P\}$. Let $T' = \cup_{P' \in \mathcal{P}'} P' = T \setminus P$. Since $|\mathcal{P}'| = p$, we can use induction hypothesis. Define:

$$h_1(t) = (g(t) - \mathbf{X}_P\mathbf{Y}_P^\top) \odot \mathcal{S}_{\mathcal{P}'}, \qquad h_2(t) = \mathbf{X}_P\mathbf{Y}_P^\top \odot \mathcal{S}_{\mathcal{P}'} + g(t) \odot \mathcal{S}_P \setminus \mathcal{S}_{\mathcal{P}'}$$

We verify that the function $h_1(t)$ satisfying the hypotheses to use induction step: h_1 continuous, $\text{supp}(h_1(t)) \subseteq \mathcal{S}_{\mathcal{P}'}$ and finally $h_1(0) = (g(0) - \mathbf{X}_P\mathbf{Y}_P^\top) \odot \mathcal{S}_{\mathcal{P}'} = \mathbf{X}_{T'}\mathbf{Y}_{T'}^\top \odot \mathcal{S}_{\mathcal{P}'} = \mathbf{X}_{T'}\mathbf{Y}_{T'}^\top$. Using the induction hypothesis with $\mathcal{P}'$, there exists a function $f_1 : [0,1] \to \mathbb{R}^{m \times r} \times \mathbb{R}^{n \times r} : f_1(t) = (\mathbf{X}_f^1(t), \mathbf{Y}_f^1(t))$ such that:

1) $\text{supp}(\mathbf{X}_f^1(t)) \subseteq I_{T'}, \text{supp}(\mathbf{Y}_f^1(t)) \subseteq J_{T'}$.

2) $f_1(0) = (\mathbf{X}_{T'}, \mathbf{Y}_{T'})$.

3) $h_1(t) = \mathbf{X}_f^1(t)\mathbf{Y}_f^1(t)^\top, \forall t \in [0,1]$.

4) $\|f_1(z) - f_1(t)\|^2 \leq \mathcal{C}'\|h_1(z) - h_1(t)\|^2$.

where $\mathcal{C}' = \max_{P' \in \mathcal{P}'} \left(\max \left(\left\| \mathbf{X}^\dagger_{R_{P'},P'} \right\|^2, \left\| \mathbf{Y}^\dagger_{C_P,P} \right\|^2 \right) \right)$.

On the other hand, $h_2(t)$ satisfies the assumptions of Lemma 5.6.3: $h_2(t)$ is continuous and $\text{supp}(h_2(t)) = \text{supp}(\mathbf{X}_P\mathbf{Y}_P^\top \odot \mathcal{S}_{\mathcal{P}'} + g(t) \odot \mathcal{S}_P \setminus \mathcal{S}_{\mathcal{P}'}) \subseteq \text{supp}(\mathbf{X}_P\mathbf{Y}_P^\top) \cup (\mathcal{S}_P \setminus \mathcal{S}_{\mathcal{P}'}) = \mathcal{S}_P$.

In addition, since $g(0) \odot \mathcal{S}_P \setminus \mathcal{S}_{\mathcal{P}'} = (\mathbf{X}\mathbf{Y}^\top) \odot \mathcal{S}_P \setminus \mathcal{S}_{\mathcal{P}'} = (\mathbf{X}_{T'}\mathbf{Y}_{T'}^\top + \mathbf{X}_P\mathbf{Y}_P^\top) \odot \mathcal{S}_P \setminus \mathcal{S}_{\mathcal{P}'} = \mathbf{X}_P\mathbf{Y}_P^\top \odot \mathcal{S}_P \setminus \mathcal{S}_{\mathcal{P}'}$, we have $h_2(0) = \mathbf{X}_P\mathbf{Y}_P^\top \odot \mathcal{S}_{\mathcal{P}'} + g(0) \odot \mathcal{S}_P \setminus \mathcal{S}_{\mathcal{P}'} = \mathbf{X}_P\mathbf{Y}_P^\top \odot (\mathcal{S}_{\mathcal{P}'} + \mathcal{S}_P \setminus \mathcal{S}_{\mathcal{P}'}) = \mathbf{X}_P\mathbf{Y}_P^\top$. Invoking Lemma 5.6.3 with the singleton $\{P\}$, there exists a function $(\mathbf{X}_f^2(t), \mathbf{Y}_f^2(t))$ such that:

1) $\mathrm{supp}(\mathbf{X}_f^2(t)) \subseteq I_P, \mathrm{supp}(\mathbf{Y}_f^2(t)) \subseteq J_P$.

2) $f_2(0) = (\mathbf{X}_P, \mathbf{Y}_P)$.

3) $h_2(t) = \mathbf{X}_f^2(t)\mathbf{Y}_f^2(t)^\top, \forall t \in [0,1]$.

4) $\|f_2(z) - f_2(t)\|^2 \leq \max\left(\left|\left|\left|\mathbf{X}_{R_P,P}^\dagger\right|\right|\right|^2, \left|\left|\left|\mathbf{Y}_{C_P,P}^\dagger\right|\right|\right|^2\right) \|h_2(z) - h_2(t)\|^2$.

We construct the functions $f(t) = (\mathbf{X}_f(t), \mathbf{Y}_f(t))$ as:

$$\mathbf{X}_f(t) = \mathbf{X}_f^1(t) + \mathbf{X}_f^2(t), \qquad \mathbf{Y}_f(t) = \mathbf{Y}_f^1(t) + \mathbf{Y}_f^2(t)$$

We verify the validity of this construction. f is clearly feasible due to the supports of $\mathbf{X}_f^i(t), \mathbf{Y}_f^i(t), i = 1, 2$. The remaining conditions are:

A1:

$$\begin{aligned}\mathbf{X}_f(0) &= \mathbf{X}_f^1(0) + \mathbf{X}_f^2(0) = \mathbf{X}_{T'} + \mathbf{X}_P = \mathbf{X}\\ \mathbf{Y}_f(0) &= \mathbf{Y}_f^1(0) + \mathbf{Y}_f^2(0) = \mathbf{Y}_{T'} + \mathbf{Y}_P = \mathbf{Y}\end{aligned}$$

A2:

$$\begin{aligned}\mathbf{X}_f(t)\mathbf{Y}_f(t)^\top &= \mathbf{X}_f^1(t)\mathbf{Y}_f^1(t)^\top + \mathbf{X}_f^2(t)\mathbf{Y}_f^2(t)^\top\\ &= h_1(t) + h_2(t)\\ &= (g(t) - \mathbf{X}_P\mathbf{Y}_P^\top) \odot \mathcal{S}_{\mathcal{P}'} + \mathbf{X}_P\mathbf{Y}_P^\top \odot \mathcal{S}_{\mathcal{P}'} + g(t) \odot \mathcal{S}_P \setminus \mathcal{S}_{\mathcal{P}'}\\ &= g(t) \odot (\mathcal{S}_{\mathcal{P}'} + \mathcal{S}_P \setminus \mathcal{S}_{\mathcal{P}'}) = g(t)\end{aligned}$$

A3:

$$\begin{aligned}&\|f(z) - f(t)\|^2\\ &= \|f_1(z) - f_1(t)\|^2 + \|f_2(z) - f_2(t)\|^2\\ &\leq \mathcal{C}'\|h_1(z) - h_1(t)\|^2 + \max\left(\left|\left|\left|\mathbf{X}_{R_P,P}^\dagger\right|\right|\right|^2, \left|\left|\left|\mathbf{Y}_{C_P,P}^\dagger\right|\right|\right|^2\right) \|h_2(z) - h_2(t)\|^2\\ &\leq \mathcal{C}(\|h_1(z) - h_1(t)\|^2 + \|h_2(z) - h_2(t)\|^2)\\ &= \mathcal{C}(\|(g(z) - g(t)) \odot \mathcal{S}_{\mathcal{P}'}\|^2 + \|(g(z) - g(t)) \odot \mathcal{S}_P \setminus \mathcal{S}_{\mathcal{P}'}\|^2)\\ &= \mathcal{C}\|g(z) - g(t)\|^2\end{aligned}$$

□

5.6.2 Proof of Lemma 5.3.4

The proof relies on two intermediate results that we state first: Lemma 5.6.4 and Corollary 5.6.5. The idea of Lemma 5.6.4 can be found in [VBB20]. Since it is not formally proved as a lemma or theorem, we reprove it here for self-containedness. In fact, Lemma 5.6.4 and Corollary 5.6.5 are special cases of Lemma 5.3.4 with no support contraints and $\mathcal{P}^\star = \mathcal{P} = \{P\}$ respectively.

Lemma 5.6.4. *Let $\mathbf{X} \in \mathbb{R}^{R\times p}, \mathbf{Y} \in \mathbb{R}^{C\times p}, \min(R, C) \leq p$. There exists a continuous function $f(t) = (\mathbf{X}_f(t), \mathbf{Y}_f(t))$ on $[0, 1]$ such that:*

- $f(0 = (\mathbf{X}, \mathbf{Y})$.
- $\mathbf{X}\mathbf{Y}^\top = \mathbf{X}_f(t)(\mathbf{Y}_f(t))^\top, \forall t \in [0, 1]$.
- $\mathbf{X}_f(1)$ *or* $\mathbf{Y}_f(1)$ *has full row rank.*

Proof. WLOG, we assume that $m \leq r$. If X has full row rank, then one can choose constant function $f(t) = (\mathbf{X}, \mathbf{Y})$ to satisfy the conditions of the lemma. Therefore, we can focus on the case where $\texttt{rank}(\mathbf{X}) = q < m$. WLOG, we can assume that the first q columns of $\mathbf{X}$ (i.e., $\mathbf{X}[:, 1], \ldots, \mathbf{X}[:, q]$) are linearly independent. The remaining columns of $\mathbf{X}$ can be expressed as:

$$\mathbf{X}_k = \sum_{i=1}^{q} \alpha_i^k \mathbf{X}[:, i], \forall q < k \leq r$$

We define a matrix $\tilde{Y}$ by their columns as follow:

$$\tilde{\mathbf{Y}}_i = \begin{cases} \mathbf{Y}[:, i] + \sum_{k=q+1}^{r} \alpha_i^k \mathbf{Y}[:, k] & \text{if } i \leq q \\ 0 & \text{otherwise} \end{cases}$$

By construction, we have $\mathbf{X}\mathbf{Y}^\top = \mathbf{X}\tilde{\mathbf{Y}}^\top$. We define the function $f_1 : [0, 1] \to \mathbb{R}^{m\times r} \times \mathbb{R}^{n\times r}$ as:

$$f_1(t) = (\mathbf{X}, (1 - t)\mathbf{Y} + t\tilde{\mathbf{Y}})$$

This function will not change the value of f since we have:

$$\mathbf{X}((1 - t)\mathbf{Y}^\top + t\tilde{\mathbf{Y}}^\top) = (1 - t)\mathbf{X}\mathbf{Y}^\top + t\mathbf{X}\tilde{\mathbf{Y}}^\top = \mathbf{X}\mathbf{Y}^\top.$$

Let $\tilde{\mathbf{X}}$ be a matrix whose first q columns are identical to that of $\mathbf{X}$ and $\texttt{rank}(\tilde{\mathbf{X}}) = m$. The second function f_2 defined as:

$$f_2(t) = ((1 - t)\mathbf{X} + t\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$$

also has their product unchanged (since first q columns of $(1 - t)\mathbf{X} + t\tilde{\mathbf{X}}$ are constant and last $(r - q)$ rows of $\tilde{\mathbf{Y}}$ are zero). Moreover, $f_2(0) = (\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$ where $\tilde{\mathbf{X}}$ has full row rank. Therefore, the concatenation of two functions f_1 and f_2 (and shrink t by a factor of 2) are the desired function f. □

Corollary 5.6.5. *Consider I, J support constraints of Equation (2.4.FSMF) with $\mathcal{P}^\star = \mathcal{P} = \{P\}$. There is a feasible continuous function $f : [0,1] \mapsto \mathbb{R}^{m\times r} \times \mathbb{R}^{n\times r} : f(t) = (\mathbf{X}_f(t), \mathbf{Y}_f(t))$ such that:*

1. *$f(0) = (\mathbf{X}, \mathbf{Y})$;*
2. *$\mathbf{X}_f(t)(\mathbf{Y}_f(t))^\top = \mathbf{X}\mathbf{Y}^\top, \forall t \in [0,1]$;*
3. *$(\mathbf{X}_f(1))[R_P, P]$ or $(\mathbf{Y}_f(1))[C_P, P]$ has full row rank.*

Proof of Corollary 5.6.5. WLOG, up to permuting columns, we can assume $P = [\![|P|]\!], R_P = [\![|R_P|]\!]$ and $C_P = [\![|C_P|]\!]$ (R_P and C_P are defined in Definition Definition 5.2.2). A feasible function $f = (\mathbf{X}_f(t), \mathbf{Y}_f(t))$ has the form:

$$\mathbf{X}_f(t) = \begin{pmatrix} \tilde{\mathbf{X}}_f(t) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}, \mathbf{Y}_f(t) = \begin{pmatrix} \tilde{\mathbf{Y}}_f(t) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$$

where $\tilde{\mathbf{X}}_f : [0,1] \mapsto \mathbb{R}^{R_P \times P}$, $\tilde{\mathbf{Y}}_f : [0,1] \mapsto \mathbb{R}^{C_P \times P}$.

Since P is a CEC, we have $p \geq \min(R_P, C_P)$. Hence we can use Lemma 5.6.4 to build $(\tilde{\mathbf{X}}_f(t), \tilde{\mathbf{Y}}_f(t))$ satisfying all conditions of Lemma 5.6.4. Such $(\tilde{\mathbf{X}}_f(t), \tilde{\mathbf{Y}}_f(t))$ fully determines f and make f our desirable function. □

Proof of Lemma 5.3.4. First, we decompose $\mathbf{X}$ and $\mathbf{Y}$ as:

$$\mathbf{X} = \mathbf{X}_{\bar{T}} + \sum_{P\in\mathcal{P}^\star} \mathbf{X}_P, \qquad \mathbf{Y} = \mathbf{Y}_{\bar{T}} + \sum_{P\in\mathcal{P}^\star} \mathbf{Y}_P$$

where $\mathbf{X}_P$ (resp. $\mathbf{Y}_P$) is a shorthand for $\mathbf{X}[:, P]$ (resp. $\mathbf{Y}[:, P]$).

Since $\bar{T}$ and $P \in \mathcal{P}^\star$ form a partition of $[\![r]\!]$, the product $\mathbf{X}\mathbf{Y}^\top$ can be written as:

$$\mathbf{X}\mathbf{Y}^\top = \mathbf{X}_{\bar{T}}\mathbf{Y}_{\bar{T}}^\top + \sum_{P\in\mathcal{P}^\star} \mathbf{X}_P\mathbf{Y}_P^\top.$$

For each $P \in \mathcal{P}^\star$, (I_P, J_P) contains one CEC. By applying Corollary 5.6.5, we can build continuous functions $(\mathbf{X}_f^P(t), \mathbf{Y}_f^P(t))$, $\mathrm{supp}(\mathbf{X}_f^P(t)) \subseteq I_P, \mathrm{supp}(\mathbf{Y}_f^P(t)) \subseteq J_P, \forall t \in [0,1]$ such that:

1. $(\mathbf{X}_f^P(0), \mathbf{Y}_f^P(0)) = (\mathbf{X}_P, \mathbf{Y}_P)$.
2. $\mathbf{X}_f^P(t)(\mathbf{Y}_f^P(t))^\top = \mathbf{X}_P\mathbf{Y}_P^\top, \forall t \in [0,1]$.
3. $(\mathbf{X}_f^P(1))[R_P, P]$ or $(\mathbf{Y}_f^P(1))[C_P, P]$ has full row rank.

Our desirable $f(t) = (\mathbf{X}_f(t), \mathbf{Y}_f(t))$ is defined as:

$$\mathbf{X}_f(t) = \mathbf{X}_{\bar{T}} + \sum_{P\in\mathcal{P}^\star} \mathbf{X}_f^P(t), \qquad \mathbf{Y}(t) = \mathbf{Y}_{\bar{T}} + \sum_{P\in\mathcal{P}^\star} \mathbf{Y}_f^P(t)$$

To conclude, it is immediate to check that $f = (\mathbf{X}_f(t), \mathbf{Y}_f(t))$ is feasible, $f(0) = (\mathbf{X}, \mathbf{Y})$, $f(1)$ is CEC-full-rank and $\mathbf{X}_f(t)\mathbf{Y}_f(t)^\top = \mathbf{X}\mathbf{Y}^\top, \forall t \in [0,1]$. □

5.6.3 Proof of Lemma 5.3.5

Denote $\mathbf{Z} = \mathbf{X}\mathbf{Y}^\top$, we construct f such that $\mathbf{X}_f(t)\mathbf{Y}_f(t)^\top = \mathbf{B}(t)$, where $\mathbf{B}(t) = \mathbf{Z} \odot \bar{\mathcal{S}}_T + (\mathbf{A}t + \mathbf{Z}(1-t)) \odot \mathcal{S}_T$. Such function f makes $L(\mathbf{X}_f(t), \mathbf{Y}_f(t))$ non-increasing since:

$$\begin{aligned} \|\mathbf{A} - \mathbf{X}_f(t)\mathbf{Y}_f(t)^\top\|^2 &= \|\mathbf{A} - \mathbf{B}(t)\|^2 \\ &= \|(\mathbf{A} - \mathbf{Z}) \odot \bar{\mathcal{S}}_T\|^2 + (1-t)^2\|(\mathbf{A} - \mathbf{Z}) \odot \mathcal{S}_T\|^2 \end{aligned} \tag{5.15}$$

Thus, the rest of the proof is devoted to show that such a function f exists by using Lemma 5.6.1. Consider the function $g(t) = \mathbf{B}(t) - \mathbf{X}_{\bar{T}}(\mathbf{Y}_{\bar{T}})^\top$. We have that $g(t)$ is continuous, $g(0) = \mathbf{B}(0) - \mathbf{X}_{\bar{T}}(\mathbf{Y}_{\bar{T}})^\top = \mathbf{Z} - \mathbf{X}_{\bar{T}}(\mathbf{Y}_{\bar{T}})^\top = \mathbf{X}_T(\mathbf{Y}_T)^\top$ and:

$$\begin{aligned} g(t) \odot \bar{\mathcal{S}}_T &= (\mathbf{B}(t) - \mathbf{X}_{\bar{T}}(\mathbf{Y}_{\bar{T}})^\top) \odot \bar{\mathcal{S}}_T \\ &= (\mathbf{Z} - \mathbf{X}_{\bar{T}}(\mathbf{Y}_{\bar{T}})^\top) \odot \bar{\mathcal{S}}_T \\ &= (\mathbf{X}_T\mathbf{Y}_T^\top) \odot \bar{\mathcal{S}}_T = \mathbf{0} \end{aligned}$$

which shows $\text{supp}(g(t)) \subseteq \mathcal{S}_T$. Since $(\mathbf{X}_T, \mathbf{Y}_T)$ is CEC-full-rank (by our assumption, $(\mathbf{X}, \mathbf{Y})$ is CEC-full-rank), invoking Lemma 5.6.1 with (I_T, J_T), there exists $f^T(t) = (\mathbf{X}_f^T(t), \mathbf{Y}_f^T(t))$ such that:

D1 $\text{supp}(\mathbf{X}_f^T(t)) \subseteq I_T, \text{supp}(\mathbf{Y}_f^C(t)) \subseteq J_T$.

D2 $f^T(0) = (\mathbf{X}_T, \mathbf{Y}_T)$.

D3 $g(t) = \mathbf{X}_f^T(t)(\mathbf{Y}_f^T(t))^\top, \forall t \in [0,1]$.

We can define our desired function $f(t) = (\mathbf{X}_f(t), \mathbf{Y}_f(t))$ as:

$$\mathbf{X}_f(t) = \mathbf{X}_{\bar{T}} + \mathbf{X}_f^T(t), \qquad \mathbf{Y} = \mathbf{Y}_{\bar{T}} + \mathbf{Y}_f^T(t)$$

f is clearly feasible due to (**D1**). The remaining condition to be checked is:

- First condition:

$$\mathbf{X}_f(0) = \mathbf{X}_f^T(0) + \mathbf{X}_{\bar{T}} = \mathbf{X}_T + \mathbf{X}_{\bar{T}} = \mathbf{X}, \quad \mathbf{Y}_f(0) = \mathbf{Y}_f^T(0) + \mathbf{Y}_{\bar{T}} = \mathbf{Y}_T + \mathbf{Y}_{\bar{T}} = \mathbf{Y}$$

- Second condition: holds thanks to Equation (5.15) and:

$$\mathbf{X}_f(t)(\mathbf{Y}_f(t))^\top = \mathbf{X}_{\bar{T}}\mathbf{Y}_{\bar{T}}^\top + \mathbf{X}_f^C(t)(\mathbf{Y}_f^C(t))^\top = \mathbf{X}_{\bar{T}}\mathbf{Y}_{\bar{T}}^\top + g(t) = \mathbf{B}(t)$$

- Third condition:

$$\begin{aligned} (\mathbf{A} - \mathbf{X}_f(1)(\mathbf{Y}_f(1))^\top) \odot \mathcal{S}_T &= (\mathbf{A} - \mathbf{B}(1)) \odot \mathcal{S}_T \\ &= (\mathbf{A} - \mathbf{Z} \odot \bar{\mathcal{S}}_T - \mathbf{A} \odot \mathcal{S}_T) \odot \mathcal{S}_T = \mathbf{0} \end{aligned}$$

5.6.4 Proof of Lemma 5.3.6

Consider $\mathbf{X}_T, \mathbf{X}^i_{\bar{T}}, \mathbf{Y}_T, \mathbf{Y}^i_{\bar{T}}, i = 1, 2$ as in Definition 5.5.1. We redefine $\mathbf{A}' = \mathbf{A} \odot \bar{\mathcal{S}}_T, I' = I^1_{\bar{T}}, J' = J^1_{\bar{T}}$ as in Theorem 5.2.9.

In light of Corollary 5.5.3, an optimal solution $(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$ has the following form:

1) $\tilde{\mathbf{X}}^1_{\bar{T}} = \tilde{\mathbf{X}} \odot I^1_{\bar{T}}, \tilde{\mathbf{Y}}^1_{\bar{T}} = \tilde{\mathbf{Y}} \odot J^1_{\bar{T}}$ is an optimal solution of (2.4.FSMF) with $(\mathbf{A}', I', J')$.

2) $\tilde{\mathbf{X}}^2_{\bar{T}} = \tilde{\mathbf{X}} \odot I^2_{\bar{T}}, \tilde{\mathbf{Y}}^2_{\bar{T}} = \tilde{\mathbf{Y}} \odot J^2_{\bar{T}}$ can be arbitrary.

3) $\tilde{\mathbf{X}}_T = \tilde{\mathbf{X}} \odot I_T, \tilde{\mathbf{Y}}_T = \tilde{\mathbf{Y}} \odot J_T$ satisfy:

$$\tilde{\mathbf{X}}_T \tilde{\mathbf{Y}}_T^\top = (\mathbf{A} - \sum_{(i,j)\neq(1,1)} \tilde{\mathbf{X}}^i_{\bar{T}} \tilde{\mathbf{Y}}^j_{\bar{T}})^\top \odot \mathcal{S}_T$$

Since (I', J') has its support constraints satisfying Theorem 5.2.3 assumptions as shown in Theorem 5.2.9, by Theorem 5.3.1, there exists a function $(\mathbf{X}^{\bar{T}}_f(t), \mathbf{Y}^{\bar{T}}_f(t))$ such that:

1) $\text{supp}(\mathbf{X}^{\bar{T}}_f(t)) \subseteq I^1_{\bar{T}}, \text{supp}(\mathbf{Y}^{\bar{T}}_f(t)) \subseteq J^1_{\bar{T}}$.

2) $\mathbf{X}^{\bar{T}}_f(0) = \mathbf{X}^1_{\bar{T}}, \mathbf{Y}^{\bar{T}}_f(0) = \mathbf{Y}^1_{\bar{T}}$.

3) $L'(\mathbf{X}^{\bar{T}}_f(t), \mathbf{Y}^{\bar{T}}_f(t)) = \|\mathbf{A}' - \mathbf{X}^{\bar{T}}_f(t)\mathbf{Y}^{\bar{T}}_f(t)^\top\|^2$ is non-increasing.

4) $(\mathbf{X}^{\bar{T}}_f(1), \mathbf{Y}^{\bar{T}}_f(1))$ is an optimal solution of the instance of (2.4.FSMF) with $(\mathbf{A}', I', J')$.

Consider the function $g(t) = \left(\mathbf{A} - (\mathbf{X}^{\bar{T}}_f(t) + \mathbf{X}^2_{\bar{T}})(\mathbf{Y}^{\bar{T}}_f(t) + \mathbf{Y}^2_{\bar{T}})^\top\right) \odot \mathcal{S}_T$. This construction makes $g(0) = \mathbf{X}_T \mathbf{Y}_T^\top$. Indeed,

$$\begin{aligned} g(0) &= \left(\mathbf{A} - (\mathbf{X}^{\bar{T}}_f(0) + \mathbf{X}^2_{\bar{T}})(\mathbf{Y}^{\bar{T}}_f(0) + \mathbf{Y}^2_{\bar{T}})^\top\right) \odot \mathcal{S}_T \\ &= \left(\mathbf{A} - (\mathbf{X}^1_{\bar{T}} + \mathbf{X}^2_{\bar{T}})(\mathbf{Y}^1_{\bar{T}} + \mathbf{Y}^2_{\bar{T}})^\top\right) \odot \mathcal{S}_T \\ &\overset{(1)}{=} \left(\mathbf{X}\mathbf{Y}^\top - (\mathbf{X}^1_{\bar{T}} + \mathbf{X}^2_{\bar{T}})(\mathbf{Y}^1_{\bar{T}} + \mathbf{Y}^2_{\bar{T}})^\top\right) \odot \mathcal{S}_T \\ &\overset{(2)}{=} \mathbf{X}_T \mathbf{Y}_T^\top \end{aligned}$$

where (1) holds by the hypothesis $(\mathbf{A} - \mathbf{X}\mathbf{Y}^\top) \odot \mathcal{S}_T = \mathbf{0}$, and (2) holds by Equation (5.10) and $\text{supp}(\mathbf{X}_T \mathbf{Y}_T^\top) \subseteq \mathcal{S}_T$. Due to our hypothesis $(\mathbf{X}, \mathbf{Y})$ is CEC-full-rank, $(\mathbf{X}_T, \mathbf{Y}_T)$ is CEC-full-rank. In addition, $g(t)$ continuous, $\text{supp}(g(t)) \subseteq \mathcal{S}_T$ and $g(0) = \mathbf{X}_T \mathbf{Y}_T^\top$. Invoking Lemma 5.6.1 with (I_T, J_T), there exist functions $(\mathbf{X}^C_f(t), \mathbf{Y}^C_f(t))$ satisfying:

1) $\text{supp}(\mathbf{X}^T_f(t)) \subseteq I_T, \text{supp}(\mathbf{Y}^T_f(t)) \subseteq J_T$.

2) $f^T(0) = (\mathbf{X}_T, \mathbf{Y}_T)$.

3) $g(t) = \mathbf{X}^T_f(t)\mathbf{Y}^T_f(t)^\top, \forall t \in [0, 1]$.

Finally, one can define the function $\mathbf{X}_f(t), \mathbf{Y}_f(t)$ satisfying Lemma 5.3.6 as:

$$\mathbf{X}_f(t) = \mathbf{X}_f^{\bar{T}}(t) + \mathbf{X}_f^{C}(t) + \mathbf{X}_{\bar{T}}^2, \qquad \mathbf{Y}_f(t) = \mathbf{Y}_f^{\bar{T}}(t) + \mathbf{Y}_f^{C}(t) + \mathbf{Y}_{\bar{T}}^2$$

f is feasible due to the supports of $X_f^P(t), Y_f^P(t), P \in \{\bar{T}, C\}$ and $X_{\bar{T}}^2, Y_{\bar{T}}^2$. The remaining conditions are satisfied as:

- First condition:

$$\mathbf{X}_f(0) = \mathbf{X}_f^{\bar{T}}(0) + \mathbf{X}_f^{C}(0) + \mathbf{X}_{\bar{T}}^2 = \mathbf{X}_{\bar{T}}^1 + \mathbf{X}_T + \mathbf{X}_{\bar{T}}^2 = \mathbf{X}$$
$$\mathbf{Y}_f(0) = \mathbf{Y}_f^{\bar{T}}(0) + \mathbf{Y}_f^{C}(0) + \mathbf{Y}_{\bar{T}}^2 = \mathbf{Y}_{\bar{T}}^1 + \mathbf{Y}_T + \mathbf{Y}_{\bar{T}}^2 = \mathbf{Y}$$

- Second condition:

$$\begin{aligned}
&\|\mathbf{A} - \mathbf{X}_f(t)\mathbf{Y}_f(t)^\top\|^2 = \|\mathbf{A} - \mathbf{X}_f^T(t)(\mathbf{Y}_f^T(t))^\top - (\mathbf{X}_f^{\bar{T}}(t) + \mathbf{X}_{\bar{T}}^2)(\mathbf{Y}_f^{\bar{T}}(t) + \mathbf{Y}_{\bar{T}}^2)^\top\|^2 \\
&= \|g(t) - \mathbf{X}_f^T(t)\mathbf{Y}_f^T(t)^\top\|^2 + \|(\mathbf{A} - \mathbf{X}_f^{\bar{T}}(t)(\mathbf{Y}_f^{\bar{T}}(t))^\top) \odot \mathcal{S}_\mathcal{P} \setminus \mathcal{S}_T\|^2 + \|\mathbf{A} \odot \bar{\mathcal{S}}_\mathcal{P}\|^2 \\
&= \|(\mathbf{A}' - \mathbf{X}_f^{\bar{T}}(t)(\mathbf{Y}_f^{\bar{T}}(t))^\top) \odot \mathcal{S}_\mathcal{P} \setminus \mathcal{S}_T\|^2 + \|\mathbf{A} \odot \bar{\mathcal{S}}_\mathcal{P}\|^2 \\
&\overset{(5.11)}{=} \|\mathbf{A}' - \mathbf{X}_f^{\bar{T}}(t)(\mathbf{Y}_f^{\bar{T}}(t))^\top\|^2
\end{aligned}$$

  Since $\|\mathbf{A}' - \mathbf{X}_f^{\bar{T}}(t)(\mathbf{Y}_f^{\bar{T}}(t))^\top\|^2$ is non-increasing, so is $\|\mathbf{A} - \mathbf{X}_f(t)\mathbf{Y}_f(t)^\top\|^2$.

- Third condition: By Theorem 5.2.9, $(\mathbf{X}_f(1), \mathbf{Y}_f(1))$ is a global minimizer since $\|\mathbf{A} - \mathbf{X}_f(1)\mathbf{Y}_f(1)^\top\|^2 = \|\mathbf{A}' - \mathbf{X}_f^{\bar{T}}(1)(\mathbf{Y}_f^{\bar{T}}(1))^\top\|^2$ where $(\mathbf{X}_f^{\bar{T}}(1), \mathbf{Y}_f^{\bar{T}}(1))$ is an optimal solution of the instance of (2.4.FSMF) with $(\mathbf{A}', I', J')$.

5.6.5 Proof of Theorem 5.3.8

The following corollary is necessary for the proof of Theorem 5.3.8.

Corollary 5.6.6. *Consider I, J support constraints of (2.4.FSMF), such that $\mathcal{P}^\star = \mathcal{P}$. Given any feasible CEC-full-rank point $(\mathbf{X}, \mathbf{Y})$ and any $\mathbf{B}$ satisfying $\operatorname{supp}(\mathbf{B}) \subseteq \mathcal{S}_\mathcal{P}$, there exists $(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$ such that:*

E1 $\operatorname{supp}(\tilde{\mathbf{X}}) \subseteq I, \operatorname{supp}(\tilde{\mathbf{Y}}) \subseteq J$

E2 $\tilde{\mathbf{X}}\tilde{\mathbf{Y}}^\top = B$.

E3 $\|\mathbf{X} - \tilde{\mathbf{X}}\|^2 + \|\mathbf{Y} - \tilde{\mathbf{Y}}\|^2 \leq \mathcal{C}\|\mathbf{X}\mathbf{Y}^\top - \mathbf{B}\|^2$.

where $\mathcal{C} = \max_{P \in \mathcal{P}^\star} \left(\max \left(\left\| \left\| \mathbf{X}_{R_P,P}^\dagger \right\| \right\|^2, \left\| \left\| \mathbf{Y}_{C_P,P}^\dagger \right\| \right\|^2 \right) \right)$.

Proof. Corollary 5.6.6 is an application of Lemma 5.6.1. Consider the function $g(t) = (1-t)\mathbf{X}\mathbf{Y}^\top + t\mathbf{B}$. By construction, $g(t)$ is continuous, $g(0) = \mathbf{X}\mathbf{Y}^\top$ and $\operatorname{supp}(g(t)) \subseteq \operatorname{supp}(\mathbf{X}\mathbf{Y}^\top) \cup \operatorname{supp}(\mathbf{B}) = \mathcal{S}_\mathcal{P}$. Since $(\mathbf{X}, \mathbf{Y})$ is CEC-full-rank, there exists a feasible function $f(t) = (\mathbf{X}_f(t), \mathbf{Y}_f(t))$ satisfying **A1** - **A3** by using Lemma 5.6.1.

We choose $(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}) = (\mathbf{X}_f(1), \mathbf{Y}_f(1))$. The verification of constraints is as follow:

E1: f is feasible.

E2: $\tilde{\mathbf{X}}\tilde{\mathbf{Y}}^\top = \mathbf{X}_f(1)\mathbf{Y}_f(1)^\top \overset{\text{A2}}{=} g(1) = \mathbf{B}$.

E3: $\|\mathbf{X}-\tilde{\mathbf{X}}\|^2 + \|\mathbf{Y}-\tilde{\mathbf{Y}}\|^2 \overset{\text{A1}}{=} \|f(1)-f(0)\|^2 \overset{\text{A3}}{\leq} \mathcal{C}\|g(0)-g(1)\|^2 \leq \mathcal{C}\|\mathbf{X}\mathbf{Y}^\top - \mathbf{B}\|^2$.

□

With Corollary 5.6.6, we proceed to the proof of Theorem 5.3.8.

Proof of Theorem 5.3.8. In this proof, we will use the shorthand $\mathbf{W}_P := \mathbf{W}[:, P] \in \mathbb{R}^{m\times n}$ for any matrix $\mathbf{W} \in \mathbb{R}^{m\times n}$ (see definition of Notation) and for any $P \subseteq [\![n]\!]$.

As mentioned in the sketch of the proof, given any $(\mathbf{X}, \mathbf{Y})$ not CEC-full-rank, Lemma 5.3.4 shows the existence of a path f along which L is constant and f connects $(\mathbf{X}, \mathbf{Y})$ to some CEC-full-rank $(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$. Therefore, this proof will be entirely devoted to show that a feasible CEC-full-rank solution $(\mathbf{X}, \mathbf{Y})$ cannot be a spurious local minimum. This fact will be shown by the two following steps:

FIRST STEP: Consider the function $L(\mathbf{X}, \mathbf{Y})$, we have:

$$L(\mathbf{X}, \mathbf{Y}) = \|\mathbf{A} - \mathbf{X}\mathbf{Y}^\top\|^2 = \|\mathbf{A} - \sum_{P'\in\mathcal{P}^\star} \mathbf{X}_{P'}\mathbf{Y}_{P'}^\top - \mathbf{X}_{\bar{T}}\mathbf{Y}_{\bar{T}}^\top\|^2$$

If $(\mathbf{X}, \mathbf{Y})$ is truly a local minimum, then $\forall P \in \mathcal{P}^\star$, $(\mathbf{X}_P, \mathbf{Y}_P)$ is also the local minimum of the following function:

$$L'(\mathbf{X}_P, \mathbf{Y}_P) = \|(\mathbf{A} - \sum_{P'\neq P} \mathbf{X}_{P'}\mathbf{Y}_{P'}^\top - \mathbf{X}_{\bar{T}}Y_{\bar{T}}^\top) - \mathbf{X}_P\mathbf{Y}_P^\top\|^2$$

where L' is equal to L but we optimize only w.r.t $(\mathbf{X}_P, \mathbf{Y}_P)$ while fixing the other coefficients. In other words, $(\mathbf{X}_P, \mathbf{Y}_P)$ is a local minimum of the problem:

$$\begin{aligned} &\underset{\mathbf{X}'\in\mathbb{R}^{m\times r}, \mathbf{Y}'\in\mathbb{R}^{n\times r}}{\text{Minimize}} && L'(\mathbf{X}', \mathbf{Y}') = \|\mathbf{B} - \mathbf{X}'\mathbf{Y}'^\top\|^2 \\ &\text{Subject to:} && \texttt{supp}(\mathbf{X}') \subseteq I_P \text{ and } \texttt{supp}(\mathbf{Y}') \subseteq J_P \end{aligned}$$

where $\mathbf{B} = \mathbf{A} - \sum_{P'\neq P} \mathbf{X}_{P'}\mathbf{Y}_{P'}^\top - \mathbf{X}_{\bar{T}}\mathbf{Y}_{\bar{T}}$. Since all columns of I_P (resp. of J_P) are identical, all rank-one contribution supports are totally overlapping. Thus, all local minima are global minima (Theorem 5.3.1). Global minima are attained when $\mathbf{X}_P\mathbf{Y}_P^\top = \mathbf{B} \odot \mathcal{S}_P$ due to the expressivity of a CEC (Lemma 5.2.6). Thus, for any $P \in \mathcal{P}^\star$, $\forall (i, j) \in \mathcal{S}_P$, we have:

$$0 = (\mathbf{B} - \mathbf{X}_P\mathbf{Y}_P^\top)_{i,j} = (\mathbf{A} - \sum_{P'\in\mathcal{P}^\star} \mathbf{X}_{P'}\mathbf{Y}_{P'}^\top - \mathbf{X}_{\bar{T}}\mathbf{Y}_{\bar{T}}^\top)_{i,j} = (\mathbf{A} - \mathbf{X}\mathbf{Y}^\top)_{i,j}$$

which implies Equation (5.7).

SECOND STEP: In this step, we assume that Equation (5.7) holds. Consider $\mathbf{X}_T, \mathbf{Y}_T$ and $\mathbf{X}^i_{\bar{T}}, , \mathbf{Y}^i_{\bar{T}}, i = 1, 2$ as in Definition 5.2.8. Let $\mathbf{A}' = \mathbf{A} \odot \bar{\mathcal{S}}_T, I' = I^1_{\bar{T}}, J' = J^1_{\bar{T}}$.

We consider two possibilities. First, if $(\mathbf{X}^1_{\bar{T}}, \mathbf{Y}^1_{\bar{T}})$ is an optimal solution of the instance of (2.4.FSMF) with $(\mathbf{A}', I', J')$, by Corollary 5.5.3, $(\mathbf{X}, \mathbf{Y})$ is an optimal solution of (2.4.FSMF) with $(\mathbf{A}, I, J)$ (since Equation (5.7) holds). Hence it cannot be a spurious

local minimum. We now focus on the second case, where $(\mathbf{X}^1_{\bar{T}}, \mathbf{Y}^1_{\bar{T}})$ is *not* the optimal solution of the instance of (2.4.FSMF) with $(\mathbf{A}', I', J')$. We show that in this case, in any neighborhood of $(\mathbf{X}, \mathbf{Y})$, there exists a point $(\mathbf{X}', \mathbf{Y}')$ such that $\text{supp}(\mathbf{X}') \subseteq I$, $\text{supp}(\mathbf{Y}') \subseteq J'$ and $L(\mathbf{X}, \mathbf{Y}) > L(\mathbf{X}', \mathbf{Y}')$. Thus $(\mathbf{X}, \mathbf{Y})$ cannot be a local minimum.

Since $(I^1_{\bar{T}}, J^1_{\bar{T}})$ satisfies Theorem 5.2.3 assumptions, (2.4.FSMF) has no spurious local minima (Theorem 5.3.1). As $(\mathbf{X}^1_{\bar{T}}, \mathbf{Y}^1_{\bar{T}})$ is not an optimal solution, it cannot be a local minimum either, i.e., in any neighborhood of $(\mathbf{X}^1_{\bar{T}}, \mathbf{Y}^1_{\bar{T}})$, there exists $(\dot{\mathbf{X}}, \dot{\mathbf{Y}})$ with $\text{supp}(\tilde{\mathbf{X}}^1_{\bar{T}}) \subseteq I', \text{supp}(\tilde{\mathbf{Y}}^1_{\bar{T}}) \subseteq J'$ and

$$\|\mathbf{A}' - \mathbf{X}^1_{\bar{T}}(\mathbf{Y}^1_{\bar{T}})^\top\|^2 > \|\mathbf{A}' - \tilde{\mathbf{X}}^1_{\bar{T}}(\tilde{\mathbf{Y}}^1_{\bar{T}})^\top\|^2 \tag{5.16}$$

By Equation (5.11), we have:

$$\begin{aligned} \|\mathbf{A}' - (\mathbf{X}^1_{\bar{T}})(\mathbf{Y}^1_{\bar{T}})^\top\|^2 &= \|(\mathbf{A} - (\mathbf{X}^1_{\bar{T}})(\mathbf{Y}^1_{\bar{T}})^\top) \odot \mathcal{S}_{\mathcal{P}} \setminus \mathcal{S}_T\|^2 + \|\mathbf{A} \odot \bar{\mathcal{S}}_{\mathcal{P}}\|^2 \\ \|\mathbf{A}' - (\tilde{\mathbf{X}}^1_{\bar{T}})(\tilde{\mathbf{Y}}^1_{\bar{T}})^\top\|^2 &= \|(\mathbf{A} - (\tilde{\mathbf{X}}^1_{\bar{T}})(\tilde{\mathbf{Y}}^1_{\bar{T}})^\top) \odot \mathcal{S}_{\mathcal{P}} \setminus \mathcal{S}_T\|^2 + \|\mathbf{A} \odot \bar{\mathcal{S}}_{\mathcal{P}}\|^2 \end{aligned} \tag{5.17}$$

By Equation (5.16) and Equation (5.17) we have:

$$\|(\mathbf{A} - (\mathbf{X}^1_{\bar{T}})(\mathbf{Y}^1_{\bar{T}})^\top) \odot \mathcal{S}_{\mathcal{P}} \setminus \mathcal{S}_T\|^2 > \|(\mathbf{A} - \tilde{\mathbf{X}}^1_{\bar{T}}(\tilde{\mathbf{Y}}^1_{\bar{T}})^\top) \odot \mathcal{S}_{\mathcal{P}} \setminus \mathcal{S}_T\|^2 \tag{5.18}$$

Consider the matrix: $\mathbf{B} := \left(\mathbf{A} - (\tilde{\mathbf{X}}^1_{\bar{T}} + \mathbf{X}^2_{\bar{T}})(\tilde{\mathbf{Y}}^1_{\bar{T}} + \mathbf{Y}^2_{\bar{T}})^\top\right) \odot \mathcal{S}_T$. Since $\text{supp}(\mathbf{B}) \subseteq \mathcal{S}_T$ and $(\mathbf{X}_T, \mathbf{Y}_T)$ is CEC-full-rank (we assume $(\mathbf{X}, \mathbf{Y})$ is CEC-full-rank), by Corollary 5.6.6, there exists $(\tilde{\mathbf{X}}_T, \tilde{\mathbf{Y}}_T)$ such that:

1) $\text{supp}(\tilde{\mathbf{X}}_T) \subseteq I_T, \text{supp}(\tilde{\mathbf{Y}}_T) \subseteq J_T$.
2) $\tilde{\mathbf{X}}_T \tilde{\mathbf{Y}}_T^\top = \mathbf{B}$.
3) $\|\mathbf{X}_T - \tilde{\mathbf{X}}_T\|^2 + \|\mathbf{Y}_T - \tilde{\mathbf{Y}}_T\|^2 \leq \mathcal{C}\|\mathbf{X}_T\mathbf{Y}_T^\top - \mathbf{B}\|^2$.

where $\mathcal{C} = \max\limits_{P \in \mathcal{P}^\star} \left(\max\left(\left\|\left\|\mathbf{X}^\dagger_{R_P,P}\right\|\right\|^2, \left\|\left\|\mathbf{Y}^\dagger_{C_P,P}\right\|\right\|^2\right)\right)$. We define the point$(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$ as:

$$\tilde{\mathbf{X}} = \tilde{\mathbf{X}}_T + \tilde{\mathbf{X}}^1_{\bar{T}} + \mathbf{X}^2_{\bar{T}}, \qquad \tilde{\mathbf{Y}} = \tilde{\mathbf{Y}}_T + \tilde{\mathbf{Y}}^1_{\bar{T}} + \mathbf{Y}^2_{\bar{T}}$$

The point $(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$ still satisfies Equation (5.7). Indeed,

$$\begin{aligned} (\mathbf{A} - \tilde{\mathbf{X}}\tilde{\mathbf{Y}}^\top) \odot \mathcal{S}_T &= \left(\mathbf{A} - \tilde{\mathbf{X}}_T\tilde{\mathbf{Y}}_T^\top - (\tilde{\mathbf{X}}^1_{\bar{T}} + \mathbf{X}^2_{\bar{T}})(\tilde{\mathbf{Y}}^1_{\bar{T}} + \mathbf{Y}^2_{\bar{T}})^\top\right) \odot \mathcal{S}_T \\ &= (\mathbf{B} - \tilde{\mathbf{X}}_T\tilde{\mathbf{Y}}_T^\top) \odot \mathcal{S}_T = \mathbf{0}. \end{aligned} \tag{5.19}$$

It is clear that $(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$ satisfies $\text{supp}(\tilde{\mathbf{X}}) \subseteq I$, $\text{supp}(\tilde{\mathbf{Y}}) \subseteq J$ due to the support of its components $(\tilde{\mathbf{X}}_T, \tilde{\mathbf{Y}}_T), (\tilde{\mathbf{X}}^1_{\bar{T}}, \tilde{\mathbf{Y}}^1_{\bar{T}}), (\mathbf{X}^2_{\bar{T}}, \mathbf{Y}^2_{\bar{T}})$. Moreover, we have:

$$\begin{aligned} \|\mathbf{A} - \tilde{\mathbf{X}}\tilde{\mathbf{Y}}^\top\|^2 &= \|(\mathbf{A} - \tilde{\mathbf{X}}\tilde{\mathbf{Y}}^\top) \odot \mathcal{S}_T\|^2 + \|(A - \tilde{\mathbf{X}}\tilde{\mathbf{Y}}^\top) \odot \mathcal{S}_{\mathcal{P}} \setminus \mathcal{S}_T\|^2 + \|\mathbf{A} \odot \bar{\mathcal{S}}_{\mathcal{P}}\|^2 \\ &\overset{(5.19)}{=} \|(\mathbf{A} - \tilde{\mathbf{X}}^1_{\bar{T}}(\tilde{\mathbf{Y}}^1_{\bar{T}})^\top) \odot \mathcal{S}_{\mathcal{P}} \setminus \mathcal{S}_T\|^2 + \|\mathbf{A} \odot \bar{\mathcal{S}}_{\mathcal{P}}\|^2 \\ &\overset{(5.18)}{<} \|(\mathbf{A} - \mathbf{X}^1_{\bar{T}}(\mathbf{Y}^1_{\bar{T}})^\top) \odot \mathcal{S}_{\mathcal{P}} \setminus \mathcal{S}_T\|^2 + \|\mathbf{A} \odot \bar{\mathcal{S}}_{\mathcal{P}}\|^2 \\ &= \|\mathbf{A} - \mathbf{X}\mathbf{Y}^\top\|^2. \end{aligned}$$

Lastly, we show that $(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$ can be chosen arbitrarily close to $(\mathbf{X}, \mathbf{Y})$ by choosing $(\tilde{\mathbf{X}}^1_{\bar{T}}, \tilde{\mathbf{Y}}^1_{\bar{T}})$ close enough to $(\mathbf{X}^1_{\bar{T}}, \mathbf{Y}^1_{\bar{T}})$. For this, denoting $\epsilon := \|\mathbf{X}^1_{\bar{T}} - \tilde{\mathbf{X}}\|^2 + \|\mathbf{Y}^1_{\bar{T}} - \tilde{\mathbf{Y}}\|^2$, we first compute:

$$\begin{aligned}\|\mathbf{X} - \tilde{\mathbf{X}}\|^2 + \|\mathbf{Y} - \tilde{\mathbf{Y}}\|^2 &= \|\mathbf{X}_T - \tilde{\mathbf{X}}_T\|^2 + \|\mathbf{Y}_T - \tilde{\mathbf{Y}}_T\|^2 + \|\mathbf{X}^1_{\bar{T}} - \tilde{\mathbf{X}}^1_{\bar{T}}\|^2 + \|\mathbf{Y}^1_{\bar{T}} - \tilde{\mathbf{Y}}^1_{\bar{T}}\|^2 \\ &\leq \mathcal{C}\|\mathbf{X}_T\mathbf{Y}_T^\top - \mathbf{B}\|^2 + \epsilon\end{aligned}$$

We will bound the value $\|\mathbf{X}_T\mathbf{Y}_T^\top - \mathbf{B}\|^2$. By using Equation (5.7), we have:

$$\begin{aligned}(\mathbf{A} - \sum_{1\leq i,j\leq 2}(\mathbf{X}^i_{\bar{T}})(\mathbf{Y}^j_{\bar{T}})^\top) \odot \mathcal{S}_T - \mathbf{X}_T\mathbf{Y}_T^\top &= (\mathbf{A} - \mathbf{X}_T\mathbf{Y}_T^\top - \sum_{1\leq i,j\leq 2}(\mathbf{X}^i_{\bar{T}})(\mathbf{Y}^j_{\bar{T}})^\top) \odot \mathcal{S}_T \\ &= (\mathbf{A} - \mathbf{X}\mathbf{Y}^\top) \odot \mathcal{S}_T \overset{(5.7)}{=} \mathbf{0}\end{aligned}$$

Therefore, $\mathbf{X}_T\mathbf{Y}_T^\top = [\mathbf{A} - (\mathbf{X}^1_{\bar{T}} + \mathbf{X}^2_{\bar{T}})(\mathbf{Y}^1_{\bar{T}} + \mathbf{Y}^2_{\bar{T}})^\top] \odot \mathcal{S}_T$. We have:

$$\begin{aligned}\|\mathbf{X}_T\mathbf{Y}_T^\top - \mathbf{B}\|^2 &= \|[\mathbf{A} - (\mathbf{X}^1_{\bar{T}} + \mathbf{X}^2_{\bar{T}})(\mathbf{Y}^1_{\bar{T}} + \mathbf{Y}^2_{\bar{T}})^\top] \odot \mathcal{S}_T - \mathbf{B}\|^2 \\ &= \|[(\tilde{\mathbf{X}}^1_{\bar{T}} + \mathbf{X}^2_{\bar{T}})(\tilde{\mathbf{Y}}^1_{\bar{T}} + \mathbf{Y}^2_{\bar{T}})^\top - (\mathbf{X}^1_{\bar{T}} + \mathbf{X}^2_{\bar{T}})(\mathbf{Y}^1_{\bar{T}} + \mathbf{Y}^2_{\bar{T}})^\top] \odot \mathcal{S}_T\|^2 \\ &\leq \|(\tilde{\mathbf{X}}^1_{\bar{T}} + \mathbf{X}^2_{\bar{T}})(\tilde{\mathbf{Y}}^1_{\bar{T}} + \mathbf{Y}^2_{\bar{T}})^\top - (\mathbf{X}^1_{\bar{T}} + \mathbf{X}^2_{\bar{T}})(\mathbf{Y}^1_{\bar{T}} + \mathbf{Y}^2_{\bar{T}})^\top\|^2\end{aligned}$$

When $\epsilon \to 0$, we have $\|(\tilde{\mathbf{X}}^1_{\bar{T}} + \mathbf{X}^2_{\bar{T}})(\tilde{\mathbf{Y}}^1_{\bar{T}} + \mathbf{Y}^2_{\bar{T}})^\top - (\mathbf{X}^1_{\bar{T}} + \mathbf{X}^2_{\bar{T}})(\mathbf{Y}^1_{\bar{T}} + \mathbf{Y}^2_{\bar{T}})^\top\| \to 0$. Therefore, with ϵ small enough, one have $\|\mathbf{X} - \mathbf{X}'\|^2 + \|\mathbf{Y} - \mathbf{Y}'\|^2$ can be arbitrarily small. This concludes the proof.

□

Conclusion

This chapter is the last one (directly) discussing about the properties of (2.4.FSMF) and it also concludes the second part of this thesis: the tractability and benign landscape of (2.4.FSMF) under additional assumptions on the support constraints (I, J) (such as those in Theorem 5.2.3 and Theorem 5.2.9). It illustrates a sharp contrast with what has been presented in Chapters 3 and 4: with arbitrary support constraints (I, J), neither the tractability nor the benign landscape are proved. Note that the conditions in Theorems 5.2.3 and 5.2.9 are only sufficient, i.e., if they are satisfied, then the corresponding instances can be solved efficiently. It is interesting to further expand this family of support constraints (I, J) beyond those of Theorem 5.2.3 and Theorem 5.2.9. Such an expansion, in our opinion, is likely to require us to:

1. Find a (more compact) mathematical description to decide whether a pair of support constraints (I, J) is well-posed/ill-posed or not since we need the existence of the optimal solutions guaranteed first, before having a (polynomial) algorithm to find them. The algorithmic response for this decision problem in Section 3.2 is still far from achieving our goal.

2. Go beyond the notion of block-wise low-rank matrix approximation, which plays a central role in this chapter.

Part III

Applications of Fixed support matrix factorization to sparse deep neural networks

We recapitulate the main results with regards to (2.4.FSMF) in Part I and Part II as follows:

1. **Non-existence of optimal solutions**: Certain inputs $(\mathbf{A}, I, J)$ are such that their corresponding (2.4.FSMF) instances does not admit optimizer (Section 3.1).

2. **NP-hardness**: Solving (2.4.FSMF) with arbitrary inputs $(\mathbf{A}, I, J)$ is NP-hard (Theorem 3.3.4).

3. **Tractability with structured support constraints**: If the support constraints (I, J) possess certain properties (Theorems 5.2.3 and 5.2.9), (2.4.FSMF) becomes tractable.

4. **Landscape**: In general, the landscape of $L(\mathbf{X}, \mathbf{Y})$ of (2.4.FSMF) has spurious local minima (Definition 4.1.1) and spurious local valleys (Definition 4.1.5). However, if the conditions in Theorems 5.2.3 and 5.2.9 are satisfied by (I, J), then the landscape is *benign* (Theorems 5.3.1, 5.3.2 and 5.3.8).

In the following chapters, we attempt to analyze these same questions but in the setting of (2.2.SNNT) with ReLU sparse DNNs. Below is an outline for Part III, the last technical part of this thesis:

1. In Chapter 6, using the results on the **Tractability with structured support constraints** of (2.4.FSMF), we study the so-called *butterfly parameterization*, a prominent approach when it comes to structured and sparse DNNs [DGE+19, DCS+22b, LRC+21, DSG+20]. We show that the use of the butterfly parameterization is equivalent to a change of parameter space that we can mathematically explicit. This result is obtained via our analysis of the problem of fixed support matrix factorization with multiple factors (more than two) having structured support constraints.

2. Developed from the results on the **Non-existence of optimal solutions** of (2.4.FSMF), the same question for sparse DNNs training is studied in Chapter 7. This chapter provides a classification of sparse DNN structures: those are such that their training problems always admit optimal solution and those are not. These results are established via an investigation of the topological properties of the function spaces corresponding to sparse DNN structures. This approach does not only allow us to study the existence of optimal solutions in the setting of sparse DNNs training but it also helps to establish several generalizations of existing results in the domain of functional analysis of deep learning.

Developing two aspects: **NP-hardness** and **Landscape** for sparse DNNs is also interesting, but this will be left for future works.

Chapter 6

Butterfly parameterization: theory and algorithm

Butterfly parameterization is a method representing linear operators as products of sparse matrices with butterfly structures (cf. Section 2.3.3). It provides us with a general method to accelerate the product matrix-vector, thus, finds various applications in scientific computing in general and sparse DNNs in particular. In this chapter, we present a study on the butterfly parameterization. More specifically, the main contributions in this chapter with regards to butterfly parameterization are: first, we introduce a compact definition for the so-called *deformable butterfly factors* [LRC+21], which unifies and extends all the existing butterfly parameterizations proposed in the literature. Secondly, under the notion of *chainability* (Definitions 6.3.7 and 6.3.10), our result in this chapter is the first work proving the approximation factor/ratio for the so-called *butterfly algorithm* [O'N07, LYM+15, LZRG22, ZRG23], a method to project a given matrix onto the set of matrices admitting a representation as a product of butterfly factors. Finally, as a corollary of the previous result, we show that, using butterfly structures in the context of sparse DNNs is equivalent to perform a change of parameter space: instead of optimizing the coefficients of a matrix (linear operator), one implicitly adds low-rank constraints on a family of submatrices of the trained linear operator. Materials in this chapter are taken from a working paper that is in preparation. This is a joint work with Léon Zheng[1].

6.1 Introduction

Algorithms for the rapid evaluation of linear operators are important tools in many domains like scientific computing, signal processing, and machine learning. In such applications where a very large number of parameters is involved, the *direct* computation of the matrix-vector multiplication hardly scales due to its quadratic complexity in the matrix size. Many existing works therefore rely on analytical or algebraic assumptions on the considered matrix to approximate the evaluation of matrix-vector multiplication with a subquadratic complexity. Examples of such structures include low-rank matrices, hierarchical matrices Definition 2.4.1, fast multipole methods [EMRV92], etc.

[1] https://leonzheng2.github.io/

Among these different structures, recent work has identified another class of matrices that can be compressed for accelerating matrix multiplication. It is the class of so-called *butterfly* matrices, and includes many matrices appearing in scientific computing problems, like kernel matrices associated to special function transforms or Fourier integral operators [O'N07]. Such matrices satisfy a certain low-rank property, named the *complementary low-rank* property: it has been shown that if specific submatrices of a target matrix $\mathbf{A}$ of size $n \times n$ are numerically low-rank, then $\mathbf{A}$ can be compressed by successive hierarchical low-rank approximations of these submatrices, in the sense that it can be approximated by a sparse factorization

$$\hat{\mathbf{A}} = \mathbf{X}_1 \dots \mathbf{X}_N \tag{6.1}$$

with $N = \mathcal{O}(\log n)$ factors $\mathbf{X}_{(\ell)}$ having at most $\mathcal{O}(n)$ nonzero entries for each $\ell = 1, \dots, N$. This sparse factorization, called in general *butterfly factorization*, would then yield a fast algorithm for the approximate evaluation of the matrix-vector multiplication by $\mathbf{A}$, in $\mathcal{O}(n \log n)$ complexity.

Butterfly parameterization in machine learning applications Inspired by these previous work on butterfly factorization, several recent works construct a generic matrix representation for structured matrices in machine learning applications. Indeed, their motivation is to propose a parameterization of structured linear maps that is *expressive* enough to capture commonly used structured linear maps, such as the Hadamard matrix, the DFT matrix, circulant matrices, Toeplitz matrices, or the Fastfood transform matrix, while being *differentiable* in order to use this parameterization in a machine learning pipeline where parameters of a model are optimized by gradient-based methods during the training phase in a learning task. The parameterization should also allow for an *efficient* implementation of the corresponding matrix multiplication.

Therefore, their representations take the form of a sparse factorization $\mathbf{A} = \mathbf{X}_1 \dots \mathbf{X}_N$ for a given number of factors $N \geq 2$, where specific sparsity patterns are enforced on the sparse factors $\mathbf{X}_1, \dots, \mathbf{X}_N$ in order to mimic the divide-and-conquer algorithm in the fast Fourier transform. In existing butterfly representations, each factor $\mathbf{X}_\ell$ for $\ell \in [\![N]\!]$ satisfies a so-called *fixed-support* constraint. The different existing butterfly representations only vary by their number of factors N, and their choice of the prescribed supports $\mathbf{S}_1, \dots, \mathbf{S}_N$. Let us illustrate existing butterfly parameterizations.

- **Square dyadic butterfly factorization** [DGE+19, DSG+20, LZRG22, ZRG23]. This representation is defined for matrices of size $n \times n$ where n is a power of two. The number of factors in this factorization is $N := \log_2 n$, and each factor is of size $n \times n$. In the square dyadic butterfly factorization $\mathbf{A} = \mathbf{X}_1 \dots \mathbf{X}_N$, the factor $\mathbf{X}_\ell$ for $\ell \in [\![N]\!]$ satisfies the support constraint $\operatorname{supp}(\mathbf{X}_\ell) \subseteq \operatorname{supp}(\mathbf{S}_\ell)$, where

  $$\forall \ell \in [\![N]\!], \quad \mathbf{S}_\ell := \mathbf{I}_{2^{\ell-1}} \otimes \mathbf{1}_{2\times 2} \otimes \mathbf{I}_{n/2^\ell}.$$

  This is the support constraints for the factors of the DFT, which are introduced in Section 2.3.3. This representation was originally used for designing structured random orthogonal matrices [Par95], and later used for quadrature rules on the hypersphere [MKBO18]. In recent machine learning applications, this parameterization has been used to replace hand-crafted structures in speech processing models

or channel shuffling in certain convolutional neural networks, or to learn latent permutation [DSG+20].

- **Monarch factorization** [DCS+22b]. A Monarch factorization parameterized by an integer b decomposes a matrix $\mathbf{A}$ of size $m \times n$ into $N := 2$ factors $\mathbf{X}_1$, $\mathbf{X}_2$ such that $\mathrm{supp}(\mathbf{X}_\ell) \subseteq \mathrm{supp}(\mathbf{S}_\ell)$ for $\ell = 1, 2$ where

$$\mathbf{S}_1 := \mathbf{1}_{p \times q} \otimes \mathbf{I}_{\frac{m}{p}}, \quad \mathbf{S}_2 := \mathbf{I}_q \otimes \mathbf{1}_{\frac{m}{p} \times \frac{n}{q}},$$

  where p, q are two integers satisfying $p|m, q|n$ (for two integers a and b, $a|b$ implies that a is a divisor of b). This parameterization is used for certain weight matrices in the transformer architecture for vision or language tasks [DCS+22b]. In comparison to square dyadic butterfly parameterization, monarch factorization can be used to represent rectangular matrices. The number of factors is, however, limited to only two.

- **Deformable butterfly factorization** [LRC+21]. Given an integer $N \geq 2$, a deformable butterfly factorization $\mathbf{A} = \mathbf{X}_1 \dots \mathbf{X}_N$ parameterized by a list of tuples $(p_\ell, q_\ell, r_\ell, s_\ell, t_\ell)_{\ell=1}^N$, each factor $\mathbf{X}_\ell$ for $\ell \in [\![N]\!]$ is of size $p_\ell \times q_\ell$ and has a support included in $\mathrm{supp}(\mathbf{S}_\ell)$ where

$$\forall \ell \in [\![N]\!], \quad \mathbf{S}_\ell := \mathbf{I}_{\frac{p_\ell}{r_\ell t_\ell}} \otimes \mathbf{1}_{r_\ell \times s_\ell} \otimes \mathbf{I}_{t_\ell}. \tag{6.2}$$

  Here it is assumed that $\frac{p_\ell}{r_\ell t_\ell} = \frac{q_\ell}{s_\ell t_\ell}$ is an integer, for each $\ell \in [\![N]\!]$. This parameterization can be used to replace kernel weights in convolutional layers in vision tasks, to obtain similar performance as the original convolutional neural network with fewer parameters.

In other words, parameterizing certain weight matrices in a neural network using a certain butterfly parameterization introduces a certain bias in the model, and if the bias is well-adapted to the given learning task for a given dataset, the obtained model could lead to good performance with respect to the original model, while having a reduced number of parameters, and eventually a reduced computational cost, provided that a fast algorithm associated to the obtained sparse factorization can be implemented efficiently.

In general, we remark in the previous examples that the fixed-support constraint on each butterfly factor $\mathbf{X}$ takes the form $\mathrm{supp}(\mathbf{X}) \subseteq \mathrm{supp}(\mathbf{I}_a \otimes \mathbf{1}_{b \times c} \otimes \mathbf{I}_d)$ for some parameters (a, b, c, d). Hence, we can define the set of so-called *deformable butterfly factors* associated to $\theta := (a, b, c, d)$ as

$$\mathcal{F}^\theta := \{\mathbf{X} \,|\, \mathrm{supp}(\mathbf{X}) \subseteq \mathrm{supp}(\mathbf{I}_a \otimes \mathbf{1}_{b \times c} \otimes \mathbf{I}_d)\}. \tag{6.3}$$

We re-use the name *deformable butterfly factors* because this parameterization is equivalent to that of [LRC+21]. While we owe [LRC+21] the original idea, the express of the support constraint using $\otimes$ is our novelty. Moreover, our parameterization uses 4 parameters, removing the redundancy in the description of deformable butterfly factors of [LRC+21] that uses 5 parameters.

Therefore, given a number of factors $N \geq 2$ and some parameters $\Theta := (\theta_\ell)_{\ell=1}^N$, we say that a matrix $\mathbf{A}$ admits a *deformable butterfly factorization* [LRC+21] associated to Θ if there exist butterfly factors $\mathbf{X}_1, \dots \mathbf{X}_N$ such that

$$\mathbf{A} = \mathbf{X}_1 \dots \mathbf{X}_N \quad \text{with} \quad (\mathbf{X}_\ell)_{\ell=1}^N \in \mathcal{F}^{\theta_1} \times \dots \times \mathcal{F}^{\theta_N}. \tag{6.4}$$

In the following, the class of matrices admitting such a factorization will be denoted

$$\mathcal{B}^\Theta := \left\{ \mathbf{X}_1 \dots \mathbf{X}_N \,|\, \mathbf{X}_\ell \in \mathcal{F}^{\theta_\ell},\, \ell \in [\![N]\!] \right\}. \tag{6.5}$$

This chapter focuses on the problem of approximating a target matrix $\mathbf{A}$ by a product of deformable butterfly factors associated to a given sequence of parameters $\Theta = (\theta_\ell)_{\ell=1}^N$:

$$\inf_{\mathbf{X}_\ell \in \mathcal{F}^{\theta_\ell}, \ell=1,\dots,N} \|\mathbf{A} - \mathbf{X}_1 \dots \mathbf{X}_N\|_F. \tag{6.6}$$

Remark 6.1.1. In general, it is necessary to take into account row and column permutations in the approximation problem for a greater flexibility. For instance, as seen in Section 2.3.3, the DFT matrix admits a square dyadic butterfly factorization *up to the bit-reversal permutation of column indices*. Therefore, as proposed in [ZPR+23], the general approximation problem that takes into account row and column permutations is:

$$\inf_{\mathbf{X}_\ell \in \mathcal{F}^{\theta_\ell}, \mathbf{P}, \mathbf{Q}} \|\mathbf{A} - \mathbf{Q}^\top \mathbf{X}_1 \dots \mathbf{X}_N \mathbf{P}\|_F,$$

where $\mathbf{P}$, $\mathbf{Q}$ are unknown permutations part of the optimization problem. Without any further assumption on the target matrix $\mathbf{A}$, solving this approximation problem is conjectured to be difficult, even though a heuristic based on spectral clustering has been proposed in [ZPR+23]. Thus, for the rest of the chapter, WLOG, we assume that the permutation matrices $\mathbf{P}$ and $\mathbf{Q}$ are identity matrices, which leads to problem (6.6).

The main motivation for studying problem (6.6) is to construct a fast algorithm for approximate matrix multiplication by $\mathbf{A}$, assuming that the choice of Θ leads to a matrix factorization with sparse factors. One of the main benefit of choosing fixed-support constraints of the form $\mathrm{supp}(\mathbf{X}) \subseteq \mathrm{supp}(\mathbf{I}_a \otimes \mathbf{1}_{b\times c} \otimes \mathbf{I}_d)$ for the butterfly factors $\mathbf{X}$ is its *block structure* that could enable efficient implementation on specific hardwares like Intelligence Processing Unit (IPU) [SAF+23] or GPU [DGE+19, DCS+22b], with practical speed-up for matrix multiplication. A detail discussion of these efficient implementations is, however, out of the scope of this thesis.

Main questions This general formulation of the butterfly factorization problem associated to a sequence of parameters $\Theta := (\theta_\ell)_{\ell=1}^N$ raises the following questions that we want to address in this chapter.

1. What are the parameters Θ for which there exists also a hierarchical algorithm that can compute an approximate solution to (6.6)?

2. For such Θ, what kind of guarantees on the control of approximation error can we prove?

3. For such Θ, what are the matrices $\mathbf{A}$ that can be reasonably well approximated by a matrix $\hat{\mathbf{A}} \in \mathcal{B}^{\Theta}$? In other words, what are the matrices $\mathbf{A}$ for which $\inf_{\hat{\mathbf{A}} \in \mathcal{B}^{\Theta}} \|\mathbf{A} - \hat{\mathbf{A}}\|_F$ is small with respect to $\|\mathbf{A}\|_F$? In particular, how can we characterize the set $\mathcal{B}^{\Theta}$?

Answering those questions leads to the main contributions of this chapter, which are:

Contributions

1. We define the notion of *chainability* (cf. Definition 6.3.7 and Definition 6.3.10). This is a sufficient condition for which the problem (6.6) admits a hierarchical factorization algorithm (cf. Algorithm 8). More importantly, all existing butterfly parameterizations in this section satisfy our condition of chainability.

2. If Θ satisfies our notion of chainability, then our proposed hierarchical algorithm (cf. Algorithm 8) is an *approximate algorithm* in the following sense: it will output feasible $\mathbf{X}_\ell, \ell = 1, \ldots, N$ such that:

$$\|\mathbf{A} - \mathbf{X}_1 \ldots \mathbf{X}_N\|_F \leq C_N E^{\Theta}(\mathbf{A})$$

   where $E^{\Theta}(\mathbf{A})$ is the optimal value[2] of (6.6) and C_N is a constant depending on N. We have two results related to this contribution, which are Theorem 6.5.9 and Theorem 6.5.14. We emphasize that ours are different in nature from another error bound in the literature [LYM+15], which is:

$$\|\mathbf{A} - \mathbf{X}_1 \ldots \mathbf{X}_N\|_F^2 \leq C_n \epsilon_0^2 \|\mathbf{A}\|_F^2, \quad \text{with } C_n = \mathcal{O}(\log n), \tag{6.7}$$

   which does not provide any comparison between the obtained factorization and the best possible approximation.

3. As a corollary of the previous contribution, we provide an analytic characterization for the set $\mathcal{B}^{\Theta}$. Different from the "synthesis" definition (cf. Equation (6.5)), which does not provide an explicit algorithm to verify if a matrix $\mathbf{A} \in \mathcal{B}^{\Theta}$, our characterization state that the verification of $\mathbf{A} \in \mathcal{B}^{\Theta}$ is bound to compute the ranks of a polynomial number of submatrices of $\mathbf{A}$. This result is presented in Corollary 6.5.12.

The rest of this chapter is organized as follows: In Section 6.2, we recall the problem of fixed support matrix factorization, with a slight change in term of notations. In Section 6.3, we formally introduce the deformable butterfly factors and the notion of chainability. Section 6.4 and Section 6.5 are devoted to describe the proposed hierarchical algorithm for (6.6). The error bounds and their corollaries are presented in Section 6.5.2. Some experiments on the proposed algorithm is conducted and discussed in Section 6.6. In the last section, we detail the proofs for results in previous sections.

[2] $E^{\Theta}(\mathbf{A})$ reads as approximation error between the matrix $\mathbf{A}$ and the set $\mathcal{B}^{\Theta}$.

6.2 Remind on two-factors fixed support matrix factorization

As discussed, the tools developed in Chapter 5 will be used for the analysis of this chapter. Therefore, we start the technical parts of this chapter by reminding our beloved fixed support matrix factorization with two factors. It is formulated as:

$$\underset{(\mathbf{X},\mathbf{Y})}{\text{Minimize}} \quad \|\mathbf{Z}-\mathbf{XY}\|_F^2, \text{ with } \operatorname{supp}(\mathbf{X}) \subseteq \mathbf{L}, \operatorname{supp}(\mathbf{Y}) \subseteq \mathbf{R}. \tag{6.1.FSMF}$$

Note that (6.1.FSMF) is basically (2.4.FSMF) (but the product is $\mathbf{XY}$, instead of $\mathbf{XY}^\top$). We chose the formulation of (2.4.FSMF) in the previous chapters since it reduced significantly the notations to *prove* the results. In this chapter, we are going to deal with multiple factors, and we only need to use the results in Chapter 5. Therefore, it will be more convenient to use (6.1.FSMF). Several definitions in Chapter 5 has to be adapted to be consistent with (6.1.FSMF). They are given below:

Definition 6.2.1 (Rank-one contribution supports - Definition 5.2.1)**.** The *rank-one contribution supports* of two binary matrices $\mathbf{L} \in \{0,1\}^{m\times r}, \mathbf{R} \in \{0,1\}^{r\times n}$ is the tuple of r binary matrices defined by:

$$\varphi(\mathbf{L},\mathbf{R}) := (\mathbf{U}_i)_{i=1}^r, \quad \text{where} \quad \mathbf{U}_i := \mathbf{L}[:,i]\mathbf{R}[i,:]. \tag{6.8}$$

Definition 6.2.2 (Equivalence classes of rank-one supports, representative rank-one supports - Definition 5.2.2)**.** Given two binary matrices $\mathbf{L} \in \{0,1\}^{m\times r}, \mathbf{R} \in \{0,1\}^{r\times n}$, denoting $(\mathbf{U}_i)_{i=1}^r := \varphi(\mathbf{L},\mathbf{R})$, define an equivalence relation on $[\![r]\!]$ as: $i \sim j \iff \mathbf{U}_i = \mathbf{U}_j$. This yields a partition of $[\![r]\!]$ into equivalence classes. The partition will be denoted $\mathcal{P}(\mathbf{L},\mathbf{R})$.

For each equivalence class $P \in \mathcal{P}(\mathbf{L},\mathbf{R})$, denote $\mathbf{U}_P$ a representative rank-one support, $R_P \subseteq [m]$ and $C_P \subseteq [n]$ the supports of rows and columns in $\mathbf{U}_P$, respectively, and denote $|P|$ the cardinal of the equivalence class P.

Theorem 6.2.3 (Tractable support constraints of (6.1.FSMF) - extended version of Theorem 5.2.3)**.** *If all elements of $\varphi(\mathbf{L},\mathbf{R})$ are pairwise disjoint or identical, then Algorithm 5 yields an optimal solution of (6.1.FSMF) in polynomial time. Denoting $\sigma_j(\cdot)$ the j-th largest singular value of a matrix, we have:*

$$\inf_{\operatorname{supp}(\mathbf{X})\subseteq\mathbf{L},\operatorname{supp}(\mathbf{Y})\subseteq\mathbf{R}} \|\mathbf{A}-\mathbf{XY}\|_F^2 = \|\mathbf{A}\|_F^2 - \sum_{P\in\mathcal{P}(\mathbf{L},\mathbf{R})} \sum_{j=1}^{|P|} \sigma_j^2(\mathbf{A}[R_P,C_P]).$$

In comparison to Theorem 5.2.3, Theorem 6.2.3 provides in addition an explicit formula for the minimum value. A complete proof of Theorem 6.2.3, which is developed based on the proof of Theorem 5.2.3, can be found in Section 6.7.1.

6.3 Deformable butterfly factorization

This section formally introduces a concise mathematical formulation of the deformable butterfly [LRC+21], a definition that encompasses other existing butterfly factorizations such

Algorithm 5 Two-factor fixed support matrix factorization (adapted from Algorithm 3)

Require: $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\mathbf{L} \in \{0,1\}^{m\times r}$ and $\mathbf{R} \in \{0,1\}^{r\times m}$.

$\mathbf{X} \leftarrow \mathbf{0}$.
$\mathbf{Y} \leftarrow \mathbf{0}$.
for $P \in \mathcal{P}(\mathbf{L}, \mathbf{R})$ **do**
    $(\mathbf{X}[R_P, P], \mathbf{Y}[P, C_P]) \leftarrow (\mathbf{H}, \mathbf{K}) \in \underset{\mathbf{H}\in\mathbb{R}^{|R_P|\times|P|}, \mathbf{K}\in\mathbb{R}^{|P|\times|C_P|}}{\arg\min} \|\mathbf{A}[R_P, C_P] - \mathbf{HK}\|_F$
    $\mathbf{A} \leftarrow \mathbf{A} - \mathbf{X}[:, P]\mathbf{Y}[P, :]$
end for
return $(\mathbf{X}, \mathbf{Y})$

as square dyadic butterfly factorization [DGE+19, DSG+20, LZRG22, ZRG23], Monarch factorization [DCS+22b] that we have seen in the introduction. Then, we propose the notion of *chainability* (cf. Definitions 6.3.7 and 6.3.10), which will be shown to be a sufficient condition allowing (6.6) to admit an approximate algorithm.

6.3.1 A mathematical formulation for deformable butterfly factor

According to [DGE+19, DSG+20, LZRG22, ZRG23, DCS+22b, LRC+21], a matrix $\mathbf{A}$ admits a certain butterfly factorization if it can be factorized into a certain number of factors $\mathbf{X}_1, \ldots, \mathbf{X}_N$ for a prescribed number $N \geq 2$, such that each factor $\mathbf{X}_\ell$ for $\ell \in [\![N]\!]$ satisfies a so-called *fixed-support* constraint, i.e., the support of $\mathbf{X}_\ell$ is included in a prescribed support $\mathbf{S}_\ell$. Existing butterfly factorizations only vary by their number of factors N, and their choice of the prescribed supports $\mathbf{S}_1, \ldots, \mathbf{S}_N$. However, as it will be explained below, we remark that the fixed-support constraint on each butterfly factor $\mathbf{X}$ is $\mathrm{supp}(\mathbf{X}) \subseteq \mathbf{I}_a \otimes \mathbf{1}_{b\times c} \otimes \mathbf{I}_d$ for some parameters (a, b, c, d). We therefore introduce the following definition.

Definition 6.3.1 (Deformable butterfly factor)**.** For $a, b, c, d \in \mathbb{N}$, a *deformable butterfly factor* (DB factor) of parameters $\theta := (a, b, c, d)$ (or (a, b, c, d)-DB factor) is a matrix for which:

1. The matrix size is $m \times n$ where $m := abd, n := acd$.
2. The matrix support is included in $\mathbf{S}_\theta := \mathbf{I}_a \otimes \mathbf{1}_{b\times c} \otimes \mathbf{I}_d$.

The tuple θ will be called a *deformable butterfly parameter* (DB parameter). The set of all θ-DB factors is denoted by $\mathcal{F}^\theta$.

Remark 6.3.2. Definition 6.3.6 formalizes the description of the supports of a deformable butterfly factor [LRC+21]. Indeed, the original paper [LRC+21] did not formalize mathematically the description of these supports as a Kronecker product, which is the purpose of Definition 6.3.6 here. This is one of our novelties and it allows us to manipulate the operations involved with DB factors much easier. Moreover, in the original paper [LRC+21], the support constraint on the butterfly factors is parameterized with 5 parameters, with 1 redundant parameter, while Definition 6.3.6 only uses 4 parameters.

Figure 6.1 illustrates the support of a θ-DB factor, for a given DB parameter $\theta = (a, b, c, d)$. A θ-DB factor matrix is block diagonal with a blocks in total. By definition, each block in the diagonal has its support included in $\mathbf{1}_{b\times c} \otimes \mathbf{I}_d$. Thus, each block is a *block matrix* of size $b \times c$, where each *sub-block* is a diagonal matrix of size $d \times d$.

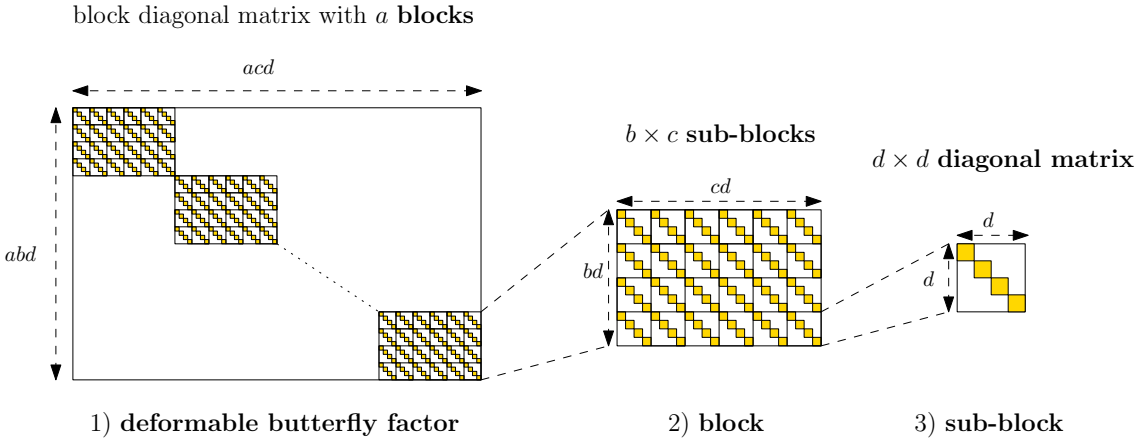


Figure 6.1: Illustration of the support of a deformable butterfly factor of parameters (a, b, c, d). The yellow squares indicate the indices belonging to the support. The sub-figures 1), 2), 3) illustrate the concepts of factor, block and sub-block respectively.

Example 6.3.3. The following matrices are θ-DB factor for certain choices of θ.

1. Dense matrix: Any matrix of size $m \times n$ is a DB factor of parameter $(1, m, n, 1)$.
2. Diagonal matrix: Any diagonal matrix of size $m \times m$ is either a DB factor of parameter $(m, 1, 1, 1)$ or $(1, 1, 1, m)$.
3. Square dyadic butterfly factorization [DGE+19, DSG+20, LZRG22, ZRG23]: for $\ell \in [\![N]\!]$, $\theta_\ell = (2^{\ell-1}, 2, 2, 2^{N-\ell})$
4. Monarch factorization [DCS+22b]: $\theta_1 = (1, p, q, m/p)$, $\theta_2 = (q, m/p, n/q, 1)$ for some integers $p|m, q|n$.

Lemma 6.3.4 (Number of nonzero entries of DB factor)**.** *For $\theta = (a, b, c, d)$, the number of nonzero entries of a θ-DB factor $\mathbf{A}$ of size $m \times n$ is at most $\|\theta\|_0 := abcd = mc = nb$.*

Proof. The number of indices in the support constraints $\mathbf{I}_a \otimes \mathbf{1}_{b\times c} \otimes \mathbf{I}_d$ is $abcd = mc = nb = \frac{mn}{ad}$. □

Therefore, if $ad \gg O(1)$ one can see DB factor is inherently sparse. Given a number of factors $N \geq 2$, a sequence of DB parameters $\Theta := (\theta_\ell)_{\ell=1}^N$ parameterizes the set

$$\mathcal{F}^\Theta := \mathcal{F}^{\theta_1} \times \ldots \times \mathcal{F}^{\theta_N}$$

of N-tuples of DB-factors, where $\times$ denotes the Cartesian product. We denote $|\Theta| := N$ the number of factors.

Definition 6.3.5. Given a sequence of DB parameters $\Theta = (\theta_\ell)_{\ell=1}^N$, we say that a sequence of matrices $(\mathbf{X}_\ell)_{\ell=1}^N$ is *associated* to Θ if $(\mathbf{X}_\ell)_{\ell=1}^N \in \mathcal{F}^\Theta$, i.e., $\mathbf{X}_\ell$ is a θ_ℓ-DB factor for each $\ell \in [\![N]\!]$.

Since we are interested in matrix products $\mathbf{X}_1 \dots \mathbf{X}_N$ for $(\mathbf{X}_\ell)_{\ell=1}^N \in \mathcal{F}^\Theta$, we will only consider Θ such that the size of $\mathbf{X}_\ell \in \mathcal{F}^{\theta_\ell}$ and $\mathbf{X}_{\ell+1} \in \mathcal{F}^{\theta_{\ell+1}}$ are compatible for computing the matrix product $\mathbf{X}_\ell \mathbf{X}_{\ell+1}$, for each $\ell \in [\![N-1]\!]$. In other words, we require that the sequence of DB parameters Θ satisfies:

$$\forall \ell \in [\![N-1]\!], \quad \underbrace{a_\ell c_\ell d_\ell}_{n_\ell} = \underbrace{a_{\ell+1} b_{\ell+1} d_{\ell+1}}_{m_{\ell+1}}. \tag{6.9}$$

Therefore, under assumption (6.9), a sequence Θ can describe a factorization of the type $\mathbf{A} = \mathbf{X}_1 \dots \mathbf{X}_N$ such that $(\mathbf{X}_\ell)_{\ell=1}^N \in \mathcal{F}^\Theta$. In general, for such a sequence Θ, we can define the problem of approximating a given target matrix $\mathbf{A}$ by a product of DB-factors $(\mathbf{X}_\ell)_{\ell=1}^N \in \mathcal{F}^\Theta$ as the minimization problem:

$$E^\Theta(\mathbf{A}) := \inf_{(\mathbf{X}_\ell)_{\ell=1}^N \in \mathcal{F}^\Theta} \|\mathbf{A} - \mathbf{X}_1 \dots \mathbf{X}_N\|_F^2, \tag{6.10}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. We call (6.10) the *deformable butterfly factorization* problem. This problem has another interesting viewpoint: (6.10) is equivalent to finding a projection of a matrix $\mathbf{A}$ onto the set $\mathcal{B}^\Theta$, which is defined as:

Definition 6.3.6 (Deformable butterfly matrix)**.** Given a sequence of DB parameters $\Theta = (\theta_\ell)_{\ell=1}^N$, the set of matrices that admit an exact factorization into θ_ℓ-DB factors for $\ell \in [\![N]\!]$ is called the set of *deformable butterfly matrices* associated to Θ, and is defined as:

$$\mathcal{B}^\Theta := \left\{ \mathbf{X}_1 \dots \mathbf{X}_N \,|\, \mathbf{X}_\ell \in \mathcal{F}^\theta,\ \ell \in [\![N]\!] \right\}.$$

Indeed, it is easy to see that (6.10) is equivalent to solve:

$$\inf_{\mathbf{B} \in \mathcal{B}^\Theta} \|\mathbf{A} - \mathbf{B}\|_F^2. \tag{6.11}$$

Thus, $E^\Theta(\mathbf{A})$ is the distance between the target matrix $\mathbf{A}$ and the set $\mathcal{B}^\Theta$. The two formulations (6.10) and (6.11) are complementary: from an optimization point of view, (6.10) is more favorable since it usually has fewer parameters and the optimization problem has no constraint. However, if one would like to answer questions such as: given a sequence Θ, which kinds of matrices that can be *reasonably approximated* by the product of $(\mathbf{X}_\ell)_{\ell=1}^N \in \mathcal{F}^\Theta$, then (6.11) is possibly more natural.

As shown in Section 3.3, the problem of approximating a given matrix by the product of *fixed support* factors (such as DB factors) is generally NP-hard and might lead to numerical instability even for $N = 2$. Nevertheless, for the choice of Θ corresponding to a square dyadic butterfly factorization, i.e., $\Theta = (2^{\ell-1}, 2, 2, 2^{N-\ell})_{\ell=1}^N$, there exists an efficient hierarchical algorithm that gives a "good" solution to problem (6.10), endowed with exact recovery guarantees [LZRG22, ZRG23], i.e., if $\mathbf{A} \in \mathcal{B}^\Theta$, the hierarchical algorithm of [LZRG22] will return butterfly factors $\mathbf{X}_\ell, \ell = 1, \dots, |\Theta|$ such that $\mathbf{A} = \prod_{\ell=1}^{|\Theta|} \mathbf{X}_\ell$. However, it is not known in the literature if this hierarchical algorithm can be extended to other choices of Θ and have theoretical guarantee even for $\mathbf{A} \notin \mathcal{B}^\Theta$.

6.3.2 Chainability

One main contribution of this chapter is to exhibit a (sufficient) general condition on the sequence of DB parameters Θ for which we can also propose a hierarchical algorithm to solve its corresponding deformable butterfly factorization problem. As a novelty, we will prove that the proposed hierarchical algorithm is also endowed with guarantees on the control of approximation error, in the sense that it yields factors $(\mathbf{X}_\ell)_{\ell=1}^N \in \mathcal{F}^\Theta$ satisfying:

$$\|\mathbf{A} - \mathbf{X}_1 \dots \mathbf{X}_N\|_F \leq C_N E^\Theta(\mathbf{A}), \tag{6.12}$$

where C_N is a constant called *approximate ratio* depending on $N = |\Theta|$. To explain the proposed general condition on the sequence Θ, let us start with the case where we consider a sequence of length $N = 2$. In this case, our condition on $\Theta = (\theta_1, \theta_2)$ is the so-called *chainability* condition, defined as follows.

Definition 6.3.7 (q-chainability and chainability)**.** Two DB parameters $\theta_1 := (a_1, b_1, c_1, d_1)$ and $\theta_2 := (a_2, b_2, c_2, d_2)$ are *q-chainable* if:

1. $\frac{a_1 c_1}{a_2} = \frac{b_2 d_2}{d_1} = q \in \mathbb{N}$.
2. $a_1 | a_2$.
3. $d_2 | d_1$.

Two DB parameters θ_1 and θ_2 are *chainable* if there exists a natural number $q \in \mathbb{N}$ such that they are q-chainable.

Remark 6.3.8. The order (θ_1, θ_2) in the definition matters, because this property is not symmetric: the chainability of (θ_1, θ_2) does not imply that of (θ_2, θ_1).

Remark 6.3.9. In fact, $a_1 c_1 d_1$ and $a_2 b_2 d_2$ are the numbers of columns of the first DB factor and rows of the second DB factor respectively. Therefore, the equality $a_1 c_1 / a_2 = b_2 d_2 / b_1$ in the first condition assures that one can multiply a θ_1 DB factor and a θ_2 DB factor. In other words, if θ_1 and θ_2 are chainable, then they satisfy the condition (6.9).

The extension of the notion of chainability to the case of sequence length $N \geq 2$ is as follows.

Definition 6.3.10 ($(q_1, \dots, q_{N-1})$-chainable sequence of DB parameters)**.** A sequence of DB parameters $\Theta := (\theta_\ell)_{\ell=1}^N$ is *$(q_1, \dots, q_{N-1})$-chainable* if θ_ℓ and $\theta_{\ell+1}$ are q_ℓ-chainable for each $\ell \in [\![N-1]\!]$, in the sense of Definition 6.3.7.

A sequence of DB parameters $\Theta := (\theta_\ell)_{\ell=1}^N$ is *chainable* if there exist $(q_1, \dots, q_{N-1})$, $q_\ell \in \mathbb{N}, \forall \ell \in [\![N-1]\!]$ such that Θ is $(q_1, \dots, q_{N-1})$-chainable.

The rest of this work will show that if Θ is chainable, then the deformable butterfly factorization problem admits an approximate algorithm whose approximate ratio/factor only depends on $|\Theta|$. We remark that the chainability in the sense of Definition 6.3.10 is not rare since it covers most of the proposals in the literature, as seen in Table 6.1.

Section 6.4 and Section 6.5.2 will explain why the condition of chainability in the sense of Definition 6.3.10 leads to an efficient hierarchical algorithm for problem (6.10). For the remainder of this section, we will give some intuition on the chainability notion and derive some useful properties that will be used for the rest of the chapter.

Table 6.1: Existing DB parameters correspond to different variants of the proposed general parameterization. (†) Note that the definition of chainability in [LRC+21] is stricter than Definition 6.3.7: all consecutive pairs of DB-parameters have to be 1-chainable.

Parameterization	Size	$\lvert\Theta\rvert$	DBP	Chainable
Low rank matrix	$m \times n$	2	$(1, m, r, 1)$ $(1, r, n, 1)$	Yes
Square dyadic butterfly [DGE+19]	$2^N \times 2^N$	N	$(2^i, 2, 2, 2^{N-i-1})$	Yes
Monarch [DCS+22b]	$m \times n$	2	$(1, p, q, m/p)$ $(q, m/p, n/q, 1)$	Yes
Deformable butterfly$^{(\dagger)}$ [LRC+21]	$m \times n$	arbitrary	(a_i, b_i, c_i, d_i)	Yes
Kaleidoscope [DSG+20]	$2^N \times 2^N$	$2N$	$(2^i, 2, 2, 2^{N-i-1})$	No

6.3.3 Properties of chainable deformable butterfly parameters

The main property of chainability is the following one.

Proposition 6.3.11. *If (θ_1, θ_2) are q-chainable, then:*

$$\mathbf{S}_{\theta_1}\mathbf{S}_{\theta_2} = q\mathbf{S}_{\theta}, \tag{6.13}$$

where $\theta = (a_2, \frac{b_1 d_1}{d_2}, \frac{a_2 c_2}{a_1}, d_2)$ is a DB parameter ($\mathbf{S}_\theta$ is defined as in Definition 6.3.1).

Remark 6.3.12. The equality (6.13) was proved in [ZRG23, Lemma 3.4] for the choice $\theta_1 = (2^{\ell-1}, 2, 2, 2^{N-\ell})$ and $\theta_2 = (2^{\ell}, 2, 2, 2^{N-\ell-1})$, for any integer $N \geq 2$ and $\ell \in [\![N-1]\!]$. Proposition 6.3.11 extends (6.13) to *all* chainable pair (θ_1, θ_2) of DB parameters.

This proposition gives an intuition for the notion of chainability. For any pair of DB parameters (θ_1, θ_2), the support $\mathrm{supp}(\mathbf{S}_{\theta_1}\mathbf{S}_{\theta_2})$ is the *maximum* support of the set $\mathcal{B}^{(\theta_1,\theta_2)} := \{\mathbf{X}_1\mathbf{X}_2 \,|\, \mathbf{X}_1 \in \mathcal{F}^{\theta_1}, \mathbf{X}_2 \in \mathcal{F}^{\theta_2}\}$, in the sense that:

$$\begin{aligned} &\forall \mathbf{X}_1 \in \mathcal{F}^{\theta_1}, \mathbf{X}_2 \in \mathcal{F}^{\theta_2}, \mathrm{supp}(\mathbf{X}_1\mathbf{X}_2) \subseteq \mathrm{supp}(\mathbf{S}_{\theta_1}\mathbf{S}_{\theta_2}), \\ \text{and} \quad &\exists \mathbf{X}_1 \in \mathcal{F}^{\theta_1}, \mathbf{X}_2 \in \mathcal{F}^{\theta_2}, \mathrm{supp}(\mathbf{X}_1\mathbf{X}_2) = \mathrm{supp}(\mathbf{S}_{\theta_1}\mathbf{S}_{\theta_2}). \end{aligned}$$

Proof. Due to the conditions of Definition 6.3.7, we can assume that $a_2 = ra_1, d_1 = sd_2$ for two integers r, s (since $a_1 | a_2, d_2 | d_1$). This yields $c_1 = a_2 q / a_1 = qr, b_2 = d_1 q / d_2 = qs$ (since $a_1 c_1 / a_2 = b_2 d_2 / d_1 = q$). Thus,

$$\begin{aligned} &\mathbf{S}_{(a_1,b_1,c_1,d_1)}\mathbf{S}_{(a_2,b_2,c_2,d_2)} \\ &= (\mathbf{I}_{a_1} \otimes \mathbf{1}_{b_1 \times qr} \otimes \mathbf{I}_{d_1}) (\mathbf{I}_{a_2} \otimes \mathbf{1}_{qs \times c_2} \otimes \mathbf{I}_{d_2}) \\ &= (\mathbf{I}_{a_1} \otimes \mathbf{1}_{b_1 \times r} \otimes \mathbf{1}_{1 \times q} \otimes \mathbf{I}_{d_1}) (\mathbf{I}_{a_2} \otimes \mathbf{1}_{q \times 1} \otimes \mathbf{1}_{s \times c_2} \otimes \mathbf{I}_{d_2}) \\ &= [(\mathbf{I}_{a_1} \otimes \mathbf{1}_{b_1 \times r}) \otimes \mathbf{1}_{1 \times q} \otimes \mathbf{I}_{d_1}] [\mathbf{I}_{a_2} \otimes \mathbf{1}_{q \times 1} \otimes (\mathbf{1}_{s \times c_2} \otimes \mathbf{I}_{d_2})] \\ &\overset{(\star)}{=} (\mathbf{I}_{a_1} \otimes \mathbf{1}_{b_1 \times r}) \otimes (\mathbf{1}_{1 \times q}\mathbf{1}_{q \times 1}) \otimes (\mathbf{1}_{s \times c_2} \otimes \mathbf{I}_{d_2}) \\ &= (\mathbf{I}_{a_1} \otimes \mathbf{1}_{b_1 \times r}) \otimes (q\mathbf{1}_{1 \times 1}) \otimes (\mathbf{1}_{s \times c_2} \otimes \mathbf{I}_{d_2}) \\ &= q (\mathbf{I}_{a_1} \otimes \mathbf{1}_{b_1 s \times r c_2} \otimes \mathbf{I}_{d_2}) \\ &= q\mathbf{S}_{\theta} \quad \left(\text{because } \frac{b_1 d_1}{d_2} = b_1 s, \frac{a_2 c_2}{c_1} = r c_2\right). \end{aligned} \tag{6.14}$$

We can use the equality $(\mathbf{A}\otimes\mathbf{C}\otimes\mathbf{E})(\mathbf{B}\otimes\mathbf{D}\otimes\mathbf{F}) = (\mathbf{AB})\otimes(\mathbf{CD})\otimes(\mathbf{EF})$ in $(\star)$ because, according to our conditions for chainability, the sizes of $\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D},\mathbf{E},\mathbf{F}$ in $(\star)$ make the matrix products $\mathbf{AB},\mathbf{CD}$ and $\mathbf{EF}$ well-defined. □

In other words, Proposition 6.3.11 says that if (θ_1,θ_2) is chainable, then the maximum support of $\mathcal{B}^{(\theta_1,\theta_2)}$ is the DB support $\mathbf{S}_\theta$ for a certain DB parameter θ.

Therefore, we can define an operator $*$ on the space of DB parameters.

Definition 6.3.13 (Operator on DB parameters)**.** For two chainable DB parameters $\theta_1 := (a_1,b_1,c_1,d_1)$ and $\theta_2 := (a_2,b_2,c_2,d_2)$, we define:

$$\theta_1 * \theta_2 := \left(a_1, \frac{b_1 d_1}{d_2}, \frac{a_2 c_2}{a_1}, d_2\right).$$

Hence, as a corollary of Proposition 6.3.11, if (θ_1,θ_2) is chainable, then the product of a θ_1-DB factor by a θ_2-DB factor is a θ-DB factor, for $\theta := \theta_1 * \theta_2$. Note that $\theta \in \mathbb{N}^4$ since $d_2|d_1$ and $a_1|a_2$.

Corollary 6.3.14. *If (θ_1,θ_2) are chainable DB parameters, then:*

$$\forall(\mathbf{X}_1,\mathbf{X}_2) \in \mathcal{F}^{\theta_1}\times\mathcal{F}^{\theta_2}, \quad \mathbf{X}_1\mathbf{X}_2 \in \mathcal{F}^{(\theta_1*\theta_2)}.$$

We also prove that the operator * in the space of DB parameters is associative.

Lemma 6.3.15 (Associativity of DBPs operator)**.** *If (θ_1,θ_2) and (θ_2,θ_3) are q_1-chainable and q_2-chainable respectively, then $(\theta_1,\theta_2*\theta_3)$ and $(\theta_1*\theta_2,\theta_3)$ are q_2-chainable and q_1-chainable respectively. Moreover, $\theta_1*(\theta_2*\theta_3) = (\theta_1*\theta_2)*\theta_3$.*

The proof of Lemma 6.3.15 is deferred to Section 6.7.2. Using the associativity of the operator $*$ and Corollary 6.3.14, we can compute the support of the product of a sequence of matrices $(\mathbf{X}_\ell)_{\ell=1}^L$ associated to a chainable Θ.

Lemma 6.3.16. *For $(\theta_\ell)_{\ell=1}^N$ a chainable sequence of DB parameters:*

$$\forall(\mathbf{X}_\ell)_{\ell=1}^N \in \mathcal{F}^{\theta_1}\times\ldots\times\mathcal{F}^{\theta_N}, \quad \mathbf{X}_1\ldots\mathbf{X}_N \in \mathcal{F}^{(\theta_1*\ldots*\theta_N)}. \tag{6.15}$$

Proof. The result follows from an induction on $|\Theta|$.

□

Remark 6.3.17. A direct calculation (cf. Section 6.7.3) will give us:

$$\theta_1*\ldots*\theta_N = \left(a_1, \frac{b_1 d_1}{d_N}, \frac{a_N c_N}{a_1}, d_N\right). \tag{6.16}$$

for each chainable sequence of DB parameters $\Theta = (\theta_\ell)_{\ell=1}^N$. Consequently, if $a_1 = d_N = 1$, then $\theta_1*\ldots*\theta_N = (a_1,m,n,d_N) = (1,m,n,1)$ for some integers m,n, which means that the product $\mathbf{X}_1\ldots\mathbf{X}_L$ for a sequence of matrices $(\mathbf{X}_\ell)_{\ell=1}^L$ associated to Θ can be a dense matrix. Vice versa, if one desires the product of butterfly factors to be dense, it is necessary to set $a_1 = d_N = 1$. In fact, all chainable sequences of DB parameters in Table 6.1 have $a_1 = d_N = 1$.

6.3.4 Non-redundant chainable sequence of DB parameters

Given a chain Θ, an important associated quantity is its total number of parameters, defined as follows.

Definition 6.3.18 (Number of parameters in a sequence of DB parameters)**.** The number of parameters in a sequence of DBPs $\Theta := (\theta_\ell)_{\ell=1}^N$ is defined as

$$\|\Theta\|_0 = \sum_{\ell=1}^{N} \|\theta_\ell\|_0, \tag{6.17}$$

where $\|\theta\|_0$ is defined as in Lemma 6.3.4.

In fact, this quantity is equal to the complexity of calculating $(\mathbf{X}_1 \dots \mathbf{X}_N)v$ for a sequence $(\mathbf{X}_\ell)_{\ell=1}^N$ associated to Θ and a vector v. Therefore, if there exist two different chainable sequences Θ_1 and Θ_2 of DB parameters such that $\mathcal{B}^{\Theta_1} = \mathcal{B}^{\Theta_2}$, it is preferable to work with those whose $\|\cdot\|_0$ (cf. Definition 6.3.18) is smaller. Indeed, if $\mathcal{B}^{\Theta_1} = \mathcal{B}^{\Theta_2}$, then their corresponding optimization problems (6.10) and (6.11) yield the same infimum. Thus, working with one having fewer parameters results into a better complexity for matrix-vector multiplication (after one found the deformable butterfly factors approximating "well" a given matrix $\mathbf{A}$ by solving (6.10)).

Up to this point, readers might wonder that if a pair of (Θ_1, Θ_2) whose $\mathcal{B}^{\Theta_1} = \mathcal{B}^{\Theta_2}$ does really exist. Let's consider the following example.

Example 6.3.19. Consider two different chainable sequences ($r_1 \neq r_2$):

$$\Theta_i = \{(1, m, r_i, 1), (1, r_i, n, 1)\}, \quad i = 1, 2.$$

For $i = 1, 2$, $\mathcal{F}^{\Theta_i} = \mathbb{R}^{m \times r_i} \times \mathbb{R}^{r_i \times n}$, therefore, $\mathcal{B}^{\Theta_i}$ is the set of matrices of size $m \times n$ whose rank is at most r_i. If $r_i \geq \min(m, n)$ for both $i = 1, 2$, then $\mathcal{B}^{\Theta_1} = \mathcal{B}^{\Theta_2} = \mathbb{R}^{m \times n}$.

We define an equivalence relation between chainable sequences (Θ_1, Θ_2) of DB parameters:

$$\Theta_1 \sim \Theta_2 \iff \mathcal{B}^{\Theta_1} = \mathcal{B}^{\Theta_2}. \tag{6.18}$$

Due to our reasoning, among Θ of the same equivalence class, it is natural to work with the one with the minimum $\|\Theta\|_0$. We therefore give a sufficient condition on a chainable sequence Θ for which we can construct an equivalent sequence Θ' such that $\|\Theta'\|_0 < \|\Theta\|_0$. Our approach is based on the following definitions.

Definition 6.3.20 (Redundant pair of DB parameters)**.** A q-chainable pair of DB parameters $\theta_1 = (a_1, b_1, c_1, d_1)$ and $\theta_2 = (a_2, b_2, c_2, d_2)$ is *redundant* if $q > \min(b_1, c_2)$.

Definition 6.3.21 (Redundant sequence of DB parameters)**.** Consider $\Theta = (\theta_\ell)_{\ell=1}^N$ a chainable sequence of DB parameters (cf. Definition 6.3.10), we say that Θ is *ℓ-redundant* if $(\theta_\ell, \theta_{\ell+1})$ is redundant.

A chainable sequence of DB parameters $\Theta = (\theta_\ell)_{\ell=1}^N$ is *redundant* if it is ℓ-redundant for some integer $\ell \in [\![N-1]\!]$.

Lemma 6.3.22. *If the chainable sequence* $\Theta = (\theta_\ell)_{\ell=1}^N$ *is redundant, then for some* $\ell \in [\![N-1]\!]$*, the chainable sequence* $\Theta' := (\theta_1, \dots, \theta_{\ell-1}, \theta_\ell * \theta_{\ell+1}, \theta_{\ell+2}, \dots, \theta_N)$ *is equivalent to* Θ*, and* $\|\Theta'\|_0 < \|\Theta\|_0$.

The proof of Lemma 6.3.22 is deferred to Section 6.7.4. When Θ is a redundant sequence, Lemma 6.3.22 gives a procedure to construct an equivalent sequence Θ' with $|\Theta'| = |\Theta| - 1$. The procedure can be repeated on the obtained Θ', until the final equivalent sequence Θ'' is no longer redundant, or $|\Theta''| = 1$. By construction, the obtained equivalent sequence Θ'' has fewer parameters than Θ.

Hereinafter, without explicit clarification, a sequence Θ is assumed to be chainable and non-redundant.

In summary, in this section, we already discussed a formal definition of DB parameters and DB factors in Definition 6.3.1. We then introduced the notion and properties of chainability and used them to define chainable sequences of DB parameters (cf. Definition 6.3.10). This is our sufficient condition for which the problems (6.10) and (6.11) can admit an approximate algorithm in the sense of (6.12). In the next section, we describe the algorithm in question.

6.4 A hierarchical algorithm for deformable butterfly factorization with chainable Θ

One of the main contributions of this chapter is to propose a hierarchical algorithm that satisfies (6.12). Our proposed algorithm is inspired by the hierarchical factorization algorithm [LZRG22, Section 3], which was developed for the square dyadic butterfly (cf. Table 6.1). It is worth emphasizing that the hierarchical factorization in [LZRG22] naturally generalizes itself under the assumption of chainability of Θ. Therefore, in this section, we will first discuss an adaptation of the algorithm of [LZRG22] for deformable butterfly factorization. While it is natural to hope that this adapted algorithm will have an error bound in the form of (6.12), we will reason that this is, in fact, impossible. This motivates us to make a small tweak in the adapted algorithm to make the bound in (6.12) viable.

The first step towards that goal is to consider the simplest case: $|\Theta| = 2$.

6.4.1 An algorithm for deformable butterfly factorization when $|\Theta| = 2$

When there are only two factors, the problem (6.10) is simply an instance of (6.1.FSMF) with $(\mathbf{L}, \mathbf{R}) = (\mathbf{S}_{\theta_1}, \mathbf{S}_{\theta_2})$. Despite being generally NP-hard (Section 3.3), (6.1.FSMF) becomes tractable under certain assumptions on $(\mathbf{L}, \mathbf{R})$ (cf. Theorem 6.2.3). Interestingly, for any chainable (θ_1, θ_2), the pair $(\mathbf{S}_{\theta_1}, \mathbf{S}_{\theta_2})$ satisfies the conditions in Theorem 6.2.3. The result is formulated in the following lemma:

Lemma 6.4.1. *If* (θ_1, θ_2) *is q-chainable (with* $\theta_i = (a_i, b_i, c_i, d_i), i = 1, 2$*), then* $(\mathbf{S}_{\theta_1}, \mathbf{S}_{\theta_2})$ *satisfies the conditions of Theorem 6.2.3. Moreover, for each* $P \in \mathcal{P}(\mathbf{S}_{\theta_1}, \mathbf{S}_{\theta_2})$*,* $|P| = q, |R_P| = b_1, |C_P| = c_2$ *(cf.* $\mathcal{P}(\mathbf{L}, \mathbf{R}), R_P, C_P$ *defined as in Definition 7.6.4).*

The proof of Lemma 6.4.1 is deferred to Section 6.7.5. By Lemma 6.4.1, we conclude that one can find the optimal factors for the deformable butterfly factorization with $|\Theta| = 2$ using Algorithm 5. Its proof is based on the structures of the DB supports defined in

Definition 6.3.1. As we will see in the upcoming section, Algorithm 5 is the first building block for the hierarchical algorithm that will work for Θ of arbitrary length.

6.4.2 A hierarchical algorithm for deformable butterfly factorization when $|\Theta| \geq 2$

When the length of a chainable Θ is arbitrary, one cannot directly apply Algorithm 5 because Algorithm 5 only works for *two-factor* matrix factorization. To overcome this difficulty, the hierarchical algorithm in [LZRG22] exploits the following idea: instead of factorizing a matrix $\mathbf{A}$ into $|\Theta|$ factors immediately, the algorithm seeks to factorize the target matrix $\mathbf{A}$ into only two factors with certain support constraints at each step and recursively applies this procedure to find all the factors with the initial support constraints. Figure 6.2 illustrates this hierarchical approach when $N = 4$. In fact, there are many ways to "split" the factorization. As shown in [LZRG22], each way of splitting is equivalent to a binary tree.

Using a similar idea, we present our adaptation of the algorithm in [LZRG22] to the deformable butterfly factorization problem as follows: in our case, a node in the binary tree corresponds to an interval $[\![l, r]\!], 1 \leq l \leq r \leq |\Theta|$. To make the notations more straightforward, we simply name a node by its corresponding interval, i.e., node $[\![l, r]\!]$. The root is always the node $[\![1, N]\!], N := |\Theta|$. Every non-leaf node $[\![l, r]\!], l < r$ has exactly two children of the form $[\![l, s]\!]$ and $[\![s + 1, r]\!], l \leq s < r$. A leaf corresponds to an interval $[\![l, l]\!]$ for some $1 \leq l \leq N$. The binary tree has exactly $(N - 1)$ non-leaf nodes and N leaves.

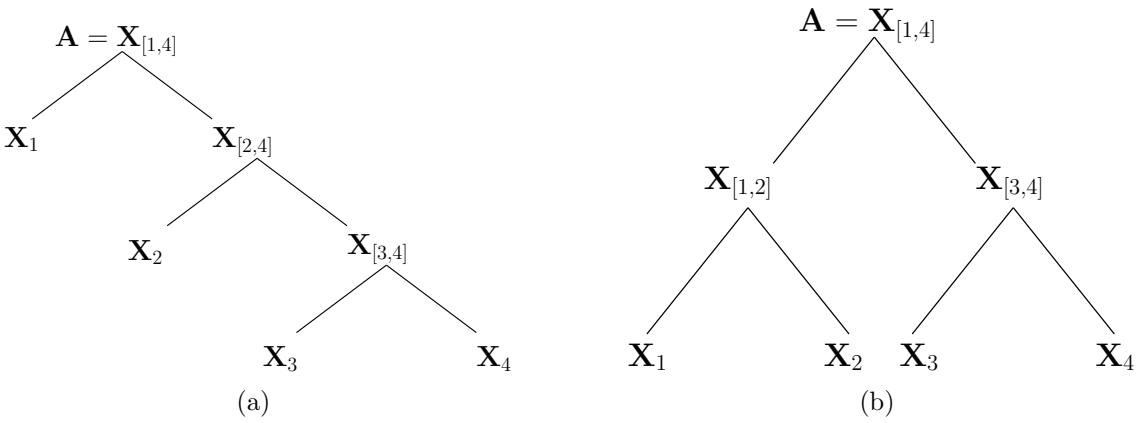


Figure 6.2: (a): factorization from left to right, (b): factorization at the middle.

Once the tree $\mathcal{T}$ is constructed, we calculate the matrix $\mathbf{X}_{[\![l,r]\!]}$ corresponding to the node $[\![l, r]\!]$ in a recursive manner: Initially, the matrix $\mathbf{X}_{[\![1,N]\!]}$ at root is equal to $\mathbf{A}$. At each node $[\![l, r]\!]$ whose matrix $\mathbf{X}_{[\![l,r]\!]}$ is already available/calculated, we compute the matrix $\mathbf{X}_{[\![l,s]\!]}$ and $\mathbf{X}_{[\![s+1,r]\!]}$ for its two children $[\![l, s]\!]$ and $[\![s + 1, r]\!]$ by using Algorithm 5 to solve

the following instance of (6.1.FSMF):

$$\begin{aligned} \text{Minimize} \quad & \|\mathbf{X}_{[\![l,r]\!]} - \mathbf{X}_{[\![l,s]\!]}\mathbf{X}_{[\![s+1,r]\!]}\|_F \\ \text{Subject to} \quad & \operatorname{supp}(\mathbf{X}_{[\![l,s]\!]}) \subseteq \mathbf{S}_{\theta_l * \ldots * \theta_s}, \\ & \operatorname{supp}(\mathbf{X}_{[\![s+1,r]\!]}) \subseteq \mathbf{S}_{\theta_{s+1} * \ldots * \theta_r}. \end{aligned} \tag{6.19}$$

In words, the matrix $\mathbf{X}_{[\![l,r]\!]}$ is an intermediate factor that will be eventually approximated by a product $\mathbf{X}_l \ldots \mathbf{X}_r$ where $\mathbf{X}_\ell \in \mathcal{F}^{\theta_\ell}, l \leq \ell \leq r$. The matrix corresponds to the leaf $[\![l,l]\!]$ clearly belongs to $\mathcal{F}^{\theta_l}$ and is, in fact, the lth factor returned by the hierarchical algorithm.

The pseudo-code of this hierarchical algorithm, written in a recursive manner, is presented in Algorithm 6.

Algorithm 6 Hierarchical algorithm (recursive version)

Require: $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\Theta = (\theta_\ell)_{\ell=1}^N$ and a binary tree $\mathcal{T}$
1: **if** $N = 1$ **then**
2: **return** $\mathbf{A}$.
3: **end if**
4: $[\![1,s]\!], [\![s+1,N]\!] \leftarrow$ two children of $[\![1,N]\!]$.
5: $\mathcal{T}_{\text{left}}, \mathcal{T}_{\text{right}} \leftarrow$ the left and right subtrees of $\mathcal{T}$.
6: $(\theta_{\text{left}}, \theta_{\text{right}}) \leftarrow (\theta_1 * \ldots * \theta_s, \theta_{s+1} * \ldots * \theta_N)$.
7: $\mathbf{X}_{[\![1,s]\!]}, \mathbf{X}_{[\![s+1,N]\!]} \leftarrow$ Algorithm 5$(\mathbf{A}, \mathbf{S}_{\theta_{\text{left}}}, \mathbf{S}_{\theta_{\text{right}}})$.
8: `left_factors` $\leftarrow$ Algorithm 6$(\mathbf{X}_{[\![1,s]\!]}, (\theta_1, \ldots, \theta_s), \mathcal{T}_{\text{left}})$.
9: `right_factors` $\leftarrow$ Algorithm 6$(\mathbf{X}_{[\![s+1,N]\!]}, (\theta_{s+1}, \ldots, \theta_N), \mathcal{T}_{\text{right}})$.
10: **return** `left_factors` and `right_factors`.

In general, the factors obtained by Algorithm 6 are not guaranteed to be the optimal solutions of Equation (6.10). However, at each two-factor factorization step (line 7 of Algorithm 6), the factors $\mathbf{X}_{[\![1,s]\!]}$ and $\mathbf{X}_{[\![s+1,N]\!]}$ are indeed optimal solutions of (6.19). Thus, this algorithm is, in fact, a greedy algorithm. This is an important observation that will be exploited later. To show that this observation is true, we rely on the following key lemma.

Lemma 6.4.2. *Consider $(\theta_\ell)_{\ell=1}^N$ a $(q_1, \ldots, q_{N-1})$-chainable sequence of DB parameters, for any $1 \leq l \leq s < r \leq N$, the DB parameters $(\theta_l * \ldots * \theta_s, \theta_{s+1} * \ldots * \theta_r)$ is q_s-chainable.*

The proof of Lemma 6.4.2 is deferred to Section 6.7.6. Its proof is based on (6.16) and the definition of chainability (cf. Definitions 6.3.7 and 6.3.10). Consequently, the supports $\mathbf{S}_{\theta_l * \ldots * \theta_s}$, $\mathbf{S}_{\theta_{s+1} * \ldots * \theta_r}$ satisfy the conditions of Theorem 6.2.3 (cf. Lemma 6.4.1), meaning that (6.19) can be solved optimally by Algorithm 5 in polynomial time.

The proposed hierarchical algorithm works for any chainable sequence Θ, and is an extension of the hierarchical algorithm from [LZRG22, ZRG23] that was introduced only for the square dyadic butterfly (cf. Table 6.1). The hierarchical approach is greedy in the sense that it seeks to factorize the target matrix $\mathbf{A}$ by performing successive factorization into two factors until the number of desired factors is obtained.

6.4.3 An inconvenience of Algorithm 6

In this section, we argue that with the current form of Algorithm 6, the control of the approximation error in the form of (6.12) is impossible. Indeed, consider the following example:

Example 6.4.3. Consider a chainable sequence of DBPs $\Theta := \{\theta_i\}_{i=1}^N, N = 3$ given by $\Theta = \{(1,1,1,4),(1,2,2,2),(2,2,2,1)\}$. We have the support constraints in binary matrix forms of Θ given by:

$$\mathbf{S}_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \mathbf{S}_2 = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}, \quad \mathbf{S}_3 = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

Consider a matrix $\mathbf{A} \in \mathcal{B}^\Theta$ given by:

$$\mathbf{A} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} = \mathbf{S}_1' \mathbf{S}_2 \mathbf{S}_3, \quad \text{where} \quad \mathbf{S}_1' = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The distance between $\mathbf{A}$ and the set $\mathcal{B}^\Theta$ is, thus, zero.

What does Algorithm 6 do to factorize $\mathbf{A}$ into 3 factors associated to Θ with a left-to-right tree (cf. Figure 6.2a)? In the first step, it will perform Algorithm 5 with $(\mathbf{A}, \mathbf{S}_1, \mathbf{1}_{4\times 4})$ (since $\theta_2 * \theta_3 = (1,4,4,1)$, a 4×4 matrix with no support constraint) and obtain two factors $\mathbf{X}_1$ and $\mathbf{X}_{[2,3]}$. Note that Algorithm 5 can output *many possible optimal solutions* $(\mathbf{X}^1, \mathbf{X}^{[2,3]})$ of (6.1.FSMF), as long as they satisfy $\mathbf{X}_1 \mathbf{X}_{[2,3]} = \mathbf{A}$. Then it will continue to factorize $\mathbf{X}^{[2,3]}$ into $\mathbf{X}_2$ and $\mathbf{X}_3$ using Algorithm 5 with input $(\mathbf{X}_{[2,3]}, \mathbf{S}_2, \mathbf{S}_3)$. Here is the problem: if $\mathbf{X}_1 = \mathbf{S}_1'$, then the first row of $\mathbf{X}_{[2,3]}$ can be arbitrary (while the remaining rows have to match those of $\mathbf{A}$). A choice of $\mathbf{X}_{[2,3]}$ that does not admit a factorization to $\mathbf{X}_2$ and $\mathbf{X}_3$ (with support constraints $\mathbf{S}_{\theta_2}$ and $\mathbf{S}_{\theta_3}$ respectively) can possibly make $\mathbf{A} \neq \mathbf{X}_1 \mathbf{X}_2 \mathbf{X}_3$. Therefore, an error bound as in (6.12) is impossible.

This problem results from the ambiguity of the choice of optimal factors given by Algorithm 5 in line 7 of Algorithm 6: there are plenty of optimal pairs of factors in each iteration, and the choice in one iteration will affect the following factorization in the recursion (by changing the input matrix of Algorithm 5).

To fix the issue presented in Example 6.4.3, we need to make a "good choice" of the input matrices of Algorithm 5 in line 7. This tweak is what we will apply to Algorithm 6 and establish a provable guarantee based on this modification. This motivates us to introduce an unrolled version of Algorithm 6.

6.4.4 An non-recursive alternative for Algorithm 6

To unroll the algorithm described in Algorithm 6, we observe that the role of the binary tree $\mathcal{T}$ is to encode how to "split" the factorization. At each two-factor factorization step, we only need to access the intervals $[\![l, r]\!]$, the values s, $l \leq s < r$, and solve (6.19). It

is noteworthy that the values s from all iterations form a permutation of $[\![N-1]\!]$ (since $1 \leq s < N$ and all values of s are distinct). Thus, an alternative to store that information encoded by the binary tree $\mathcal{T}$ is to use:

1. A permutation σ of $[\![N-1]\!]$, to store the values s in the order the two-factor factorization is performed.

2. A list of intervals `partition` consists of $[\![l, r]\!]$ whose $\mathbf{X}_{[\![l,r]\!]}$ is already computed.

Incorporating this idea gives us Algorithm 7, a non-recursive alternative of Algorithm 6. Algorithm 6 and Algorithm 7 are exactly equivalent (if the permutation σ encodes the

Algorithm 7 Hierarchical algorithm (non-recursive version)

Require: $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\Theta = (\theta_\ell)_{\ell=1}^N$, a permutation $\sigma := (\sigma_k)_{k=1}^{N-1}$ of $[\![N-1]\!]$

1: `partition` $\leftarrow \{P_1\}$ where we denote $P_1 = [\![1, N]\!]$.
2: `factors` $\leftarrow \{\mathbf{X}_{P_1}\}$ where we denote $\mathbf{X}_{P_1} = \mathbf{A}$.
3: **for** $\ell = 1, \ldots, N-1$ **do**
4: $\quad (P_j)_{j=1}^{\ell} \leftarrow$ `partition`
5: $\quad (\mathbf{X}_{P_j})_{j=1}^{\ell} \leftarrow$ `factors`
6: $\quad j \leftarrow$ the unique $j \in [\![\ell]\!]$ such that $P_j := [\![l, r]\!] \ni s := \sigma_i$
7: $\quad (\theta_{\text{left}}, \theta_{\text{right}}) \leftarrow (\theta_l * \ldots * \theta_s, \theta_{s+1} * \ldots * \theta_r)$
8: $\quad (\mathbf{X}_{[\![l,s]\!]}, \mathbf{X}_{[\![s+1,r]\!]}) \leftarrow$ Algorithm 5$(\mathbf{X}_{[\![l,r]\!]}, \mathbf{S}_{\theta_{\text{left}}}, \mathbf{S}_{\theta_{\text{right}}})$
9: $\quad$ `partition` $\leftarrow (P_1, \ldots, P_{j-1}, [\![l, s]\!], [\![s+1, r]\!], P_{j+1}, \ldots, P_\ell)$
10: $\quad$ `factors` $\leftarrow (\mathbf{X}_{P_1}, \ldots, \mathbf{X}_{P_{j-1}}, \mathbf{X}_{[\![l,s]\!]}, \mathbf{X}_{[\![s+1,r]\!]}, \mathbf{X}_{P_{j+1}}, \ldots, \mathbf{X}_{P_\ell})$
11: **end for**
12: **return** `factors`

same information as the binary tree $\mathcal{T}$). As explained in Example 6.4.3, we cannot possibly achieve any error bound with Algorithm 7. However, the unrolled version Algorithm 7 offers an idea that is more difficult to see from the recursive version Algorithm 6: at the ℓth factorization, instead of directly using $\mathbf{X}_{P_j}$ as the target matrix for Algorithm 5, one can produce a sequence $(\mathbf{X}'_{P_1}, \ldots, \mathbf{X}'_{P_\ell})$ such that:

1. For all $1 \leq i \leq \ell$, we have: $\operatorname{supp}(\mathbf{X}'_{P_i}) \subseteq \mathbf{S}_{\theta_l * \ldots * \theta_r}$ where $P_i = [\![l, r]\!]$.

2. The product $\mathbf{X}'_{P_1} \ldots \mathbf{X}'_{P_\ell} = \mathbf{X}_{P_1} \ldots \mathbf{X}_{P_\ell}$.

We then use $\mathbf{X}'_{P_j}$ (j defined as in line 6 of Algorithm 7) as the target matrix instead of $\mathbf{X}_{P_j}$. Such sequence of $(\mathbf{X}'_{P_1}, \ldots, \mathbf{X}'_{P_\ell})$ is not rare. For example, we can take $(\mathbf{X}'_{P_1}, \ldots, \mathbf{X}'_{P_\ell}) = (\mathbf{X}_{P_1}\mathbf{D}_1, \mathbf{D}_1^{-1}\mathbf{X}_{P_2}\mathbf{D}_2, \ldots, \mathbf{D}_{\ell-1}^{-1}\mathbf{X}'_{P_\ell})$ for any sequence $(\mathbf{D}_1, \ldots, \mathbf{D}_{\ell-1})$ of invertible diagonal matrices. This choice is valid since the support is invariant under the multiplication to the left and right by invertible diagonal matrices. This flexibility allows us to possibly avoid the difficulty presented in Example 6.4.3. With this idea, the main question is: how to choose $\mathbf{X}'_{P_i}$ given the factors $\mathbf{X}_{P_i}$? We propose such a choice in the following section. In addition, we provide the error bounds of the resulted algorithm - Algorithm 8 - which is obtained by incorporating our choice of $\mathbf{X}'_{P_i}$.

6.5 A hierarchical algorithm for deformable butterfly factorization with error bound guarantee

In this section, we first present Algorithm 8, a version of Algorithms 6 and 7 that we will prove to admit an error bound of the form (6.12) in Section 6.5.1. After that, we provide and discuss in detail the error bounds of Algorithm 8 in Section 6.5.2.

6.5.1 A hierarchical algorithm for deformable butterfly factorization with orthonormalization

We present in Algorithm 8, a modified version of Algorithms 6 and 7. In comparison to Algorithm 7, Algorithm 8 has two additional "for" loops that uses Algorithm 9:

1. The first "for loop" (lines 8-10): Performing Algorithm 9 for all pairs $(\mathbf{X}_{P_k}, \mathbf{X}_{P_{k+1}}), k < j$ (j is defined as in line 6) where $\theta_{P_k} = \theta_{l_k} * \ldots * \theta_{r_k}$, $P_k = [\![l_k, r_k]\!]$. This operation is carried out forwards (i.e., from $k = 1$ to $j - 1$).

2. The second "for loop" (lines 11-13): Likewise, for all pairs $(\mathbf{X}_{P_k}, \mathbf{X}_{P_{k+1}}), k > j$. However, this operation has to be performed backwards (cf. line 11).

We call this part in blue: *orthonormalization operations*. Below is the pseudo-code for

Algorithm 8 Hierarchical algorithm with orthonormalization

Require: $\mathbf{A} \in \mathbb{R}^{m \times n}$, non-redundant $\Theta = (\theta_\ell)_{\ell=1}^N$, a permutation $\sigma := (\sigma_k)_{k=1}^{N-1}$ of $[\![N-1]\!]$.

1: $(q_\ell)_{\ell=1}^{N-1} \leftarrow$ parameters such that $(\theta_\ell, \theta_{\ell+1})$ is q_ℓ-chainable for any $\ell \in [\![N-1]\!]$
2: `partition` $\leftarrow \{P_1\}$ where we denote $P_1 = [\![N-1]\!]$.
3: `factors` $\leftarrow \{\mathbf{X}_{P_1}\}$ where we denote $\mathbf{X}_{P_1} = \mathbf{A}$.
4: **for** $\ell = 1, \ldots, N-1$ **do**
5: $\quad (P_j)_{j=1}^{\ell} \leftarrow$ `partition`.
6: $\quad (\mathbf{X}_{P_j})_{j=1}^{\ell} \leftarrow$ `factors`.
7: $\quad j \leftarrow$ the unique $j \in [\![\ell]\!]$ such that $P_j := [\![l, r]\!] \ni s := \sigma_i$.
8: $\quad$ **for** $k = 1, \ldots, j-1$ **do**
9: $\quad\quad (\mathbf{X}_{P_k}, \mathbf{X}_{P_{k+1}}) \leftarrow$ Algorithm 9$(\theta_{P_k}, \theta_{P_{k+1}}, \mathbf{X}_{P_k}, \mathbf{X}_{P_{k+1}}, \texttt{column})$.
10: $\quad$ **end for**
11: $\quad$ **for** $k = \ell, \ldots, j+1$ **do**
12: $\quad\quad (\mathbf{X}_{P_{k-1}}, \mathbf{X}_{P_k}) \leftarrow$ Algorithm 9$(\theta_{P_{k-1}}, \theta_{P_k}, \mathbf{X}_{P_{k-1}}, \mathbf{X}_{P_k}, \texttt{row})$.
13: $\quad$ **end for**
14: $\quad (\theta_{\text{left}}, \theta_{\text{right}}) \leftarrow (\theta_l * \ldots * \theta_s, \theta_{s+1} * \ldots * \theta_r)$.
15: $\quad (\mathbf{X}_{[\![l,s]\!]}, \mathbf{X}_{[\![s+1,r]\!]}) \leftarrow$ Algorithm 5$(\mathbf{X}_{[\![l,r]\!]}, \mathbf{S}_{\theta_{\text{left}}}, \mathbf{S}_{\theta_{\text{right}}})$.
16: $\quad$ `partition` $\leftarrow (P_1, \ldots, P_{j-1}, [\![l, s]\!], [\![s+1, r]\!], P_{j+1}, \ldots, P_\ell)$.
17: $\quad$ `factors` $\leftarrow (\mathbf{X}_{P_1}, \ldots, \mathbf{X}_{P_{j-1}}, \mathbf{X}_{[\![l,s]\!]}, \mathbf{X}_{[\![s+1,r]\!]}, \mathbf{X}_{P_{j+1}}, \ldots, \mathbf{X}_{P_\ell})$.
18: **end for**
19: **return** `factors`

Algorithm 9. A more formal presentation of Algorithm 9 will be provided in Section 6.7.8. We discuss here some intuitions and properties of Algorithm 9 that are useful for the

Algorithm 9 Column/row-orthonormalization

Require: A chainable, non-redundant (θ_1, θ_2), $\mathbf{X} \in \mathcal{F}^{\theta_1}$, $\mathbf{Y} \in \times\mathcal{F}^{\theta_2}$, $t \in \{\texttt{column}, \texttt{row}\}$.
1: $(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}) \leftarrow (\mathbf{0}, \mathbf{0})$.
2: **for** $P \in \mathcal{P}(\mathbf{S}_{\theta_1}, \mathbf{S}_{\theta_2})$ **do**
3: **if** t is `column` **then**
4: $(\mathbf{Q}, \mathbf{R}) \leftarrow$ QR-decomposition of $\mathbf{X}[R_P, P]$.
5: $\tilde{\mathbf{X}}[R_P, P] \leftarrow \mathbf{Q}$.
6: $\tilde{\mathbf{Y}}[P, C_P] \leftarrow \mathbf{R}\mathbf{Y}[P, C_P]$.
7: **else if** t is `row` **then**
8: $(\mathbf{Q}, \mathbf{R}) \leftarrow$ QR-decomposition of $\mathbf{Y}[P, C_P]^\top$.
9: $\tilde{\mathbf{X}}[R_P, P] \leftarrow \mathbf{X}[R_P, P]\mathbf{R}^\top$.
10: $\tilde{\mathbf{Y}}[P, C_P] \leftarrow \mathbf{Q}^\top$.
11: **end if**
12: **end for**
13: **return** $(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$.

analysis of Algorithm 8. Results in the following (namely Lemmas 6.5.1, 6.5.3 and 6.5.4 are proved in Section 6.7.9 while Lemma 6.5.2 is proved in Section 6.7.7). .

1. Algorithm 9 takes a pair of chainable, non-redundant (θ_1, θ_2), two θ_i-DB factors $(i = 1, 2)$ $\mathbf{X}$ and $\mathbf{Y}$ and a variable t admitting two valid values: `column` or `row`. It then returns two θ_i-DB factors $(i = 1, 2)$ $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$. This result is formalized as:

   Lemma 6.5.1. *Consider the output* $(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$ *of Algorithm 9 with a valid input* $(\theta_1, \theta_2, \mathbf{X}, \mathbf{Y}, t)$, *we have:* $\tilde{\mathbf{X}} \in \mathcal{F}^{\theta_1}, \tilde{\mathbf{Y}} \in \mathcal{F}^{\theta_2}$.

2. Note that to be able to use Algorithm 9, $(\theta_{P_k}, \theta_{P_{k+1}})$ of lines 9 and 12 has to be chainable and *non-redundant*. In fact, since $(\theta_{P_k}, \theta_{P_{k+1}})$ always has the form of $(\theta_{l_k} * \ldots \theta_{r_k}, \theta_{l_{k+1}} * \ldots * \theta_{r_{k+1}})$ with $r_k + 1 = l_{k+1}$, the chainability is followed by Lemma 6.4.2. The non-redundancy is shown in the following *lemma*.

   Lemma 6.5.2. *Let* $\Theta = (\theta_\ell)_{\ell=1}^N$ *be a chainable and non-redundant sequence of DB parameters, then for any interger* $1 \leq l \leq s < r \leq N$, *the pair* $(\theta_l * \ldots * \theta_s, \theta_{s+1} * \ldots * \theta_r)$ *is not redundant.*

3. If $t =$ `column`, then the submatrices of $\tilde{\mathbf{X}}$ of the form $\tilde{\mathbf{X}}[R_P, P], P \in \mathcal{P}(\mathbf{S}_{\theta_1}, \mathbf{S}_{\theta_2})$ become orthonormal by column ($\mathcal{P}, R_P$ are defined as in Definition 7.6.4). Indeed, they are set to $\mathbf{Q}$ (in line 5 of Algorithm 9) where $(\mathbf{Q}, \mathbf{R})$ is the QR-decomposition [3] of submatrices of $\mathbf{X}[R_P, P]$. The orthonormality by column of $\tilde{\mathbf{X}}[R_P, P]$ is very important in the proof of our error bound of Algorithm 8. In fact, we require (θ_1, θ_2) to be non-redundant because it is impossible to have a matrix of size $|R_P| \times |P|$ orthonormal by column if $|R_P| = b_i < p = |P|$ (cf. Lemma 6.4.1). The non-redundancy introduced in Section 6.3 is not only for the efficiency of matrix-vector product, but it also serves as an important technical assumptions. However, it is

[3] In this context, a QR-decomposition of a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}, m \geq n$ is a pair of matrices $(\mathbf{Q}, \mathbf{R})$ in which $\mathbf{Q} \in \mathbb{R}^{m \times n}$ is orthonormal by column and $\mathbf{R}^{n \times n}$ is an upper triangular matrix.

worth emphasizing that this does not make our results more restricted: thanks to Lemma 6.3.22, Θ can be assumed to be non-redundant, without loss of generality.

4. Likewise, if $t = \texttt{column}$, then the submatrices of $\tilde{\mathbf{Y}}$ of the form $\tilde{\mathbf{Y}}[P, C_P], P \in \mathcal{P}(\mathbf{S}_{\theta_1}, \mathbf{S}_{\theta_2})$ become orthonormal by row. That is why the pseudo-code in blue is called orthonormalization operations.

   Using the third and fourth remarks, we can prove the following result:

   Lemma 6.5.3. *Consider the values of the list* $\texttt{factors} := (\mathbf{X}_{P_i})_{i=1}^{\ell}$ *at line 15 and the value* j *(defined as in line 7) of the* ℓ*th iteration of Algorithm 8, define:*

$$\mathbf{X}_{left}^{(\ell)} := \mathbf{X}_{P_1} \dots \mathbf{X}_{P_{j-1}} \tag{6.20}$$

$$\mathbf{X}_{right}^{(\ell)} := \mathbf{X}_{P_{j+1}} \dots \mathbf{X}_{P_\ell} \tag{6.21}$$

   with convention that $\mathbf{X}_{left}$ *(resp.* $\mathbf{X}_{right}$*) is the identity matrix with of size* $a_1 b_1 d_1$ *if* $j = 1$ *(resp.* $a_N c_N d_N$ *if* $j = \ell$*). For any* $(\theta_l * \dots * \theta_r)$*-DB factor* $\mathbf{Y}$ *(*l, r *defined as in line 7), we have:*

$$\|\mathbf{X}_{left}^{(\ell)} \mathbf{Y} \mathbf{X}_{right}^{(\ell)}\|_F = \|\mathbf{Y}\|_F \tag{6.22}$$

   We can see that the matrix $\mathbf{X}_{\text{left}}^{(\ell)}$ and $\mathbf{X}_{\text{right}}^{(\ell)}$ behave like orthonormal matrices by column and row respectively, in the sense that their multiplication preserves the Frobenius norm of the matrix $\mathbf{Y}$ (with additional assumption on the matrix $\mathbf{Y}$). Intuitively, this choice will avoid "ill-conditioned" factors such as $\mathbf{X}_1$ in Example 6.4.3.

5. Last but not least, in both cases $t \in \{\texttt{row}, \texttt{column}\}$, we have:

$$(\mathbf{X}_1 \mathbf{X}_2)[R_P, C_P] = (\tilde{\mathbf{X}}_1 \tilde{\mathbf{X}}_2)[R_P, C_P]$$

   This allows us to prove the following result:

   Lemma 6.5.4. *Consider the output* $(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$ *of Algorithm 9 with a valid input* $(\theta_1, \theta_2, \mathbf{X}, \mathbf{Y}, t)$*, we have:* $\tilde{\mathbf{X}}\tilde{\mathbf{Y}} = \mathbf{X}\mathbf{Y}$*.*

   A direct corollary of Lemma 6.5.4 is given as follows:

   Corollary 6.5.5. *At the* ℓ*th iteration of Algorithm 8, the product* $\mathbf{X}_{P_1} \dots \mathbf{X}_{P_\ell}$ *is invariant during the orthonormalization operations (line 8-13).*

6.5.2 Approximation error of the hierarchical algorithm with orthonormalization

It remains to prove a bound for the approximation error of Algorithm 8. To describe such a bound, we introduce the following definition.

Definition 6.5.6 (First level factorization)**.** Given a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ and a chainable, non-redundant $\Theta := (\theta_\ell)_{\ell=1}^N$, for $1 \le s \le N-1$ define:

$$E_s^{\Theta}(\mathbf{A}) := \min_{\mathbf{X},\mathbf{Y}} \|\mathbf{A} - \mathbf{X}\mathbf{Y}\|_F \quad \text{such that} \quad \text{supp}(\mathbf{X}) \subseteq \mathbf{S}_{1,s}, \text{supp}(\mathbf{Y}) \subseteq \mathbf{S}_{s+1,N} \tag{6.23}$$

where $\mathbf{S}_{l,r} := \mathbf{S}_{\theta_l * \dots * \theta_r}, 1 \le l \le r \le N$.

The problem in Definition 6.5.6 is named "first level factorization" because it indeed corresponds to the first factorization of the form (6.19) of Algorithm 8. It is important in the analysis that follows, for two reasons: It corresponds to a polynomially solvable instance of (6.1.FSMF) (cf. Theorem 6.2.3 and Lemma 6.4.1); moreover, the optimal value of the first level factorization is also a lower bound for the optimal error approximation of (6.10). It is presented in the following lemma.

Lemma 6.5.7. *For each matrix* $\mathbf{A}$*, each chainable* $\Theta := (\theta_\ell)_{\ell=1}^N$ *and* $1 \leq s \leq N-1$*, we have:* $E_s^\Theta(\mathbf{A}) \leq E^\Theta(\mathbf{A})$ *(*$E^\Theta(\mathbf{A})$ *is defined in* (6.10)*).*

Proof. If $\mathbf{B} \in \mathcal{B}^\Theta$, there exists $\mathbf{X}_\ell, \ell = 1, \ldots, N$ such that $\mathbf{B} = \prod_{\ell=1}^N \mathbf{X}_\ell$ such that $\mathbf{X}_\ell \in \mathcal{F}^{\theta_\ell}$. Since $\operatorname{supp}(\mathbf{X}_1 \ldots \mathbf{X}_\ell) \subseteq \mathbf{S}_{1,s}$ and $\operatorname{supp}(\mathbf{X}_{\ell+1} \ldots \mathbf{X}_N) \subseteq \mathbf{S}_{s,N}$ (cf. Lemma 6.3.16), we have:

$$\mathcal{B}^\Theta \subseteq \{\mathbf{XY} \,|\, \operatorname{supp}(\mathbf{X}) \subseteq \mathbf{S}_{1,s}, \operatorname{supp}(\mathbf{Y}) \subseteq \mathbf{S}_{s+1,N}\}, \forall 1 \leq s \leq N-1. \tag{6.24}$$

Therefore, $E_s^\Theta(\mathbf{A}) \leq \|\mathbf{A} - \mathbf{B}\|_F, \forall \mathbf{B} \in \mathcal{F}^\Theta$. Thus, $E_s^\Theta(\mathbf{A}) \leq E^\Theta(\mathbf{A})$. □

A direct corollary of Lemma 6.5.7 is given by:

Corollary 6.5.8. *Under the same assumptions as Lemma 6.5.7, we have:*

$$\max_{1 \leq s \leq N-1} E_s^\Theta(\mathbf{A}) \leq E^\Theta(\mathbf{A}).$$

Approximation error of Algorithm 8 for arbitrary permutation σ

The following theorem is the main result of this chapter: it provides a bound for the approximation error obtained by the hierarchical algorithm described in Algorithm 8 with an arbitrary permutation σ.

Theorem 6.5.9 (Error approximation of hierarchical algorithm)**.** *Consider a matrix* $\mathbf{A} \in \mathbb{R}^{m \times n}$*, a chainable, non-redundant* $\Theta := (\theta_\ell)_{\ell=1}^N$ *and a permutation* σ *of* $[\![N-1]\!]$*. Using* $\mathbf{A}, \Theta, \sigma$ *as inputs, Algorithm 8 returns* N *factors* $(\mathbf{X}_\ell)_{\ell=1}^N \in \mathcal{F}^\Theta$*, such that:*

$$\|\mathbf{A} - \prod_{\ell=1}^N \mathbf{X}_\ell\|_F \leq \sum_{k=1}^{N-1} 2^{N-1-k} E_{\sigma_k}^\Theta(\mathbf{A}). \tag{6.25}$$

Before proving this result, we discuss several of its interesting corollaries.

Corollary 6.5.10 (Hierarchical factorization with identity permutation)**.** *Under the same assumptions of Theorem 6.5.9 with the identity permutation* $\sigma = (1, \ldots, N-1)$*, we have:*

$$\|\mathbf{A} - \prod_{\ell=1}^N \mathbf{X}_\ell\|_F \leq \sum_{\ell=1}^{N-1} 2^{N-1-\ell} E_\ell^\Theta(\mathbf{A}), \tag{6.26}$$

Corollary 6.5.11 (Hierarchical factorization with arbitrary permutation)**.** *Under the same assumptions of Theorem 6.5.9, we have:*

$$\|\mathbf{A} - \prod_{\ell=1}^N \mathbf{X}_\ell\|_F \leq (2^{N-1} - 1) E^\Theta(\mathbf{A}), \tag{6.27}$$

Proof. It is immediate using Theorem 6.5.9, Corollary 6.5.8 and the famous equality $\sum_{k=1}^{N-1} 2^{N-1-k} = \sum_{k=0}^{N-2} 2^k = 2^{N-1} - 1$. □

Corollary 6.5.12 (A characterization of DB matrix)**.** *Given a chainable, non-redundant* $\Theta := (\theta_\ell)_{\ell=1}^N$*, let* $\mathcal{M}_\ell^\Theta := \{\mathbf{XY} \mid \text{supp}(\mathbf{X}) \subseteq \mathbf{S}_{1,\ell}, \text{supp}(\mathbf{Y}) \subseteq \mathbf{S}_{\ell+1,N}\}$ *(*$\mathbf{S}_{l,r}$ *is defined as in Definition 6.5.6), we have:*

$$\mathcal{B}^\Theta = \bigcap_{\ell=1}^{N-1} \mathcal{M}_\ell^\Theta \tag{6.28}$$

Proof. On the one hand, due to Equation (6.24), we have: $\mathcal{B}^\Theta \subseteq \mathcal{M}_\ell^\Theta$. Thus,

$$\mathcal{B}^\Theta \subseteq \bigcap_{\ell=1}^{N-1} \mathcal{M}_\ell^\Theta.$$

On the other hand, if $\mathbf{A} \in \bigcap_{\ell=1}^{N-1} \mathcal{M}_\ell^\Theta$, then $E_\ell^\Theta(\mathbf{A}) = 0, \forall \ell < N$. Using Theorem 6.5.9, we conclude that Algorithm 8 will return a matrix $\mathbf{A} \in \mathcal{B}^\Theta$ such that $\|\mathbf{A} - \mathbf{B}\|_F = 0$, i.e., $\mathbf{A} = \mathbf{B}$. Thus, $\mathbf{A} \in \mathcal{B}^\Theta$. It implies that:

$$\bigcap_{\ell=1}^{N-1} \mathcal{M}_\ell^\Theta \subseteq \mathcal{B}^\Theta.$$

The proof is, thus, concluded. □

Corollary 6.5.12 provides a way to verify if a matrix $\mathbf{A}$ is an element of $\mathcal{B}^\Theta$. It can be done by checking whether $\mathbf{A} \in \mathcal{M}_\ell^\Theta, \forall \ell = 1, \ldots, |\Theta| - 1$. In fact, one can characterize $\mathcal{M}_\ell^\Theta$ as:

$$\mathcal{M}_\ell^\Theta = \{\mathbf{A} \mid \text{supp}(\mathbf{A}) \subseteq \mathbf{S}_{1,N} \text{ and } \text{rank}(\mathbf{A}[R_P, C_P]) \leq q_\ell, \forall P \in \mathcal{P}(\mathbf{S}_{1,\ell}, \mathbf{S}_{\ell+1,N})\}$$

where q_ℓ is the integer such that $(\theta_\ell, \theta_{\ell+1})$ is q_ℓ-chainable. This characterization is explained in Corollary 6.7.3 of Section 6.7.1. Note that the set $\{\mathbf{A} \mid \text{supp}(\mathbf{A}) \subseteq \mathbf{S}_{1,N}\} = \{\mathbf{A} \mid \mathbf{A}[i,j] = 0, \forall (i,j) \notin \mathbf{S}_{1,N}\}$ and the constraints $\mathbf{A}[i,j] = 0$ is equivalent to constraint the rank of the matrix $(\mathbf{A}[i,j])$ of size 1×1 to be of rank 0. Therefore, if Θ is chainble, checking the membership of a matrix in the set $\mathcal{B}^\Theta$ reduces to compute the ranks of its submatrices.

Back to the proof of Theorem 6.5.9, it is based on Lemma 6.5.13, which is stated as follows:

Lemma 6.5.13. *Consider the* ℓ*th iteration of Algorithm 8 whose inputs are a chainable, non-redundant* $\Theta := (\theta_\ell)_{\ell=1}^N$ *and a matrix* $\mathbf{A}$ *and a permutation* σ*. Define* $\mathbf{X}_{left}^{(\ell)}$ *and* $\mathbf{X}_{right}^{(\ell)}$ *as in Lemma 6.5.3. We have:*

$$E_s^\Theta(\mathbf{X}_{left}^{(\ell)} \mathbf{X}_{[\![l,r]\!]} \mathbf{X}_{right}^{(\ell)}) = \|\mathbf{X}_{[\![l,r]\!]} - \mathbf{X}_{[\![l,s]\!]} \mathbf{X}_{[\![s+1,r]\!]}\|_F, \tag{6.29}$$

where (l, s, r) *and* $(\mathbf{X}_{[\![l,r]\!]}, \mathbf{X}_{[\![l,s]\!]}, \mathbf{X}_{[\![s+1,r]\!]})$ *are defined as in line 7 and line 15 of Algorithm 8, respectively.*

The proof of Lemma 6.5.13 is deferred to Section 6.7.10. We now prove Theorem 6.5.9.

Proof of Theorem 6.5.9. In general, the hierarchical algorithm will perform $(N-1)$ two-factor matrix factorization (line 15 - Algorithm 8). After the ℓth factorization, define $\mathbf{P}_\ell = \prod_{j=1}^{\ell+1} \mathbf{X}_{P_j}$ which is equal to the product of all current factors in the list `factors`. For $\ell = 0$, $\mathbf{P}_\ell = \mathbf{A}$. Also, the matrix $\mathbf{P}_{N-1} \in \mathcal{B}^\Theta$ and we want to show that:

$$\|\mathbf{A} - \mathbf{P}_{N-1}\|_F \leq \sum_{k=1}^{N-1} 2^{N-1-k} E^\Theta_{\sigma_k}(\mathbf{A}),$$

Define $R^\ell = \|\mathbf{A} - \mathbf{P}_\ell\|_F$, we will derive a relationship between $R^{\ell-1}$ and R^ℓ. In fact,

$$R^\ell = \|\mathbf{A} - \mathbf{P}_\ell\|_F \leq \|\mathbf{A} - \mathbf{P}_{\ell-1}\|_F + \|\mathbf{P}_{\ell-1} - \mathbf{P}_\ell\|_F = R_{\ell-1} + \|\mathbf{P}_{\ell-1} - \mathbf{P}_\ell\|_F \tag{6.30}$$

Note that the product of factors in the list `factors` changes from $\mathbf{P}_{\ell-1}$ to $\mathbf{P}_\ell$ when we perform a new two-factor factorization (line 15) at the ℓth iteration. The orthonormalization operations (line 8-13) do not change this product (cf. Corollary 6.5.5). Therefore, after the orthonormalization operations, we still have:

$$\begin{aligned} \mathbf{P}_{\ell-1} &= \mathbf{X}^{(\ell)}_{\text{left}} \mathbf{X}_{[\![l,r]\!]} \mathbf{X}^{(\ell)}_{\text{right}} \\ \mathbf{P}_{\ell} &= \mathbf{X}^{(\ell)}_{\text{left}} \mathbf{X}_{[\![l,s]\!]} \mathbf{X}_{[\![s+1,r]\!]} \mathbf{X}^{(\ell)}_{\text{right}} \end{aligned} \tag{6.31}$$

where $(\mathbf{X}_{[\![l,r]\!]}, \mathbf{X}_{[\![l,s]\!]}, \mathbf{X}_{[\![s+1,r]\!]})$ are defined as in line 15 of Algorithm 8.

Using Lemma 6.5.13, we obtain:

$$E^\Theta_s(\mathbf{P}_{\ell-1}) = \|\mathbf{X}_{[\![l,r]\!]} - \mathbf{X}_{[\![l,s]\!]} \mathbf{X}_{[\![s+1,r]\!]}\|_F \tag{6.32}$$

Moreover,

$$\begin{aligned} \|\mathbf{X}_{[\![l,r]\!]} - \mathbf{X}_{[\![l,s]\!]} \mathbf{X}_{[\![s+1,r]\!]}\|_F &= \|\mathbf{X}_{\text{left}} (\mathbf{X}_{[\![l,r]\!]} - \mathbf{X}_{[\![l,s]\!]} \mathbf{X}_{[\![s+1,r]\!]}) \mathbf{X}_{\text{right}}\|_F \\ &\overset{(6.31)}{=} \|\mathbf{P}_{\ell-1} - \mathbf{P}_\ell\|_F, \end{aligned}$$

where the first equality holds because:

1. If $\ell = 1$ (at the first iteration): then we factorize $\mathbf{X}_{[\![1,N]\!]} = \mathbf{A}$, i.e., $j = \ell = 1$. Thus, both $\mathbf{X}^{(1)}_{\text{left}}$ and $\mathbf{X}^{(1)}_{\text{left}}$ are identity matrices (see the convention in Lemma 6.5.4). The equality holds trivially.

2. If $\ell > 1$ (from second iteration): We have $\mathbf{X}_{[\![l,r]\!]}$ is supported in $\mathbf{S}_{l,r}$. Note that this claim *fails* to hold only for $\mathbf{A} = \mathbf{X}_{[\![1,N]\!]}$. However, since $\ell > 1$, $[\![l,r]\!] \neq [\![1,N]\!]$ and $\text{supp}(\mathbf{X}_{[\![l,r]\!]}) \subseteq \mathbf{S}_{l,r}$. So is the product $\mathbf{X}_{[\![l,s]\!]} \mathbf{X}_{[\![s+1,r]\!]}$ (cf. Lemma 6.3.16). Thus, we can apply Lemma 6.5.3.

Let $\mathbf{A}_s$ be the optimal solution of the sth first level factorization (cf Definition 6.5.6), we have:

$$\|\mathbf{P}_{\ell-1} - \mathbf{P}_\ell\|_F = E^\Theta_s(\mathbf{P}_{\ell-1}) \leq \|\mathbf{P}_{\ell-1} - \mathbf{A}_s\|_F$$

because $\mathbf{A}_s = \mathbf{XY}$ for some $(\mathbf{X}, \mathbf{Y})$ feasible to the sth first level factorization. Using this equation gives us:

$$\|\mathbf{P}_{\ell-1} - \mathbf{P}_\ell\|_F \leq \|\mathbf{P}_{\ell-1} - \mathbf{A}_s\|_F \leq \|\mathbf{P}_{\ell-1} - \mathbf{A}\|_F + \|\mathbf{A} - \mathbf{A}_s\|_F = R_{\ell-1} + E^\Theta_s(\mathbf{A}) \tag{6.33}$$

Combine Equation (6.30) and Equation (6.33), we obtain:

$$R_\ell \leq 2R_{\ell-1} + E^\Theta_s(\mathbf{A}), \forall \ell \geq 1 \tag{6.34}$$

The result follows immediately by unrolling the recursion in (6.34). □

A finer bound for approximation error of Algorithm 8 with identity permutation

In this section, we present a *finer* result for the hierarchical algorithm with σ equal to the identity permutation by replacing the triangle inequality in (6.30) by a Pythagorean-like equality. We distance this result with the general result Theorem 6.5.9, which works for any permutation.

Theorem 6.5.14. *Under the same assumptions of Theorem 6.5.9 except that we use the identity permutation $\sigma = (1, \dots, N-1)$, we have:*

$$\|\mathbf{A} - \prod_{\ell=1}^{N} \mathbf{X}_\ell\|_F^2 \leq 3^{N-2} E_1^{\Theta}(\mathbf{A})^2 + 2 \sum_{kl=2}^{N-1} 3^{N-1-k} E_k l^{\Theta}(\mathbf{A})^2, \tag{6.35}$$

The following corollary is immediate:

Corollary 6.5.15 (Modified butterfly algorithm with identity permutation)**.** *Under the same assumptions of Theorem 6.5.9, we have:*

$$\|\mathbf{A} - \prod_{\ell=1}^{N} \mathbf{X}_\ell\|_F \leq \sqrt{2 \times 3^{N-2} - 1} \quad E^{\Theta}(\mathbf{A}), \tag{6.36}$$

Proof. It is obvious using Theorem 6.5.14, Corollary 6.5.8 and the equality:

$$3^{N-2} + 2 \sum_{k=2}^{N-1} 3^{N-1-k} = 2 \times 3^{N-2} - 1$$

□

Because $\sqrt{2 \times 3^{N-2} - 1} \leq 2^{N-1} - 1, \forall N \geq 2$, Theorem 6.5.14 provide a tighter result than Theorem 6.5.9 for identity permutation σ. The full proof of Theorem 6.5.14 can be found in Section 6.7.11. Here, we only provide a proof sketch.

Proof sketch for Theorem 6.5.14. When σ is the identity permutation, there are two important observations:

1. Lines 11-13 are always skipped (because we hierarchically factorize $\mathbf{A}$ from left to right, as illustrated in Figure 6.2a).

2. After the orthonormalization at the $(\ell+1)$th iteration, $\mathbf{X}_{P_\ell}$ does not change anymore. This is true because after the ℓth iteration, $P_\ell = \{\ell\}$. Thus, it will not be factorized in line 15 anymore. Moreover, after the orthonormalization at the $(\ell+1)$th iteration, the submatrix $\mathbf{X}_\ell[R_P, P]$ in line 5, Algorithm 9 is already orthonormal by column. The QR-decomposition of $\mathbf{X}_\ell[R_P, P]$ returns $\mathbf{Q} = \mathbf{X}_\ell[R_P, P]$ and $\mathbf{R} = \mathbf{I}$ an identity matrix. Thus, the value of $\mathbf{X}_\ell$ remains the same.

These observations allow us to improve the estimation in (6.30) to:

$$R_\ell^2 = R_{\ell-1}^2 + \|\mathbf{P}_{\ell-1} - \mathbf{P}_\ell\|_F^2$$

while (6.30) is a simple triangle inequality. Applying (6.33) and the inequality $(a+b)^2 \leq 2(a^2+b^2)$, we improve relation between R_ℓ and $R_{\ell-1}$ to:

$$R_\ell^2 \leq 3R_{\ell-1}^2 + 2E_s^\Theta(\mathbf{A})^2, \forall \ell \geq 2$$

Using $R_1 = E_1^\theta(\mathbf{A})$ (since the first factorization is precisely the matrix factorization problem in Definition 6.5.6 with $s = 1$), we deduce the bound in Equation (6.36). □

Applying Corollaries 6.5.11 and 6.5.15 to concrete chainable Θ gives us the following table:

Table 6.2: The approximation ratio C_N of Algorithm 8 with several chainable sequences of DB parameters Θ in Table 6.1.

Parameterization	**Size**	$\lvert\Theta\rvert$	C_N **of Corollary 6.5.11**	C_N **of Corollary 6.5.15**
Low rank matrix	$m \times n$	2	1	1
Monarch [DCS+22b]	$m \times n$	2	1	1
Square dyadic butterfly [DGE+19]	$n \times n$	$\log n$	$n/2 - 1 = O(n)$	$\frac{\sqrt{2}}{3}\sqrt{n^{\log 3} - 1} = O(n^{0.7925})$

6.6 Experimental results

In this section, we conduct several experiments to compare Algorithm 7, Algorithm 8, and other available algorithms in the literature.

Our first experiment is to factorize the Hadamard Transformation of size $2^N \times 2^N$ into $N = 9$ square dyadic butterfly factors (see Table 6.1) using Algorithm 7, Algorithm 8, a gradient-based method [DGE+19] and an alternating least square algorithm (ALS) [LRC+21]. All algorithms are implemented by Pytorch. In the following, we provide the descriptions for the candidates:

1. Algorithm 7 and Algorithm 8: We use $P = \{1, \ldots, N-1\}$ (left-to-right factorization). In line 4 of Algorithm 5, we choose $\mathbf{H} = \mathbf{U}[:, [\![|P|]\!]]\mathbf{D}^{1/2}, \mathbf{V} = \mathbf{D}^{1/2}\mathbf{X}[:, [\![|P|]\!]]$ where $\mathbf{U}\mathbf{D}\mathbf{V}^\top = \mathbf{A} \odot S_i$ and $\mathbf{U}[:, [\![|P|]\!]]$ is the submatrix corresponding to the first $|P|$ columns of $\mathbf{U}$.

2. Gradient-based method [DGE+19]: Thanks to the parameterization given by Θ, this method uses (variants of) gradient descent to optimize the coefficients inside the support constraints of butterfly factors. We use the protocol described in [DGE+19]: first, we perform a iterations of ADAM [GBC16, Chapter 7], then followed by b iterations of L-BFGS [NW06, Section 7.2]. In this experiment, we choose $a = 100, b = 20$ to ensure the convergence. Note that if the gradient norm is smaller than 10^{-7} (default chosen by Pytorch), L-BFGS will terminate.

3. Alternating least square: In each iteration, for every factor, we optimize its coefficients while fixing the others. Therefore, each iteration requires us to solve a sequence of linear regression problems. We reuse the ALS implementation of [LRC+21].

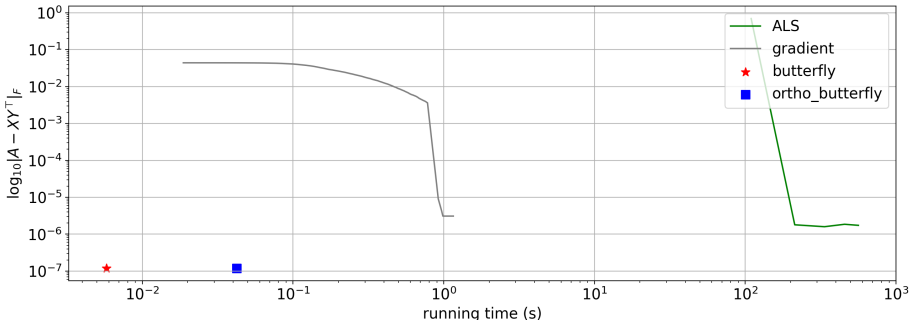


Figure 6.3: Comparison between projection methods with the factorization of the $2^9 \times 2^9$ Hadamard Transform.

The (relative) approximation errors (i.e. $\|\mathbf{A} - \mathbf{X}\|_F / \|\mathbf{A}\|_F$ and running time of four algorithms are illustrated in Figure 6.3. It can be seen that all methods can recover the exact factorization of the Hadamard Transform, up to machine precision (our implementation in Pytorch uses 32-bit float number format). In terms of running time, Algorithm 7 and Algorithm 8 are faster than the other algorithms, with gains of several orders of magnitude. Also in this setting, Algorithm 7 is also faster than Algorithm 8 since it does not perform the orthonormalization steps.

Our second experiment studies how well Algorithm 7 and Algorithm 8 scale with large matrices, different Θ and different permutation σ. In this setting, we choose the matrix $\mathbf{A}$ of size $2^L \times 2^L, L \in \{7, 8, \ldots, 13\}$. We fix the number of factors $N = 4$. For each size $2^L \times 2^L$, we generate a $(r, \ldots, r)$-*compatible* (cf. Definition 6.3.10) sequence of N DB parameters for $r \in \{1, 2, 4\}$. For each r, we choose Θ so that the total number of parameters $\mathcal{P}(\Theta)$ is minimized. This simulates a practical setting in deep learning: one would like to replace a (large) matrix with the product of (deformable) butterfly factors. The parameter r control the expressivity of $\mathcal{B}^\Theta$: the larger is r, the more expressive is $\mathcal{B}^\Theta$. Finally, among all possible chains, it is natural to choose one whose number of parameters is the smallest.

The matrix $\mathbf{A}$ is generated as: $\mathbf{X} + \mathbf{E}$ where $\mathbf{X} = \mathbf{X}_1 \ldots \mathbf{X}_N$ and the coefficients of $\mathbf{X}_i$ are i.i.d generated from $\mathcal{N}(0, 1)$; $\mathbf{E}$ is an i.i.d centered Gaussian matrix with the standard deviation 0.1. The permutation matrices σ_1 and σ_2 are those of Figure 6.2a) and b), respectively. We use exactly the same implementation of Algorithm 7 and Algorithm 8 in the previous experiment. For each N, r, we perform the two algorithms 10 times.

Figure 6.4 illustrates the running time and relative error approximation of Algorithm 7 and Algorithm 8. It can be seen that in the regime of very large matrices ($2^{13} \times 2^{13}$), the difference in running time between the two algorithms is negligible. The orthonormalization steps, thus, are not the bottleneck. In terms of error approximation, we observe that with σ_1, Algorithm 7 and Algorithm 8 are practically the same (but Algorithm 8 has to perform orthonormalization steps in addition). The interesting case is the factorization with σ_2. We can see that orthonormalization steps seem to do their job well since the error approximation of Algorithm 8 is consistently smaller than Algorithm 7 in all cases. Except for Algorithm 7 with σ_2, other variants consistently obtain factors whose products yield relative approximation errors smaller than $\|\mathbf{E}\|_F / \|\mathbf{A}\|_F$, the level of noise that we add to the matrix $\mathbf{X}$.

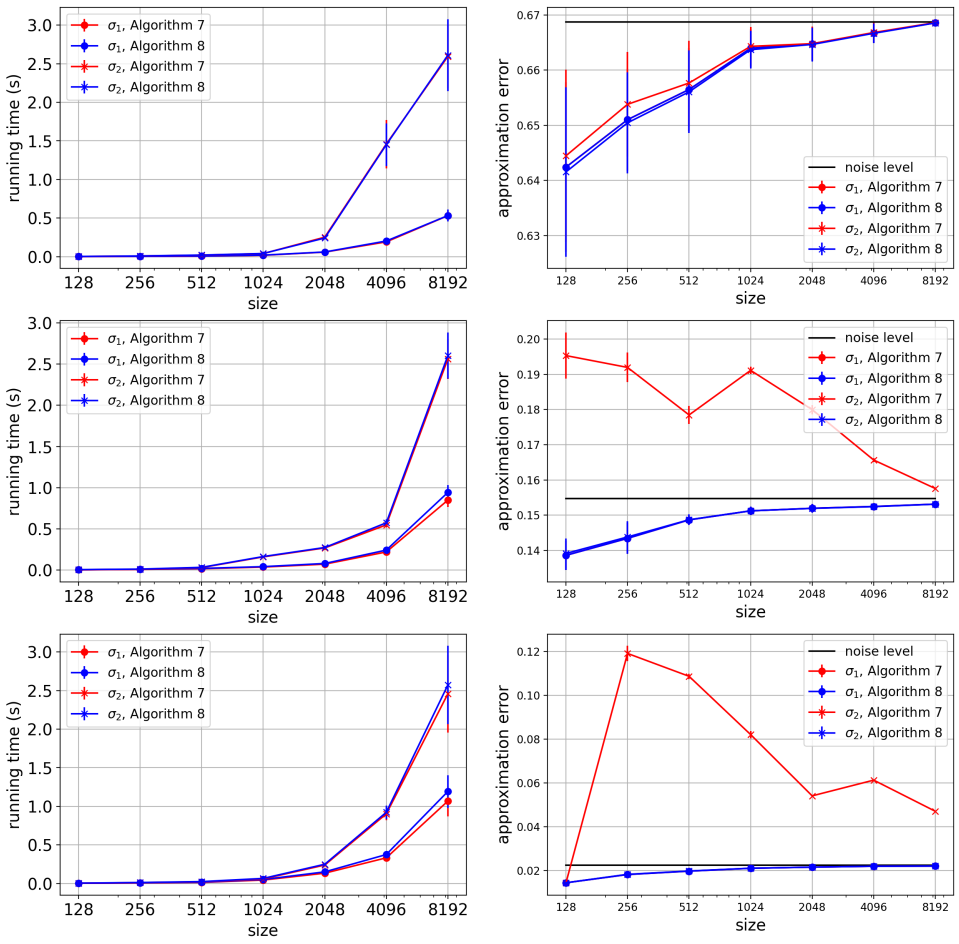


Figure 6.4: Comparison between Algorithm 7 and Algorithm 8with two permutations σ_1, σ_2. The error bar for each curve is the standard variation of either its running time or approximation error of 10 runs.

6.7 Postponed proofs for results in Chapter 6

In this section, we provide proof for several results that are not yet proved. To ease the reading when it comes to chainable sequences of DB parameters Θ, we will use the following short-hands:

Definition 6.7.1. Given a chainable sequence of DB parameters $\Theta := (\theta_\ell)_{\ell=1}^N$, define:

1. $\mathbf{S}_\ell^\Theta$ (or simply $\mathbf{S}_\ell$) $:= \mathbf{S}_{\theta_\ell} = \mathbf{S}_{(a_\ell, b_\ell, c_\ell, d_\ell)}$.

2. $\mathbf{S}^{\Theta}_{k,\ell}$ (or $\mathbf{S}_{k,\ell}$) $:= \mathbf{S}_{(\theta_k * \ldots * \theta_\ell)}, \forall k \leq \ell$.

3. $\theta_{k,\ell} = \theta_k * \ldots * \theta_\ell$.

4. $\mathcal{F}^i := \mathcal{F}^{\theta_i}$.

5. $\mathcal{F}^{k,\ell} := \mathcal{F}^{\theta_{k,\ell}}$.

We make a convention that $\mathbf{S}_{1,0} = \mathbf{I}_{a_1 b_1 d_1}$ and $\mathbf{S}_{N+1,N} = \mathbf{I}_{a_N c_N d_N}$.

As a reminder, the term *block* refers to the large block matrix on the diagonal, while the term *sub-block* refers to the small diagonal blocks inside each block. Readers might want to see the illustrations in Figure 6.1 for a better visualization.

We introduce a technical lemma which is then used multiple times in this section.

Lemma 6.7.2. *Consider support constraints* $(\mathbf{L}, \mathbf{R})$ *satisfying the conditions in Theorem 6.2.3, for all* $(\mathbf{X}, \mathbf{Y})$ *such that* $\text{supp}(\mathbf{X}) \subseteq \mathbf{L}, \text{supp}(\mathbf{Y}) \subseteq \mathbf{R}$*, we have:*

$$(\mathbf{XY})[R_P, C_P] = \mathbf{X}[R_P, P]\mathbf{Y}[P, C_P], \forall P \in \mathcal{P}(\mathbf{L}, \mathbf{R}). \tag{6.37}$$

where R_P, C_P *are defined as in Definition 7.6.4.*

Proof. For any pair of matrices $(\mathbf{X}, \mathbf{Y})$ (that do not necessarily satisfy the support constraints $(\mathbf{L}, \mathbf{R})$), we have:

$$\mathbf{XY} = \sum_{i=1}^{r} \mathbf{X}[:, i]\mathbf{Y}[i, :] = \sum_{P \in \mathcal{P}(\mathbf{L}, \mathbf{R})} \mathbf{X}[:, P]\mathbf{Y}[P, :].$$

If $(\mathbf{L}, \mathbf{R})$ satisfies the conditions of Theorem 6.2.3, then for any $\tilde{P} \neq P, \tilde{P} \in \mathcal{P}(\mathbf{L}, \mathbf{R})$, we have: $(\mathbf{X}[:, \tilde{P}]\mathbf{Y}[\tilde{P}, :])[R_P, C_P] = \mathbf{0}$ since $\text{supp}((\mathbf{X}[:, \tilde{P}]\mathbf{Y}[\tilde{P}, :])) \subseteq \mathbf{U}_{\tilde{P}}$ and $\mathbf{U}_P$ and $\mathbf{U}_{\tilde{P}}$ are disjoint (due to our assumption). Hence:

$$(\mathbf{XY})[R_P, C_P] = (\mathbf{X}[:, P]\mathbf{Y}[P, :])[R_P, C_P] = \mathbf{X}[R_P, P]\mathbf{Y}[P, C_P].$$

□

6.7.1 Proof for Theorem 6.2.3

The following proof of Theorem 6.2.3 is mainly taken from the proof of Theorem 5.2.3. We do an additional derivation for the infimum value of (6.1.FSMF).

Proof for Theorem 6.2.3. In this proof, we use the simple shorthand $\mathcal{P}$ for $\mathcal{P}(\mathbf{L}, \mathbf{R})$. Define $\overline{\mathbf{U}_\mathcal{P}} := (\llbracket m \rrbracket \times \llbracket n \rrbracket) \setminus \left(\bigcup_{P \in \mathcal{P}} \mathbf{U}_P\right)$. Because $\text{supp}(\mathbf{X}[:, P]\mathbf{Y}[P, :]) \subseteq \mathbf{U}_P, \forall P \in \mathcal{P}$, we have: $\text{supp}(\mathbf{XY}) \subseteq \bigcup_{P \in \mathcal{P}} \mathbf{U}_P$. Thus, $(\mathbf{XY}) \odot \overline{\mathbf{U}_\mathcal{P}} = \mathbf{0}$. According to Equation (5.5), the objective function $\|\mathbf{A} - \mathbf{XY}\|_F^2$ can be re-written as:

$$\|\mathbf{A} - \mathbf{XY}\|_F^2 = \left(\sum_{P \in \mathcal{P}} \|\mathbf{A}[R_P, C_P] - \mathbf{X}[R_P, P]\mathbf{Y}[P, C_P]\|^2\right) + \|\mathbf{A} \odot \overline{\mathbf{U}_\mathcal{P}}\|^2 \tag{6.38}$$

Therefore, if we ignore the constant term $\|\mathbf{A} \odot \bar{\mathbf{U}}_\mathcal{P}\|^2$, the function $L(\mathbf{X}, \mathbf{Y})$ is decomposed into a sum of functions $\|\mathbf{A}[R_P, C_P] - \mathbf{X}[R_P, P]\mathbf{Y}[P, C_P]\|^2$, which are instances of low-rank

matrix approximation problem. Since all the optimized parameters are $\{(\mathbf{X}[R_P, P], \mathbf{Y}[P, C_P])\}_{P\in\mathcal{P}}$, an optimal solution of L is $\{(\mathbf{X}^\star[R_P, P], \mathbf{Y}^\star[P, C_P])\}_{P\in\mathcal{P}}$, where $(\mathbf{X}^\star[R_P, P], \mathbf{Y}^\star[P, C_P])$ is a minimizer of $\|\mathbf{A}_{R_P,C_P} - \mathbf{X}[R_P, P]\mathbf{Y}[P, C_P]\|^2$ which is computed efficiently using a truncated SVD. Since the blocks associated to distinct P are disjoint, these SVDs can be performed blockwise, in any order, and even in parallel. That is why Algorithm 5 yields optimal solutions.

To deduce the value of the infimum, it is sufficient to use the fact that the distance between a matrix $\mathbf{A}$ to the set of matrices of rank at most k is given by:

$$\|\mathbf{A}\|_F^2 - \sum_{j=1}^{k} \sigma_j^2(\mathbf{A}). \tag{6.39}$$

Applying this equality into (6.38) gives us the infimum of (6.1.FSMF):

$$\begin{aligned}
\inf \|\mathbf{A} - \mathbf{XY}\|_F^2 &= \inf \left(\sum_{P\in\mathcal{P}} \|\mathbf{A}[R_P, C_P] - \mathbf{X}[R_P, P]\mathbf{Y}[P, C_P]\|^2 \right) + \|\mathbf{A} \odot \overline{\mathbf{U}_\mathcal{P}}\|^2 \\
&= \left(\sum_{P\in\mathcal{P}} \inf \|\mathbf{A}[R_P, C_P] - \mathbf{X}[R_P, P]\mathbf{Y}[P, C_P]\|^2 \right) + \|\mathbf{A} \odot \overline{\mathbf{U}_\mathcal{P}}\|^2 \\
&\overset{(6.39)}{=} \left(\sum_{P\in\mathcal{P}} \|\mathbf{A}[R_P, C_P]\|_F^2 - \sum_{j=1}^{|P|} \sigma_j^2(\mathbf{A}[R_P, C_P]) \right) + \|\mathbf{A} \odot \overline{\mathbf{U}_\mathcal{P}}\|^2 \\
&= \underbrace{\left(\sum_{P\in\mathcal{P}} \|\mathbf{A}[R_P, C_P]\|_F^2 \right) + \|\mathbf{A} \odot \overline{\mathbf{U}_\mathcal{P}}\|^2}_{\|\mathbf{A}\|_F^2} - \sum_{P\in\mathcal{P}} \sum_{j=1}^{|P|} \sigma_j^2(\mathbf{A}[R_P, C_P]).
\end{aligned}$$

□

An immediate corollary of Theorem 5.2.3 and Theorem 6.2.3 is:

Corollary 6.7.3. *Given* $(\mathbf{L}, \mathbf{R})$ *satisfying the conditions in Theorem 6.2.3, define* $\mathcal{M}_{\mathbf{L},\mathbf{R}} := \{\mathbf{XY} \mid \mathrm{supp}(\mathbf{X}) \subseteq \mathbf{L}, \mathrm{supp}(\mathbf{Y}) \subseteq \mathbf{R}\}$. *The set* $\mathcal{M}_{\mathbf{L},\mathbf{R}}$ *is equal to the intersection of two sets* $\mathcal{Z}_1$ *and* $\mathcal{Z}_2$*, given by:*

$$\begin{aligned}
\mathcal{Z}_1 &:= \{\mathbf{A} \mid \mathrm{rank}(\mathbf{A}[R_P, C_P]) \leq |P|, \forall P \in \mathcal{P}(\mathbf{L}, \mathbf{R})\} \\
\mathcal{Z}_2 &:= \{\mathbf{A} \mid \mathrm{supp}(\mathbf{A}) \subseteq \mathbf{U}\}, \qquad \mathbf{U} := \bigcup_{P\in\mathcal{P}(\mathbf{L},\mathbf{R})} (R_P \times C_P)
\end{aligned}$$

Proof. Thanks to Theorem 5.2.3, a matrix $\mathbf{A} \in \mathcal{M}_{\mathbf{L},\mathbf{R}}$ if and only if:

$$\|\mathbf{A}\|_F^2 - \sum_{P\in\mathcal{P}(\mathbf{L},\mathbf{R})} \sum_{j=1}^{|P|} \sigma_j^2(\mathbf{A}[R_P, C_P]) = 0.$$

Moreover, due to the property of eigenvalues, which is:

$$\|\mathbf{A}\|_F^2 = \sum_j \sigma_j^2(\mathbf{A}), \tag{6.40}$$

and the rank one representatives are either disjoint or overlapping pairwise, we have:

$$\begin{aligned}
0 &= \|\mathbf{A}\|_F^2 - \sum_{P\in\mathcal{P}(\mathbf{L},\mathbf{R})}\sum_{j=1}^{|P|}\sigma_j^2(\mathbf{A}[R_P,C_P]) \\
&= \|\mathbf{A}\odot\bar{\mathbf{U}}\|_F^2 + \sum_{P\in\mathcal{P}(\mathbf{L},\mathbf{R})}\left(\|\mathbf{A}[R_P,C_P]\|_F^2 - \sum_{j=1}^{|P|}\sigma_j^2(\mathbf{A}[R_P,C_P])\right) \\
&= \|\mathbf{A}\odot\bar{\mathbf{U}}\|_F^2 + \sum_{P\in\mathcal{P}(\mathbf{L},\mathbf{R})}\left(\sum_{j=|P|+1}^{\min|R_P|,|C_P|}\sigma_j^2(\mathbf{A}[R_P,C_P])\right),
\end{aligned}$$

where $\bar{\mathbf{U}}$ is the complement of $\mathbf{U}$. Since all the terms in the RHS are non-negative, the equality holds if and only if all terms are equal to zero. Consider them term by term gives us:

1. $\|\mathbf{A}\odot\bar{\mathbf{U}}\|_F^2 = 0$: if and only if $\text{supp}(\mathbf{A})\subseteq\mathbf{U}$.
2. $\sum_{j=|P|+1}^{\min|R_P|,|C_P|}\sigma_j^2(\mathbf{A}[R_P,C_P]) = 0$: if and only if $\text{rank}(\mathbf{A}[R_P,C_P])\leq r_i$.

That concludes the proof. □

6.7.2 Proof for Lemma 6.3.15

Proof. Denote $\theta_\ell = (a_\ell, b_\ell, c_\ell, d_\ell)$ for $\ell = 1,2,3$. Let us show the q_1-chainability between θ_1 and $\theta_2 * \theta_3$. By definition of $*$, we have:

$$(\tilde{a},\tilde{b},\tilde{c},\tilde{d}) = \tilde{\theta} := \theta_2 * \theta_3 = \left(a_2, \frac{b_2d_2}{d_3}, \frac{a_3c_3}{a_2}, d_3\right) \tag{6.41}$$

It is sufficient to verify three conditions of Definition 6.3.7:

1. We have: $a_1c_1/\tilde{a} = a_1c_1/a_2$ and $\tilde{b}\tilde{d}/d_1 = b_2d_2/d_1$. Since θ_1 and θ_2 are q_1-chainable, we also have: $a_1c_1/a_2 = b_2d_2/d_1 = q_1 \in \mathbb{N}$. Thus $a_1c_1/\tilde{a} = \tilde{b}\tilde{d}/d_1 \in \mathbb{N}$.
2. Since θ_1,θ_2 are chainable and $a_2 = \tilde{a}$, we have $a_1|\tilde{a}$.
3. Since θ_1,θ_2 and θ_2,θ_3 are chainable, we have: $d_2|d_1$ and $d_3|d_2$. Thus, $\tilde{d}|d_1$ (because $\tilde{d} = d_3$).

Thus, θ_1 and $(\theta_2 * \theta_3)$ are q_1-chainable. Similarly, $(\theta_1 * \theta_2)$ and θ_3 are also q_2-chainable. Moreover, computing explicitly $\theta_1 * (\theta_2 * \theta_3)$ gives:

$$\theta_1 * (\theta_2 * \theta_3) = \left(a_1, \frac{b_1d_1}{d_3}, \frac{a_3c_3}{a_1}, d_3\right)$$

The computing of $(\theta_1 * \theta_2) * \theta_3$ yields the same DB parameter, which ends the proof. □

6.7.3 The explicit formula for $(\theta_1 * \ldots * \theta_N)$

Corollary 6.7.4. *For $\Theta = (\theta_\ell)_{\ell=1}^N = (a_\ell, b_\ell, c_\ell, d_\ell)_{\ell=1}^N$ a chainable sequence of DB parameters:*

$$\theta_1 * \ldots * \theta_N = \left(a_1, \frac{b_1 d_1}{d_N}, \frac{a_N c_N}{a_1}, d_N\right). \tag{6.42}$$

Proof. The proof is carried out by induction. Consider:

1. Base case: If $N = 2$, the result is trivial thanks to Definition 6.3.13.

2. Assume that the statement of this corollary holds for any chainable sequence Θ whose length $|\Theta| \leq N$. We prove that it is still valid for $\Theta = (\theta_\ell)_{\ell=1}^{N+1}$ of length $N+1$. Indeed, by induction hypothesis, we have:

$$\theta_1 * \ldots * \theta_N = \left(a_1, \frac{b_1 d_1}{d_N}, \frac{a_N c_N}{a_1}, d_N\right)$$

Therefore,

$$\begin{aligned}
\theta_1 * \ldots * \theta_N * \theta_{N+1} &= \left(a_1, \frac{b_1 d_1}{d_N}, \frac{a_N c_N}{a_1}, d_N\right) * (a_{N+1}, b_{N+1}, c_{N+1}, d_{N+1}) \\
&= \left(a_1, \frac{b_1 d_1 d_N}{d_N d_{N+1}}, \frac{a_{N+1} c_{N+1}}{a_1}, d_{N+1}\right) \\
&= \left(a_1, \frac{b_1 d_1}{d_{N+1}}, \frac{a_{N+1} c_{N+1}}{a_1}, d_{N+1}\right).
\end{aligned}$$

That concludes the proof. □

6.7.4 Proof for Lemma 6.3.22

Proof. Because Θ is redundant, there exists $\ell \in [\![N-1]\!]$ such that the q_ℓ-chainable pair $(\theta_\ell, \theta_{\ell+1})$ is redundant. It implies that $q_\ell > \min(b_\ell, c_{\ell+1})$.

The proof of this lemma consists of proving that for $\Theta' := (\theta_1, \ldots, \theta_{\ell-1}, \theta_\ell * \theta_{\ell+1}, \theta_{\ell+2}, \ldots, \theta_N)$:

1. Θ' is chainable.
2. $\|\Theta'\|_0 < \|\Theta\|_0$.
3. $\mathcal{B}^{\Theta'} = \mathcal{B}^{\Theta}$.

The proof of each statement is given as follows:

1. Θ' is chainable: This is a direct corollary of Lemma 6.4.2.

2. $\|\Theta'\|_0 < \|\Theta\|_0$: This is equivalent to show that $\|\theta_\ell * \theta_{\ell+1}\|_0 < \|\theta_\ell\|_0 + \|\theta_{\ell+1}\|_0$. The explicit values of the two terms are given by:

$$\begin{aligned}
\|\theta_\ell\|_0 + \|\theta_{\ell+1}\|_0 &= a_\ell b_\ell c_\ell d_\ell + a_{\ell+1} b_{\ell+1} c_{\ell+1} d_{\ell+1} \\
&= a_{\ell+1} q_\ell b_\ell d_\ell + a_{\ell+1} d_\ell q_\ell c_{\ell+1} \\
&= a_{\ell+1} d_\ell (q_\ell d_\ell + q_\ell c_{\ell+1}) \\
\|\theta_i * \theta_{\ell+1}\|_0 &= a_{\ell+1} d_\ell b_\ell c_{\ell+1}
\end{aligned}$$

Since $q_\ell > \min(b_\ell, c_{\ell+1})$, we have: $b_\ell c_{\ell+1} < q_\ell d_\ell + q_\ell c_{\ell+1}$. Therefore, $\|\theta_i * \theta_{\ell+1}\|_0 < \|\theta_\ell\|_0 + \|\theta_{\ell+1}\|_0$.

3. $\mathcal{B}^{\Theta'} = \mathcal{B}^{\Theta}$: it is sufficient to prove that:

$$\mathcal{F}^{\ell,\ell+1} = \{\mathbf{X}_\ell \mathbf{X}_{\ell+1} | \mathbf{X}_\ell \in \mathcal{F}^\ell, \mathbf{X}_{\ell+1} \in \mathcal{F}^{\ell+1}\},$$

where $\mathcal{F}^{\ell,\ell+1} := \mathcal{F}^{\theta_\ell * \theta_{\ell+1}}$ is defined as in Definition 6.7.1. It is easy to see that $\{\mathbf{X}_\ell \mathbf{X}_{\ell+1} | \mathbf{X}_\ell \in \mathcal{F}^\ell, \mathbf{X}_{i+\ell} \in \mathcal{F}^{\ell+1}\} \subseteq \mathcal{F}^{\ell,\ell+1}$ due to Corollary 6.3.14.

To prove the other direction, we notice that the maximum support of a matrix in $\mathcal{F}^{(\ell,\ell+1)}$ is $\mathbf{S}_{\ell,\ell+1} = \text{supp}(\mathbf{S}_\ell \mathbf{S}_{\ell+1}) \subseteq \bigcup_{P \in \mathcal{P}(\mathbf{S}_\ell, \mathbf{S}_{\ell+1})} \mathbf{U}_P$ (cf. Proposition 6.3.11). Therefore, if a matrix $\mathbf{A} = \mathbf{X}_\ell \mathbf{X}_{\ell+1}$ for some $\mathbf{X}_\ell \in \mathcal{F}^\ell, \mathbf{X}_{\ell+1} \in \mathcal{F}^{\ell+1}$, it is equivalent to:

$$\mathbf{A}[R_P, C_P] = (\mathbf{X}_\ell \mathbf{X}_{\ell+1})[R_P, C_P], \forall P \in \mathcal{P}(\mathbf{S}_\ell, \mathbf{S}_{\ell+1})$$

Moreover, since $(\theta_\ell, \theta_{\ell+1})$ is q_ℓ-chainable, $(\mathbf{S}_\ell, \mathbf{S}_{\ell+1})$ satisfies the condition of Lemma 6.7.2. Therefore,

$$(\mathbf{X}_\ell \mathbf{X}_{\ell+1})[R_P, C_P] = \mathbf{X}_\ell[R_P, P] \mathbf{X}_{\ell+1}[P, C_P], \forall P \in \mathcal{P}(\mathbf{S}_\ell, \mathbf{S}_{\ell+1})$$

Since $(\theta_\ell, \theta_{\ell+1})$ is q_ℓ-chainable, $|P| = q_\ell, \forall P \in \mathcal{P}(\mathbf{S}_\ell, \mathbf{S}_{\ell+1})$. Therefore, for any matrix $\mathbf{A} \in \mathcal{F}^{\ell,\ell+1}$, we choose $\mathbf{X}_k \in \mathcal{F}^k, k \in \{\ell, \ell+1\}$ such that:

$$\mathbf{X}_\ell[R_P, P] \mathbf{X}_{\ell+1}[P, C_P] = \mathbf{A}[R_P, C_P].$$

This is always possible because $|R_P| = b_\ell, |C_P| = c_{\ell+1}$ and $\min(|R_P|, |C_P|) < |P| = q_\ell$ (which is our assumption). Since this holds for any $\mathbf{A} \in \mathcal{F}^{\ell,\ell+1}$, we deduce that: $\mathcal{F}^{\ell,\ell+1} \subseteq \{\mathbf{X}_\ell \mathbf{X}_{\ell+1} | \mathbf{X}_\ell \in \mathcal{F}^\ell, \mathbf{X}_{\ell+1} \in \mathcal{F}^{\ell+1}\}$.

That concludes the proof. □

6.7.5 Proof for Lemma 6.4.1

Proof. First, for any DB parameter θ, due to the definition of $\mathbf{S}_\theta$, the support of columns/rows of $\mathbf{S}_\theta$ is pairwise disjoint or identical. Therefore, any pair of elements of $\varphi(\mathbf{S}_{\theta_1}, \mathbf{S}_{\theta_2})$ is either identical (if their corresponding column and row supports coincide) or disjoint (otherwise). That concludes the proof.

Note that $\mathbf{S}_{\theta_1} \mathbf{S}_{\theta_2} = \sum_{\mathbf{U}_i \in \varphi(\mathbf{S}_{\theta_1}, \mathbf{S}_{\theta_2})} \mathbf{U}_i$ (this is simply the rank one decomposition of matrix multiplication). By Proposition 6.3.11:

$$\sum_{\mathbf{U}_i \in \varphi(\mathbf{S}_{\theta_1}, \mathbf{S}_{\theta_2})} \mathbf{U}_i = \mathbf{S}_{\theta_1} \mathbf{S}_{\theta_2} = q \mathbf{S}_{(\theta_1 * \theta_2)}. \tag{6.43}$$

Since $\mathbf{U}_i$ are pairwise disjoint or identical, the above equation implies that each $\mathbf{U}_i$ needs to repeat exactly q times. Therefore, for $P \in \mathcal{P}(\mathbf{S}_{\theta_1}, \mathbf{S}_{\theta_2})$, we have: $|P| = q$.

Finally, R_P and C_P are simply the support constraints of the ℓth column $\mathbf{S}_{\theta_1}$ and ℓth row of $\mathbf{S}_{\theta_2}$ respectively, for any $\ell \in P$. Therefore, $|R_P| = b_1, |C_P| = c_2$. □

6.7.6 Proof for Lemma 6.4.2

Proof. Thanks to Corollary 6.7.4, we have:

$$\begin{aligned}(a,b,c,d) := \theta_l * \ldots * \theta_s &= \left(a_l, \frac{b_l d_l}{d_s}, \frac{a_s c_s}{a_l}, d_s\right) \\ (a',b',c',d') := \theta_{s+1} * \ldots * \theta_r &= \left(a_{s+1}, \frac{b_{s+1} d_{s+1}}{d_r}, \frac{a_r c_r}{a_{s+1}}, d_r\right)\end{aligned} \tag{6.44}$$

Veryfying three conditions of q-chainability gives us:

1. First condition: $ac/a' = a_s c_s / a_{s+1} = q_s$, $b'd'/d = b_{s+1}d_{s+1}/d_s = q_s$ (because (θ_s, θ_{s+1}) is q_s-chainable.
2. Second and third conditions: Since $a_\ell | a_{\ell+1}$ and $d_{\ell+1} | d_\ell, \forall \ell = 1, \ldots, N-1$, we have: $a_l | a_{s+1}$ and $d_r | d_s$.

□

6.7.7 Proof for Lemma 6.5.2

Proof. The proof is based on the explicit formulas of $\theta_l * \ldots * \theta_s$ and $\theta_{s+1} * \ldots * \theta_r$, which are given by:

$$\begin{aligned}(a,b,c,d) := \theta_l * \ldots * \theta_s &= \left(a_l, \frac{b_l d_l}{d_s}, \frac{a_s c_s}{a_l}, d_s\right) \\ (a',b',c',d') := \theta_{s+1} * \ldots * \theta_r &= \left(a_{s+1}, \frac{b_{s+1} d_{s+1}}{d_r}, \frac{a_r c_r}{a_{s+1}}, d_r\right)\end{aligned} \tag{6.45}$$

We just need to verify that $q_s = b'd'/d = ac/a' \leq \min(b, c')$ because they are already q_s-chainable, as proved in Lemma 6.4.2.

We will prove that $q_s = b'd'/d \leq b$. A similar argument will yield $q_s = ac/a' \leq c'$. As a consequence, $q_s \leq \min(b, c)$ and the proof is concluded.

In fact, since $b'd'/d = b_{s+1}d_{s+1}/d_s, b = b_1 d_1/d_s$, proving $b'd'/d \leq b$ is equivalent to proving $b_{s+1}d_{s+1} \leq b_1 d_1$. Because Θ is non-redundant, for all $\ell \in [\![N-1]\!]$, we have: $q_\ell = b_{\ell+1}d_{\ell+1}/d_\ell \leq b_\ell$. Therefore, $b_{\ell+1}d_{\ell+1} \leq b_\ell d_\ell, \forall \ell \in [\![N-1]\!]$. Thus,

$$b_{s+1}d_{s+1} \leq \ldots \leq b_1 d_1.$$

□

6.7.8 Orthonormal DB factors and the properties of Algorithm 9

This section is devoted to introduce introduce a notion of orthonormality for a DB factor. It provides us with formal definitions and technical tools to prove results involved Algorithm 9 (namely, Lemmas 6.5.1, 6.5.3 and 6.5.4). We start with the following definition.

Definition 6.7.5. Let θ be a DB parameter, and q be an integer, define:

1. $\mathcal{P}_c(\theta, q) := \mathcal{P}(\mathbf{S}_\theta, \mathbf{S}_{\tilde{\theta}})$ for any DB parameter $\tilde{\theta}$ such that $(\theta, \tilde{\theta})$ is q-chainable if such an $\tilde{\theta}$ exists.

2. $\mathcal{P}_r(\theta, q) := \mathcal{P}(\mathbf{S}_{\tilde{\theta}}, \mathbf{S}_\theta)$ for any DB parameter $\tilde{\theta}$ such that $(\tilde{\theta}, \theta)$ is q-chainable if such an $\tilde{\theta}$ exists.

Definition 6.7.5 is indeed well-defined. In fact, one needs to prove that $\mathcal{P}(\mathbf{S}_\theta, \mathbf{S}_{\tilde{\theta}})$ and $\mathcal{P}(\mathbf{S}_{\tilde{\theta}}, \mathbf{S}_\theta)$ do not depend on the choice of $\tilde{\theta}$. An illustration for Definition 6.7.5 is given in Figure 6.5.

We only prove that $\mathcal{P}_c(\theta, q)$ is well-defined. The same result for $\mathcal{P}_r(\theta, q)$ can be handled similarly. All we need to do is to prove the following lemma.

Lemma 6.7.6 (Well-definedness of $\mathcal{P}_c(\theta, q)$)**.** *If $\theta : (a, b, c, d)$ is a DB parameter such that there exists $\tilde{\theta}$ satisfying that $(\theta, \tilde{\theta})$ is q-chainable, then $\mathcal{P}(\mathbf{S}_\theta, \mathbf{S}_{\tilde{\theta}})$ does not depend on the choice of $\tilde{\theta}$ and its elements have the following form:*

$$\{k + tqd, k + d + tqd, \ldots, k + d(q-1) + tqd\}, \quad \forall k = 1, \ldots, d, \forall t = 0, \ldots, ca/q - 1. \tag{6.46}$$

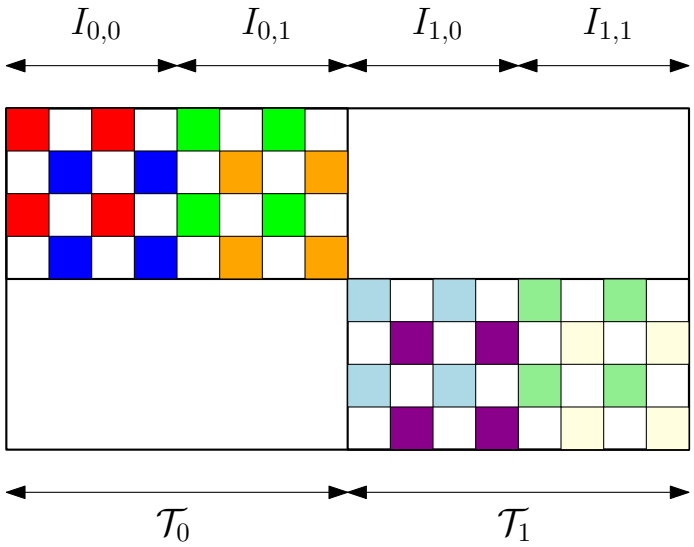


Figure 6.5: Illustration for $\mathcal{P}_c(\theta, q)$ with $\theta = (2, 2, 4, 2)$ and $q = 2$. Columns with same color belong to the same equivalence class. Illustration for $\mathcal{T}_t$ (cf. Equation (6.47)) and $I_{t,s}$ (cf. Equation (6.48)) are also provided.

Proof for Lemma 6.7.6. We remind readers of our terminology: *blocks* refers to the large blocks in the diagonal of size $bd \times cd$, and *sub-blocks* to refer the small ones of size $d \times d$ (see Figure 6.1). We assume that $\tilde{\theta} := (\tilde{a}, \tilde{b}, \tilde{c}, \tilde{d})$.

It is sufficient to prove that $\mathcal{P}_c(\theta, q)$ consists of elements of the form of (6.46) because these elements trivially do not depend on $\tilde{\theta}$. Remind that each block has size $bd \times cd$. The set of column indices corresponding to a block has the form:

$$\mathcal{T}_t := [\![1 + tcd, (t+1)cd]\!], \forall t = 0, \ldots, a - 1 \tag{6.47}$$

Apparently, columns in different blocks have disjoint supports. Consequently, they cannot be partitioned into the same equivalent class.

For each $\mathcal{T}_t$, we further decompose them into $c/q = \tilde{a}/a \in \mathbb{N}$ equal consecutive intervals of the forms:

$$I_{t,s} = [\![1 + sdq + tcd, (s+1)dq + tcd]\!], \forall s = 0, \ldots, c/q - 1 \tag{6.48}$$

An illustration of $\mathcal{T}_t$ and $I_{t,s}$ can be found in Figure 6.5.

We argue that indices in *different* $I_{t,s}$ cannot be in the same equivalent class. Indeed, consider a $\tilde{\theta}$ such that $(\theta, \tilde{\theta})$ is q-chainable, since $ac = q\tilde{a}$ (cf. Definition 6.3.7), the number of columns of each block of θ-DB factors is c/q times that of rows of each block of $\tilde{\theta}$-DB factors. It implies that indices in different $I_{t,s}$ belongs to different blocks of $\tilde{\theta}$-DB factors. Since the row supports of different blocks (of $\tilde{\theta}$-DB factors) are disjoint, the corresponding rank one contribution supports have to be disjoint between indices of different $I_{t,s}$. Thus, equivalence classes are subsets of $I_{t,s}$ for some $t, s \in \mathbb{N}$.

Finally, let's examine closely an $I_{t,s}$. To make it simple, we consider the first interval of the first block $I_{0,0} = [\![dq]\!]$. The columns of the same support in this interval repeat with frequency d (since each block is a block matrix whose sub-blocks are $d \times d$ diagonal matrices). Thus, $[\![dq]\!]$ can be partitioned exactly into d subsets (of size q) of indices whose columns share the same support. By Lemma 6.4.1, this must also be an equivalence class. Our following argument remains valid for other $I_{t,s}$ as well. It implies that each partition has the form as in Equation (6.46). □

Using Definition 6.7.5, we can define q-column/row-orthonormal DB factors as follow:

Definition 6.7.7 (q-column/row-orthonormal DB factors)**.** Given an integer q and a DB parameter θ such that $\mathcal{P}_c(\theta, q)$ (resp. $\mathcal{P}_r(\theta, q)$) is defined, a θ-DB factor $\mathbf{A}$ is *q-column-orthonormal* (resp. q-row-orthonormal) if the submatrix $\mathbf{A}[:, P]$ is orthonormal by column (resp. $\mathbf{A}[P, :]$ is orthonormal by row) for each $P \in \mathcal{P}_c(\theta, q)$ (resp. $P \in \mathcal{P}_r(\theta, q)$).

We refer to both q-column/row-orthonormal DB factors as *orthonormal DB factors*.

Remark 6.7.8. Another equivalent definition of q-column (resp. row)-orthonormal θ-DB factor is to require $\mathbf{A}[R_P, P]$ (resp. $\mathbf{S}[P, C_P]$) to be column (resp. row)-orthonormal (R_P, C_P are defined as in Definition 7.6.4). Indeed, the submatrix $\mathbf{A}[:, P]$ has the form: $\mathbf{A}[:, P] = \begin{pmatrix} \mathbf{A}[R_P, P] \\ \mathbf{0} \end{pmatrix}$ up to some row permutation. Thus, if $\mathbf{A}[:, P]$ is column-orthonormal, so is $\mathbf{A}[R_P, P]$ and vice-versa. The same argument applies for $\mathbf{A}[P, :]$ and $\mathbf{A}[P, C_P]$.

We are now ready to precisely describe what Algorithm 9 aims to achieve:

Lemma 6.7.9. *Let (θ_1, θ_2) be a pair of q-chainable DB parameters that is not redundant. For any $(\mathbf{X}_1, \mathbf{X}_2) \in \mathcal{F}^{\theta_1} \times \mathcal{F}^{\theta_2}$, Algorithm 9 will return:*

1. *If $t =$ `column`: a pair of factors $(\tilde{\mathbf{X}}_1, \tilde{\mathbf{X}}_2) \in \mathcal{F}^{\theta_1} \times \mathcal{F}^{\theta_2}$ such that $\tilde{\mathbf{X}}_1$ is q-column-orthonormal DB factor and $\tilde{\mathbf{X}}_1\tilde{\mathbf{X}}_2 = \mathbf{X}_1\mathbf{X}_2$;*
2. *If $t =$ `row`: a pair of factors $(\tilde{\mathbf{X}}_1, \tilde{\mathbf{X}}_2) \in \mathcal{F}^{\theta_1} \times \mathcal{F}^{\theta_2}$ such that $\tilde{\mathbf{X}}_2$ is q-row-orthonormal DB factor and $\tilde{\mathbf{X}}_1\tilde{\mathbf{X}}_2 = \mathbf{X}_1\mathbf{X}_2$.*

The proof of Lemma 6.7.9 is demonstrated in Section 6.7.9 altogether with its corollaries: Lemmas 6.5.1, 6.5.3 and 6.5.4. Below, we list several properties of DB factors, which will be used to prove for other remaining results.

Inner product preservation

The interest of introducing q-column/row-orthonormal DB-factors is that they can act similarly to classical column/row-orthonormal matrices. Roughly speaking, they preserve the inner product between DB-factors up to matrix multiplication operators under certain assumptions. It is shown in the following lemma:

Lemma 6.7.10. *Given a θ-DB factor $\mathbf{X}$ and a pairs of θ'-DB factors $(\mathbf{Y}_1, \mathbf{Y}_2)$ where (θ, θ') is q-chainable, if $\mathbf{X}$ is a q-column-orthonormal DB factor, then:*

$$\langle \mathbf{X}\mathbf{Y}_1, \mathbf{X}\mathbf{Y}_2 \rangle = \langle \mathbf{Y}_1, \mathbf{Y}_2 \rangle$$

where $\langle \cdot, \cdot \rangle$ denotes the usual Frobenius inner product of two matrices.

Proof. Since $\mathrm{supp}(\mathbf{X}\mathbf{Y}_i) \subseteq \cup_{P \in \mathcal{P}(\mathbf{S}_\theta, \mathbf{S}_{\theta'})} R_P \times C_P, i = 1, 2$, and $R_P \times C_P$ are pairwise disjoint (cf. Lemma 6.4.1), we have:

$$\langle \mathbf{X}\mathbf{Y}_1, \mathbf{X}\mathbf{Y}_2 \rangle = \sum_{P \in \mathcal{P}(\mathbf{S}_\theta, \mathbf{S}_{\theta'})} \langle (\mathbf{X}\mathbf{Y}_1)[R_P, C_P], (\mathbf{X}\mathbf{Y}_2)[R_P, C_P] \rangle \tag{6.49}$$

Moreover, $\mathbf{S}_{\theta'} = \bigcup_{P \in \mathcal{P}(\mathbf{S}_\theta, \mathbf{S}_{\theta'})} P \times C_P$ and $P \times C_P$ are pairwise disjoint (because they belong to different subsets of rows of $\mathbf{Y}_i, i = 1, 2$), we have:

$$\langle \mathbf{Y}_1, \mathbf{Y}_2 \rangle = \sum_{P \in \mathcal{P}(\mathbf{S}_\theta, \mathbf{S}_{\theta'})} \langle \mathbf{Y}_1[P, C_P], \mathbf{Y}_2[P, C_P] \rangle. \tag{6.50}$$

By Lemma 6.7.2, we have:

$$(\mathbf{X}\mathbf{Y}_i)[R_P, C_P] = \mathbf{X}[R_P, P]\mathbf{Y}_i[P, C_P], \forall P \in \mathcal{P}(\mathbf{S}_\theta, \mathbf{S}_{\theta'}), i = 1, 2.$$

Since we assume that $\mathbf{X}$ is a q-column-orthonormal DB-factor, $\mathbf{X}[R_P, P]$ is orthonormal by column (see Remark 6.7.8), i.e., $\mathbf{X}[R_P, P]^\top \mathbf{X}[R_P, P] = \mathbf{I}$ - the identity matrix. Thus,

$$\begin{aligned}
\langle (\mathbf{X}\mathbf{Y}_1)[R_P, C_P], (\mathbf{X}\mathbf{Y}_2)[R_P, C_P] \rangle &= \langle \mathbf{X}[R_P, P]\mathbf{Y}_1[P, C_P], \mathbf{X}[R_P, P]\mathbf{Y}_2[P, C_P] \rangle \\
&= \mathrm{Tr}(\mathbf{Y}_1[P, C_P]^\top \mathbf{X}[R_P, P]^\top \mathbf{X}[R_P, P]\mathbf{Y}_2[P, C_P]) \\
&= \mathrm{Tr}(\mathbf{Y}_1[P, C_P]^\top \mathbf{Y}_2[P, C_P]) \\
&= \langle \mathbf{Y}_1[P, C_P], \mathbf{Y}_2[P, C_P] \rangle
\end{aligned} \tag{6.51}$$

Using this equality, we obtain:

$$\langle \mathbf{X}\mathbf{Y}_1, \mathbf{X}\mathbf{Y}_2 \rangle \overset{(6.49)+(6.51)}{=} \sum_{P \in \mathcal{P}(\mathbf{S}_\theta, \mathbf{S}_{\theta'})} \langle \mathbf{Y}_1[P, C_P], \mathbf{Y}_2[P, C_P] \rangle \overset{(6.50)}{=} \langle \mathbf{Y}_1, \mathbf{Y}_2 \rangle$$

□

A similar result to Lemma 6.7.10, but for q-row-orthonormal DB factors is:

Lemma 6.7.11. *Given a θ-DB factor $\mathbf{Y}$ and a pairs of θ'-DB factors $(\mathbf{X}_1, \mathbf{X}_2)$ where (θ', θ) is q-chainable, if $\mathbf{Y}$ is a q-row-orthonormal DB factor, then:*

$$\langle \mathbf{X}_1\mathbf{Y}, \mathbf{X}_2\mathbf{Y} \rangle = \langle \mathbf{X}_1, \mathbf{X}_2 \rangle$$

Combining Lemma 6.7.10 and Lemma 6.7.11, we have:

Lemma 6.7.12. *Given three DB parameters $\theta_i, i = 1, 2, 3$ where (θ_1, θ_2) and (θ_2, θ_3) are q_1 and q_2-chainable respectively, then for any q_1-column-orthonormal DB factor $\mathbf{X} \in \mathcal{F}^{\theta_1}$, q_2-row-orthonormal DB factor $\mathbf{Z} \in \mathcal{F}^{\theta_3}$ and $\mathbf{Y} \in \mathcal{F}^{\theta_2}$, we have:*

$$\langle \mathbf{X}\mathbf{Y}_1\mathbf{Z}, \mathbf{X}\mathbf{Y}_2\mathbf{Z} \rangle = \langle \mathbf{Y}_1, \mathbf{Y}_2 \rangle$$

Proof. Since $\mathbf{Y}_i\mathbf{Z}, i = 1, 2$ are $(\theta_2 * \theta_3)$-DB factors (cf. Corollary 6.3.14) and θ_1 and $(\theta_2 * \theta_3)$ are q_1-chainable (cf. Lemma 6.3.15), using Lemma 6.7.10, we have:

$$\langle \mathbf{X}\mathbf{Y}_1\mathbf{Z}, \mathbf{X}\mathbf{Y}_2\mathbf{Z} \rangle = \langle \mathbf{Y}_1\mathbf{Z}, \mathbf{Y}_2\mathbf{Z} \rangle$$

Applying Lemma 6.7.11 this time with θ_2 and θ_3, we have:

$$\langle \mathbf{Y}_1\mathbf{Z}, \mathbf{Y}_2\mathbf{Z} \rangle = \langle \mathbf{Y}_1, \mathbf{Y}_2 \rangle$$

□

Lemma 6.7.13. *Under the same assumptions as Lemma 6.7.12, we have:*

$$\|\mathbf{X}\mathbf{Y}\mathbf{Z}\|_F = \|\mathbf{Y}\|_F \tag{6.52}$$

Proof for Lemma 6.7.13. Lemma 6.7.13 is an immediate corollary of Lemma 6.7.12 by taking $\mathbf{Y}_1 = \mathbf{Y}_2 = \mathbf{Y}$. □

Stable under matrix multiplication

Another similarity between orthonormal DB-factors and classical orthonormal matrices is their stability under matrix multiplication (with additional assumptions for DB-factors). It is formulated in the following result.

Lemma 6.7.14. *If $\mathbf{A}_i, i = 1, 2$ are q_i-column (resp. row)-orthonormal θ_i-DB factors respectively and (θ_1, θ_2) is q_1-chainable (resp. q_2-chainable), then $\mathbf{A}_1\mathbf{A}_2$ is a q_2-column (resp. q_1-row) orthonormal $(\theta_1 * \theta_2)$-DB factor.*

The proof of Lemma 6.7.14 needs the following lemmas.

Lemma 6.7.15. *Let $\mathbf{A}$ be a q-column orthonormal θ-DB factor with $\theta = (a, b, c, d)$. For any subset of column indices of the form $I_\ell = [\![1 + \ell qd, (\ell + 1)qd]\!], \forall \ell = 0, \ldots, ac/q - 1$, the submatrix $\mathbf{A}[:, I_\ell]$ is an orthonormal matrix.*

Proof. In fact, I_ℓ is equal to the quantity $I_{t,s}$ in Equation (6.48) for some $t, s \in \mathbb{N}$. Illustrations for I_ℓ are exactly those of $I_{t,s}$ in Figure 6.5.

Using Equation (6.46), an element of $\mathcal{P}_c(\theta, q)$ has the following form:

$$Z_{k,t} := \{k + tqd, k + d + tqd, \ldots, k + d(q - 1) + tqd\}, \forall k = 1, \ldots, d, \forall t = 0, \ldots, ca/q - 1.$$

It is noteworthy that $I_\ell = \bigcup_{k=1}^{d} Z_{k,\ell}$. The supports of columns of different equivalence classes $Z_{k,t}, k = 1, \ldots, d$ are pairwise disjoint (since sub-blocks are diagonal matrices of size $d \times d$). Readers can visually verify this fact in Figure 6.5. Thus, columns of $\mathbf{X}$ from the different equivalence classes $Z_{k,t}, k = 1, \ldots, d$ are orthogonal. Since $\mathbf{X}$ is assumed to be q-column-orthonormal, the columns of $\mathbf{X}$ from the *same* equivalence classes are also pairwise orthogonal and they are also of unit norm. Thus, $\mathbf{X}[:, I_\ell]$ is a column orthonormal matrix. □

Lemma 6.7.16. *Let θ_2 be a DB-parameter such that $\mathcal{P}_c(\theta_2, q)$ is well-defined for some $q \in \mathbb{N}$. For every θ_1 such that (θ_1, θ_2) is chainable, $\mathcal{P}_c(\theta_1 * \theta_2, q)$ is also well-defined and we have: $\mathcal{P}_c(\theta_2, q) = \mathcal{P}_c(\theta_1 * \theta_2, q)$.*

Proof. Assume that $\theta_1 = (a_1, b_1, c_1, d_1)$ and $\theta_2 = (a_2, b_2, c_2, d_2)$. Because $\mathcal{P}_c(\theta_2, q)$ is well-defined, there exists θ such that (θ_2, θ) is q-chainable. To prove that $\mathcal{P}_c(\theta_1 * \theta_2, q)$ is well-defined, it is sufficient to prove that $(\theta_1 * \theta_2, \theta)$ is q-chainable. This is, however, correct thanks to Lemma 6.4.2.

To prove that $\mathcal{P}_c(\theta_2, q) = \mathcal{P}_c(\theta_1 * \theta_2, q)$, we use the definition of $*$ operator in Definition 6.3.13:

$$\theta_1 * \theta_2 = \left(a_1, \frac{b_1 d_1}{d_2}, \frac{a_2 c_2}{a_1}, d_2\right)$$

By using Equation (6.46) of Lemma 6.7.6, it is easy to verify that $\mathcal{P}_c(\theta_2, q) = \mathcal{P}_c(\theta_1 * \theta_2, q)$, which share all elements of the form:

$$\{k + tq_2d_2, k + d_2 + tq_2d_2, \ldots, k + d_2(q_2 - 1) + tq_2d_2\}, \forall k \in [\![d_2]\!], \forall t = 0, \ldots, a_2c_2/q_2 - 1.$$

□

Proof for Lemma 6.7.14. We will prove the claim for the column-orthonormal case. The ones with row-orthonormal factors can be dealt with similarly.

Using Lemma 6.7.16, we conclude $\mathcal{P}_c(\theta_2, q_1)$ and $\mathcal{P}_c(\theta_1 * \theta_2, q_1)$ are identical and their elements are of the form (cf. Lemma 6.7.6):

$$\{k + tq_2d_2, k + d_2 + tq_2d_2, \ldots, k + d_2(q_2 - 1) + tq_2d_2\}, \forall k \in [\![d_2]\!], \forall t = 0, \ldots, a_2c_2/q_2 - 1.$$

Since $(\mathbf{A}_1\mathbf{A}_2)[:, P] = \mathbf{A}_1(\mathbf{A}_2[:, P])$ for any P subset of row indices, it is sufficient to show that the matrix: $\mathbf{A}_1(\mathbf{A}_2[:, P])$ is orthonormal for all $P \in \mathcal{P}_c(\theta_2, q_2)$. Note that column indices are those of the same block since they share the same supports (see Figure 6.1 for a reminder of *block* and *subblock*).

Apparently, these columns belong to the same (assumed to be) ℓth block of θ_2-DB factor. Therefore, the support of columns of $\mathbf{X}_2$ whose indices are in P is a subset of:

$$I := [\![1 + \ell b_2 d_2, (\ell + 1) b_2 d_2]\!] = [\![1 + \ell d_1 q_1, (\ell + 1) d_1 q_1]\!]$$

because $b_2d_2/d_1 = q_1$. Therefore,

$$\mathbf{A}_1(\mathbf{A}_2[:, P]) = \mathbf{A}_1[:, I]\mathbf{A}_2[I, P]$$

To conclude the proof, we will show that both $\mathbf{A}_1[:, I]$ and $\mathbf{A}_2[I, P]$ are orthonormal matrices by column. Indeed, $\mathbf{A}_2[I, P]$ is a column-orthonormal matrix since $P \in \mathcal{P}_r(\theta_2, q_2)$ and $\mathbf{A}_2$ is assumed to be a q_2-column-orthonormal matrix. Also, $\mathbf{A}_1[:, I]$ is also a column orthonormal matrix thanks to Lemma 6.7.15. The proof is concluded by using the fact that the product of two column-orthonormal matrices remains column-orthonormal. □

This is the end of our presentation of orthonormal DB factors. We now used these results to prove Lemmas 6.5.1, 6.5.3 and 6.5.4.

6.7.9 Proof for Lemmas 6.5.1, 6.5.3 and 6.5.4

In fact, three of them are corollaries of Lemma 6.7.9. Before proving Lemma 6.7.9, let's see how we can apply it to prove all Lemmas 6.5.1, 6.5.3 and 6.5.4. In fact, the correctness of Lemmas 6.5.1 and 6.5.4 is trivial from that of Lemma 6.7.9. It remains to show that Lemma 6.5.3 is also true. The proof of Lemma 6.5.3 uses another corollary of Lemma 6.7.9:

Corollary 6.7.17. *Consider the same setting as Lemma 6.5.3, we have:*

1. $\mathbf{X}_{left}^{(\ell)}$ *is* q_{l-1}*-column-orthnormal* $(\theta_1 * \ldots * \theta_{l-1})$*-DB factor.*
2. $\mathbf{X}_{right}^{(\ell)}$ *is* q_r*-row-orthonormal* $(\theta_{r+1} * \ldots * \theta_N)$*-DB factor.*

Proof. In fact, the goal of Algorithm 9 is to output two new factors $(\tilde{\mathbf{X}}_1, \tilde{\mathbf{X}}_2) \in \mathcal{F}^{\theta_1} \times \mathcal{F}^{\theta_2}$ that share the same product with $\mathbf{X}_1\mathbf{X}_2$ and either $\tilde{\mathbf{X}}_1$ becomes q-column orthonormal or $\tilde{\mathbf{X}}_2$ becomes q-row-orthonormal (depending on the value of t). Therefore, if Θ is $(q_1, \ldots, q_{N-1})$-chainable (cf. Definition 6.3.10), then:

1. The first "for loop" (lines 8-10): By performing line 9, all $\mathbf{X}_{P_k}, k < j$ (j is defined as in line 6) becomes q_{r_k}-column-orthonormal θ_{P_k}-DB factors where $\theta_{P_k} = \theta_{l_k} * \ldots * \theta_{r_k}$, $P_k = [\![l_k, r_k]\!]$.
2. The second "for loop" (lines 11-13): Likewise, all $\mathbf{X}_{P_k}, k > j$ becomes $q_{l_k} - 1$-row-orthonormal θ_{P_k}-DB factors.

Using Lemma 6.7.14 and Lemma 6.4.2, we conclude that $\mathbf{X}_{\text{left}}^{(\ell)}$ and $\mathbf{X}_{\text{right}}^{(\ell)}$ are q_{l-1}-column-orthnormal $(\theta_1 * \ldots * \theta_{l-1})$-DB factor and q_r-row-orthonormal $(\theta_{r+1} * \ldots * \theta_N)$-DB factor respectively (because the union of all intervals $P_k, k < j$ and $P_k, k > j$ are $[\![1, l-1]\!]$ and $[\![r+1, N]\!]$, respectively). □

Proof of Lemma 6.5.3. By Corollary 6.7.17, we have that $\mathbf{X}_{\text{left}}^{(\ell)}$ and $\mathbf{X}_{\text{right}}^{(\ell)}$ are q_{l-1}-column-orthnormal $(\theta_1 * \ldots * \theta_{l-1})$-DB factor and q_r-row-orthonormal $(\theta_{r+1} * \ldots * \theta_N)$-DB factor respectively.

Finally, we recall that $(\theta_1 * \ldots * \theta_{l-1}, \theta_l * \ldots * \theta_r)$ and $(\theta_l * \ldots * \theta_r, \theta_{r+1} * \ldots * \theta_N)$ are q_{l-1}-chainable and q_r-chainable respectively (cf. Lemma 6.4.2). Therefore, Lemma 6.7.13 can be applied and that concludes the proof. □

Proof of Lemma 6.7.9. We focus on the first point since the second point can be dealt with in a similar manner. The proof is based on the following equality: for all $P \in \mathcal{P}(\mathbf{S}_{\theta_1}, \mathbf{S}_{\theta_2})$, for any $(\mathbf{X}_1, \mathbf{X}_2) \in \mathcal{F}^{\theta_1} \times \mathcal{F}^{\theta_2}$, we have:

$$(\mathbf{X}_1\mathbf{X}_2)[R_P, C_P] = \mathbf{X}_1[R_P, P]\mathbf{X}_2[P, C_P], \forall P \in \mathcal{P}(\mathbf{S}_{\theta_1}, \mathbf{S}_{\theta_2}).$$

This fact is proved in Lemma 6.7.2. Since $\text{supp}(\mathbf{X}_1\mathbf{X}_2) \subseteq \bigcup_{P \in \mathcal{P}(\mathbf{S}_{\theta_1}, \mathbf{S}_{\theta_2})} R_P \times C_P$, two matrix products $\mathbf{X}_1\mathbf{X}_2 = \tilde{\mathbf{X}}_1\tilde{\mathbf{X}}_2$ if and only if:

$$(\mathbf{X}_1\mathbf{X}_2)[R_P, C_P] = (\tilde{\mathbf{X}}_1\tilde{\mathbf{X}}_2)[R_P, C_P] = \tilde{\mathbf{X}}_1[R_P, P]\tilde{\mathbf{X}}_2[P, C_P].$$

Moreover, thanks to Remark 6.7.8, $\tilde{\mathbf{X}}_1$ is q-column-orthonormal if and only if $\tilde{\mathbf{X}}_1[R_P, P]$ is orthonormal by column for all $P \in \mathcal{P}(\mathbf{S}_{\theta_1}, \mathbf{S}_{\theta_2})$. This can be accomplished by many different methods, for example:

1. Performing a QR-decomposition[4] of $\mathbf{X}[R_P, P] = \mathbf{QR}$.

2. We set $\tilde{\mathbf{X}}[R_P, P] = \mathbf{Q}, \tilde{\mathbf{Y}}[P, C_P] = \mathbf{RY}_2[P, C_P]$. Thus, $\tilde{\mathbf{X}}[R_P, P]$ is automatically orthonormal by column. Moreover,

$$\begin{aligned}\tilde{\mathbf{X}}[R_P, P]\tilde{\mathbf{Y}}[P, C_P] &= \mathbf{QRY}_2[P, C_P] \\ &= \mathbf{X}_1[R_P, P]\mathbf{Y}_2[P, C_P] = (\mathbf{X}_1\mathbf{X}_2)[R_P, C_P].\end{aligned}$$

Note that the QR-decomposition is possible (with $\mathbf{Q} \in \mathbb{R}^{m\times n}, \mathbf{R} \in \mathbb{R}^{n\times n}$) if $q = |P| = n \leq m = |R_P| = b$. In addition, if $b < q$, it is impossible to make $\mathbf{X}_1[R_P, P]$ orthonormal by column because the number of columns is larger than the dimension. This is why we need (θ_1, θ_2) to be non-redundant. □

6.7.10 Proof for Lemma 6.5.13

Lemma 6.7.18. *Given a chainable, non-redundant $\Theta := (\theta_\ell)_{\ell=1}^N$, three integer numbers $(l, s, r), 1 \leq l \leq s < r \leq N$ and three matrices $(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ such that $\mathrm{supp}(\mathbf{X}) \subseteq \mathbf{S}_{1,l-1}, \mathrm{supp}(\mathbf{Y}) \subseteq \mathbf{S}_{l,r}, \mathrm{supp}(\mathbf{Z}) \subseteq \mathbf{S}_{r+1,N}$ ($\mathbf{S}_{l,r}$ is defined as in Definition 6.5.6) with a convention that $\mathbf{X} \in \mathbf{S}_{1,0}$, if $\mathbf{X}$ is an identity matrix of size $a_1b_1d_1$ and $\mathbf{Z} \in \mathbf{S}_{N+1,N}$ if $\mathbf{Z}$ is an identity matrix of size $a_Nc_Nd_N$. Assume that:*

1. *$\mathbf{X}$ is a q_{l-1}-column-orthonormal $(\theta_1 * \ldots * \theta_{l-1})$-DB factor.*

2. *$\mathbf{Z}$ is a q_r-row-orthonormal $(\theta_{r+1} * \ldots * \theta_N)$-DB factor.*

then for $\mathbf{C} = \mathbf{XYZ}$, we have:

$$E_s^\Theta(\mathbf{C}) = \inf_{\mathbf{Y}_1, \mathbf{Y}_2} \{\|\mathbf{Y} - \mathbf{Y}_1\mathbf{Y}_2\|_F \mid \mathrm{supp}(\mathbf{Y}_1) \subseteq \mathbf{S}_{l,s}, \mathrm{supp}(\mathbf{Y}_2) \subseteq \mathbf{S}_{s+1,r}\}. \tag{6.53}$$

If one admits that Lemma 6.7.18 is true, then proving Lemma 6.5.13 is straight forward. Indeed,

Proof of Lemma 6.5.13. Using Corollary 6.7.17, we have:

1. $\mathbf{X}_{\text{left}}^{(\ell)}$ is q_{l-1}-column-orthnormal $(\theta_1 * \ldots * \theta_{l-1})$-DB factor.

2. $\mathbf{X}_{\text{right}}^{(\ell)}$ is q_r-row-orthonormal $(\theta_{r+1} * \ldots * \theta_N)$-DB factor.

There are, in fact, two possibilities to consider:

1. If $\ell > 1$ (from the second iteration onwards): In this case, $\mathrm{supp}(\mathbf{X}_{[\![l,r]\!]}) \subseteq \mathbf{S}_{l,r}$. Moreover, the $\mathbf{X}_{[\![l,s]\!]}$ and $\mathbf{X}_{[\![s+1,r]\!]}$ are indeed the optimal solutions of the LHS problem of Lemma 6.7.18. It is sufficient to apply Lemma 6.7.18.

2. If $\ell = 1$ (the first iteration): There is only one interval $P_1 = [\![1, N]\!]$. Therefore, $\mathbf{X}_{\text{left}}^{(1)}$ and $\mathbf{X}_{\text{right}}^{(1)}$ are simply identity matrices and $l = 1, r = N$. Thus, the result holds trivially.

[4] In this context, a QR-decomposition of a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}, m \geq n$ is a pair of matrices $(\mathbf{Q}, \mathbf{R})$ in which $\mathbf{Q} \in \mathbb{R}^{m\times n}$ is orthonormal by column and $\mathbf{R}^{n\times n}$ is an upper triangular matrix.

□

Before proving Lemma 6.7.18, we need a technical result.

Corollary 6.7.19. *Consider θ and $\tilde{\theta}$ where $(\theta, \tilde{\theta})$ is q-chainable. For every q-column-orthonormal θ-DB factor $\mathbf{X}$, $\mathbf{X}[:, T_k]$ is a column-orthonormal matrix for every $T_k = \text{supp}(\mathbf{S}_{\theta'}[:, k]), \forall k$, the support of the kth column of a $\tilde{\theta}$-DB factor.*

Proof of Lemma 6.7.18. Assume that $\theta = (a, b, c, d), \tilde{\theta} = (\tilde{a}, \tilde{b}, \tilde{c}, \tilde{d})$ and the column index k belongs to the ℓth block of θ'-DB factor. Then T_k is a subset of the interval I_ℓ:

$$I_\ell := [\![1 + (\ell - 1)\tilde{b}\tilde{d}, \ell\tilde{b}\tilde{d}]\!] = [\![1 + (\ell - 1)dq, \ell dq]\!], \ell \in [\![\tilde{a}]\!]$$

Applying Lemma 6.7.15 yields the proof immediately because a subset of columns of a column-orthonormal matrix constitutes a column-orthonormal matrix. □

Proof for Lemma 6.7.18. The key idea is to use Theorem 6.2.3 because both terms are optimal values of the instances of (6.1.FSMF). We consider two terms one by one as follows:

LHS: Since $\mathbf{S}_{1,s}$ and $\mathbf{S}_{s+1,N}$ are q_s-perfect covering (cf. Lemma 6.4.1), by Theorem 6.2.3, the value of $E_s^{\Theta}(\mathbf{C})$ has the following analytical form:

$$E_s^{\Theta}(\mathbf{C})^2 = \|\mathbf{C}\|_F^2 - \sum_{P \in \mathcal{P}(\mathbf{S}_{1,s}, \mathbf{S}_{s+1,N})} \sum_{j=1}^{q_s} \sigma_j^2(\mathbf{C}[R_P^1, C_P^1]) \tag{6.54}$$

where $R_P^1 = \text{supp}(\mathbf{S}_{1,s}[:, \ell]), C_P^1 = \text{supp}(\mathbf{S}_{s+1,N})[\ell, :]$ for any $\ell \in P$.

RHS: Similarly, we have the right-hand side equal to:

$$\|\mathbf{Y}\|_F^2 - \sum_{P \in \mathcal{P}(\mathbf{S}_{l,s}, \mathbf{S}_{s+1,r})} \sum_{j=1}^{q_s} \sigma_j^2(\mathbf{Y}[R_P^2, C_P^2])$$

where $R_P^2 = \text{supp}(\mathbf{S}_{l,m}[:, \ell]), C_P^2 = \text{supp}(\mathbf{S}_{s+1,r})[\ell, :]$ for any $\ell \in P$.

In fact, R_P^k, C_P^k is essentially equivalent to R_P, C_P defined in Definition 7.6.4 with their respective support constraints. However, we add additional superscripts to distinguish the two cases.

Our first observation is: $\mathcal{P}(\mathbf{S}_{1,s}, \mathbf{S}_{s+1,N}) = \mathcal{P}(\mathbf{S}_{l,m}, \mathbf{S}_{s+1,r}) = \mathcal{P}_c(\theta_s, q_s)$. In fact, $(\theta_1 * \ldots * \theta_s, \theta_{s+1} * \ldots * \theta_N)$ is q_s-chainable (cf. Lemma 6.4.2). Therefore, $\mathcal{P}(\mathbf{S}_{\theta_1 * \ldots * \theta_s}, \mathbf{S}_{\theta_{s+1} * \ldots * \theta_N}) = \mathcal{P}_c(\theta_1 * \ldots * \theta_s, q_s)$. Using Lemma 6.7.16, we have $\mathcal{P}_c(\theta_1 * \ldots * \theta_s, q_s) = \mathcal{P}_c(\theta_s, q_s)$. Therefore, $\mathcal{P}(\mathbf{S}_{1,s}, \mathbf{S}_{s+1,N}) = \mathcal{P}_c(\theta_s, q_s)$

Similarly, we can also prove that $\mathcal{P}(\mathbf{S}_{l,m}, \mathbf{S}_{s+1,r}) = \mathcal{P}_c(\theta_s, q_s)$, thus the claim.

Due to our assumptions of $\mathbf{X}$ and $\mathbf{Z}$ being q_{l-1}-column-orthonormal and q_r-row-orthonormal DB-factors respectively, we have: $\|\mathbf{C}\|_F = \|\mathbf{XYZ}\|_F = \|\mathbf{Y}\|_F$ (cf. Lemma 6.7.13). Therefore, it is sufficient to prove the negative terms of LHS and RHS are equal. Indeed, we will prove that the spectrum (the set of non-zero eigenvalues, including their multiplicities) of $\mathbf{Y}[R_P^2, C_P^2]$ is identical to $\mathbf{C}[R_P^1, C_P^1]$, for all $P \in \mathcal{P}_c(\theta_s, q_s)$. That implies their jth largest eigenvalues are equal.

Since we assume $\text{supp}(\mathbf{Y}) \subseteq \mathbf{S}_{\theta_l * \ldots * \theta_r} = \cup_{P \in \mathcal{P}_c(\theta_s, q_s)} R_P^2 \times C_P^2$. Thus, we can decompose $\mathbf{C} = \mathbf{XYZ}$:

$$\mathbf{C} = \mathbf{XYZ} = \sum_{P \in \mathcal{P}_c(\theta_s, q_s)} \mathbf{X}[:, R_P^2]\mathbf{Y}[R_P^2, C_P^2]\mathbf{Z}[C_P^2, :]$$

Let's investigate the support of each summand of the decomposition of $\mathbf{C}$.

$$\begin{aligned} & \text{supp}(\text{supp}(\mathbf{X}[:, R_P^2])\text{supp}(\mathbf{Y}[R_P^2, C_P^2])\text{supp}(\mathbf{Z}[C_P^2, :])) \\ \subseteq \quad & \text{supp}(\mathbf{S}_{1,l-1}[:, R_P^2]\mathbf{1}_{|R_P^2| \times |C_P^2|}\mathbf{S}_{r+1,N}[C_P^2, :]) \\ \subseteq \quad & \text{supp}(\underbrace{\mathbf{S}_{1,l-1}[:, R_P^2]\mathbf{1}_{|R_P^2| \times 1}}_{\mathbf{S}_{1,l-1}\mathbf{S}_{l,m}[:,\ell]} \underbrace{\mathbf{1}_{1 \times |C_P^2|}\mathbf{S}_{r+1,N}[C_P^2, :]}_{\mathbf{S}_{l+1,r}[\ell,:]\mathbf{S}_{r+1,m}}) \end{aligned}$$

where ℓ can be arbitrary element of P. Note that $\text{supp}(\mathbf{S}_{1,l-1}\mathbf{S}_{l,m}[:, \ell])$ is the support of the ℓth column of $\mathbf{S}_{1,m}$ (cf. Equation (6.13)). Likewise, $\text{supp}(\mathbf{S}_{l+1,r}[\ell, :]\mathbf{S}_{r+1,m})$ is the support of the ℓth row of $\mathbf{S}_{s+1,N}$. Therefore, $\text{supp}(\mathbf{X}[:, R_P^2]\mathbf{Y}[R_P^2, C_P^2]\mathbf{Z}[C_P^2, :]) \subseteq R_P^1 \times C_P^1$. As a consequence, the supports of summands are pairwise disjoint. Thus, we have:

$$\mathbf{X}[:, R_P^2]\mathbf{Y}[R_P^2, C_P^2]\mathbf{Z}[C_P^2, :] = \begin{pmatrix} \mathbf{C}[R_P^1, C_P^1] & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$$

up to some permutation of rows and columns (we want to write $\mathbf{C}[R_P^1, C_P^1]$ as a continuous block for ease of presentation).

To finish the proof, we prove that $\mathbf{X}[:, R_P^2]$ and $\mathbf{Z}[C_P^2, :]$ are column and row orthonormal matrices respectively. It will result immediately that $\mathbf{C}[R_P^1, C_P^1]$ and $\mathbf{Y}[R_P^2, C_P^2]$ share the same spectrum. We show only that $\mathbf{X}[:, R_P^2]$ is a column orthonormal matrix (the similar claim for $\mathbf{Z}[C_P^2, :]$ can be handled with a similar argument). In fact, R_P^2 is the support of a column of $\theta_{l,r}$-DB factors. By associativity (cf. Lemma 6.3.15), $(\theta_{1,l-1}, \theta_{l,r})$ is q_{l-1}-chainable. Since $\mathbf{X}$ is assumed to be q_{l-1}-column-orthonormal DB factor, using Corollary 6.7.19, we conclude that $\mathbf{X}[:, R_P^2]$ is an orthonormal matrix. □

6.7.11 Proof for Theorem 6.5.14

Proof of Theorem 6.5.14. In comparison to the proof of Theorem 6.5.9, we will show a tighter relation between R_ℓ and $R_{\ell-1}$. More specifically, if the ℓ factorization splits at the sth factor, we will prove that:

$$R_\ell^2 \leq 3R_{\ell-1}^2 + 2E_s^\Theta(\mathbf{A})^2, \forall \ell \geq 2$$

The result follows immediately from this relation, as discussed in the proof sketch, right after the statement of Corollary 6.5.15.

To prove the new relation, it is sufficient to show that:

$$\langle \mathbf{P}_{\ell-1} - \mathbf{P}_\ell, \mathbf{P}_p - \mathbf{P}_q \rangle = 0, \forall \ell \geq 1, p, q \geq \ell \tag{6.55}$$

where $\mathbf{P}_\ell$ is defined as in the proof of Theorem 6.5.9, with the convention that $\mathbf{P}_0 = \mathbf{A}$.

Indeed, assume that Equation (6.55) holds, we have:

$$\begin{aligned}
\mathbf{R}_\ell{}^2 &= \|\mathbf{A} - \mathbf{P}_\ell\|_F^2 \\
&= \|(\mathbf{P}_0 - \mathbf{P}_1) + \ldots + (\mathbf{P}_{\ell-1} - \mathbf{P}_\ell)\|_F^2 \\
&= \sum_{i=1}^{\ell} \|\mathbf{P}_{i-1} - \mathbf{P}_i\|_F^2 + 2\sum_{i=1}^{\ell}\sum_{j>i} \langle \mathbf{P}_{i-1} - \mathbf{P}_i, \mathbf{P}_{j-1} - \mathbf{P}_j \rangle \\
&\overset{(6.55)}{=} \|\mathbf{P}_0 - \mathbf{P}_1\|_F^2 + \ldots + \|\mathbf{P}_{\ell-1} - \mathbf{P}_\ell\|_F^2
\end{aligned}$$

Therefore, we deduce that $R_\ell^2 = R_{\ell-1}^2 + \|\mathbf{P}_{\ell-1} - \mathbf{P}_\ell\|_F^2$. By (6.33), we have: $\|\mathbf{P}_{\ell-1} - \mathbf{P}_\ell\|_F \leq R_{\ell-1} + E_s^\theta(\mathbf{A})$. Therefore, $\|\mathbf{P}_{\ell-1} - \mathbf{P}_\ell\|_F^2 \leq 2(R_{\ell-1}^2 + E_s^\theta(\mathbf{A})^2)$ by the Cauchy-Schwarz inequality. Plugging this into the last equation yields to the proof.

It remains to show that Equation (6.55) holds. This is where the observations in the proof sketch come in handy. The key observation in our analysis is: for the identity permutation, the value of $\mathbf{X}_{\ell-1}$ does not change anymore after the orthonormalization at the ℓth iteration. Let $\mathbf{X} = \prod_{i=1}^{\ell-1} \mathbf{X}_i$, we have $\text{supp}(\mathbf{X}) \subseteq \mathbf{S}_{1,\ell-1}$. Since all $\mathbf{X}_k, 1 \leq k < \ell$ is q_k-column-orthonormal, $\mathbf{X}$ is also $q_{\ell-1}$-column-orthonormal (cf. Lemma 6.7.14). In addition, because the value of $\mathbf{X}_{\ell-1}$ does not change, there exist $\mathbf{P}'_t, \forall t \in \{\ell-1, \ell, p, q\}$ such that $\text{supp}(\mathbf{P}'_t) \subseteq \mathbf{S}_{\ell,N}$ and $\mathbf{P}_t = \mathbf{X}\mathbf{P}'_t$ such that with. Using Lemma 6.7.10, we have:

$$\langle \mathbf{P}_{\ell-1} - \mathbf{P}_\ell, \mathbf{P}_p - \mathbf{P}_q \rangle = \langle \mathbf{X}(\mathbf{P}'_{\ell-1} - \mathbf{P}'_\ell), \mathbf{X}(\mathbf{P}'_p - \mathbf{P}'_q) \rangle = \langle \mathbf{P}'_{\ell-1} - \mathbf{P}'_\ell, \mathbf{P}'_p - \mathbf{P}'_q \rangle \tag{6.56}$$

In particular, $\mathbf{P}'_{\ell-1} = \mathbf{X}_{[\![l,N]\!]}$. Moreover, due to our algorithm, $\mathbf{P}'_\ell$ is further factorized and orthonormalized in the next iteration into $\mathbf{X}_\ell \mathbf{P}''_\ell = \mathbf{P}'_\ell$ where $\text{supp}(\mathbf{X}_\ell) \subseteq \mathbf{S}_\ell$ and $\text{supp}(\mathbf{P}''_\ell) \subseteq \mathbf{S}_{\ell+1,N}$. Since the orthonormalization (in the $(\ell+1)$ step) does not change the product of $\mathbf{X}_\ell \mathbf{P}''_\ell = \mathbf{P}'_\ell$, it implies $(\mathbf{X}_\ell, \mathbf{P}''_\ell)$ is an optimal solution of the problem (6.19) with $(l, s, r) = (\ell, \ell, N)$.

Consider $\mathcal{P} := \mathcal{P}(\mathbf{S}_{\ell,\ell}, \mathbf{S}_{\ell+1,N})$, we want to prove that:

$$\langle (\mathbf{P}'_{\ell-1} - \mathbf{P}'_\ell)[R_P, C_P], (\mathbf{P}'_p - \mathbf{P}'_q)[R_P, C_P] \rangle = 0, \forall P \in \mathcal{P}. \tag{6.57}$$

Since $\mathbf{S}_{\ell,N} = \cup_{P \in \mathcal{P}} R_P \times C_P$, we have:

$$\begin{aligned}
\langle \mathbf{P}'_{\ell-1} - \mathbf{P}'_\ell, \mathbf{P}'_p - \mathbf{P}'_q \rangle &= \sum_{P \in \mathcal{P}} \langle (\mathbf{P}'_{\ell-1} - \mathbf{P}'_\ell)[R_P, C_P], (\mathbf{P}'_p - \mathbf{P}'_q)[R_P, C_P] \rangle \\
&\overset{(6.57)}{=} 0
\end{aligned} \tag{6.58}$$

Combine Equation (6.56) and Equation (6.58) yields Equation (6.55).

It remains to prove Equation (6.57). For $(\mathbf{X}_\ell, \mathbf{P}''_\ell)$ to be the optimal solution calculated by Algorithm 5, $\mathbf{P}'_\ell[R_P, C_P]$ has to be the best rank-$|P|$ approximation of $\mathbf{P}'_{\ell-1}[R_P, C_P], \forall P \in \mathcal{P}$. It implies that the span of columns of $\mathbf{P}'_\ell[R_P, C_P]$ and that of $(\mathbf{P}'_{\ell-1} - \mathbf{P}'_\ell)[R_P, C_P]$ are orthogonal for all $P \in \mathcal{P}$.

Moreover, due to Lemma 6.7.2, we have:

$$\mathbf{P}'_\ell[R_P, C_P] = (\mathbf{X}_\ell \mathbf{P}''_\ell)[R_P, C_P] = \mathbf{X}_\ell[R_P, P]\mathbf{P}''_\ell[P, C_P], \forall P \in \mathcal{P}. \tag{6.59}$$

If the rank of $\mathbf{P}'_\ell[R_P, C_P]$ is at most $(|P| - 1)$, so is that of $\mathbf{P}_\ell[R_P, C_P]$. Therefore, $(\mathbf{P}'_{\ell-1} - \mathbf{P}'_\ell)[R_P, C_P] = \mathbf{0}$ and eq. (6.57) holds trivially. Otherwise, the span of columns of $\mathbf{P}'_\ell[R_P, C_P]$ and $\mathbf{X}_\ell[R_P, P]$ are identical. Thus, the span of columns of $\mathbf{X}_\ell[R_P, P]$ and $(\mathbf{P}'_{\ell-1} - \mathbf{P}'_\ell)[R_P, C_P]$ are orthogonal. In the rest of this proof, we will only consider this second case.

Again, since $p, q \geq \ell$, $\mathbf{P}'_t = \mathbf{X}_\ell \mathbf{P}''_t, \text{supp}(\mathbf{P}''_t) \subseteq \mathbf{S}_{\ell+1,N}, t \in \{p, q\}$ because $\mathbf{X}_\ell$ is fixed after orthonormalization step of the next iteration. Similar to our previous argument (cf. Equation (6.59)), the column span of $\mathbf{P}'_p[R_P, C_P], \mathbf{P}'_q[R_P, C_P], P \in \mathcal{P}$ are *subsets* of that of $\mathbf{X}_\ell[R_P, P]$, which is orthogonal to that of $(\mathbf{P}'_{\ell-1} - \mathbf{P}'_\ell)[R_P, C_P]$. As a consequence, Equation (6.57) holds. That concludes our proof. □

6.8 Conclusion

In this chapter, we consider the problem of deformable butterfly factorization. Our main results (cf. Theorem 6.5.9 and Theorem 6.5.14) are the first of its kind, providing an analysis for the error approximation for an algorithm (cf. Algorithm 8) to solve the deformable butterfly factorization problem. As a corollary, we prove that Algorithm 8 is an approximate algorithm, in the sense of (6.12). To the best of our knowledge, Algorithm 8 is the first algorithm to enjoy such a theoretical guarantee, comparing the error of obtained factorization and the best possible approximation, i.e., $E^\Theta(\mathbf{A})$. Another corollary of our main results is an analytic characterization of the set $\mathcal{B}^\Theta$ of a chainble sequence of DB parameters Θ. It implies that most of tge existing butterfly parameterizations (including those in Table 6.1) simply impose low-rank constraints to certain submatrices of the linear operators. The results, however, can be improved or open up many new questions:

1. The constants C_N in both Theorems 6.5.9 and 6.5.14 grow *exponentially* with $N = |\Theta|$. A natural question is whether these C_N (especially that of Theorem 6.5.14) are tight for Algorithm 8.

2. Is there another *version* (i.e., the choice of the permutation σ, additional operations such as orthonormalization) of Algorithm 7 or a completely different algorithm that yields a better constant C_N?

3. All presented algorithms (Algorithms 6 to 8) need to access all the elements of the target matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$. Thus, the complexity of all algorithms is $O(mn)$. This complexity, however, fails to scale for large m, n (for example, $m, n = \Omega(10^6)$). If one assumes that $\mathbf{A} \in \mathcal{B}^\Theta$, is it possible to recover the deformable factors of $\mathbf{A}$, with a faster algorithm, ideally of complexity $\|\Theta\|_0$ (cf. Definition 6.3.18)? Note that this question was already considered in the scientific computing community [LYM+15, LY17] and certain algorithms can achieve the lower-bound $\|\Theta\|_0$. Our interest might shift to provide a theoretical guarantee for these algorithms (with modification, if necessary). As seen in Section 6.6, these tweaks can improve significantly the approximation quality.

4. While theoretically being plausible for linear operator acceleration, in reality, a fast implementation for deformable butterfly factors is non-trivial, especially if the computation involves with GPUs. Therefore, using deformable butterfly factors in sparse

DNNs does not necessarily yield a faster training/inference (when GPUs are used). We are currently working on a fast implementation of deformable butterfly factors for GPUs to harness all of their advantages.

Chapter 7

Existence of optimal solutions in ReLU sparse neural networks

Given a training set, a loss function, and a neural network architecture, it is often taken for granted that optimal network parameters exist, and a common practice is to apply available optimization algorithms to search for them.

In this chapter, we show that the existence of an optimal solution is not always guaranteed, especially in the context of *sparse* ReLU neural networks. In particular, we first show that optimization problems involving deep networks with certain sparsity patterns do not always have optimal parameters, and that optimization algorithms may then diverge. This study leverages the analysis made in Chapter 3 on the ill-posedness of (2.4.FSMF).

Then, the existence of a global optimum is proved for every concrete optimization problem involving a one-hidden-layer sparse ReLU neural network of output dimension one. Overall, the analysis is based on the investigation of two topological properties of the space of functions implementable as sparse ReLU neural networks: a best approximation property, and a closedness property, both in the uniform norm. This is studied both for (finite) domains corresponding to practical training on finite training sets, and for more general domains such as the unit cube. This allows us to provide conditions for the guaranteed existence of an optimum given a sparsity pattern. The results apply not only to several sparsity patterns proposed in recent works on network pruning/sparsification, but also to classical dense neural networks, including architectures not covered by existing results. The materials in this chapter are taken from [LRG23].

7.1 Introduction

The optimization phase in deep learning consists in minimizing an objective function w.r.t. the set of parameters θ of a neural network (NN) such as (2.6) or (2.7). While it is arguably sufficient for optimization algorithms to find local minima in practice, training is also expected to achieve the infimum in many situations (for example, in overparameterized regimes networks are trained to zero learning error).

In this chapter, we take a step back and study a rather fundamental question: *Given a deep learning architecture possibly with sparsity constraints, does its corresponding optimization problem actually admit an optimal θ^*?* The question can be seen as an extension

of what we have seen in Chapter 3 to the setting of sparse DNNs training: many optimization problems such as tensor decomposition, robust principal component analysis, sparse matrix factorization and even classical (dense) DNNs training (Chapter 2) do not have optimal parameters. Such a question is interesting from both practical and theoretical point of views (see Section 3.1 for more detail). Moreover, the answer to this question depends on the architecture of the neural networks (specified by the number of layers, layers width, activation function, and so forth). A response to this question might suggest a guideline for model and architecture selection.

In reality, one usual practical (and also theoretical) trick to bypass the question of the existence of optimal solutions is to add a regularization term, which is usually coercive, e.g., the L^2 norm of the parameters. The existence of optimal solutions then follows by a classical argument on the extrema of a continuous function in a compact domain. Nevertheless, there are many settings where minimizing the regularized version might result in a high value of the loss since the algorithm has to make a trade-off between the loss and the regularizer. Such a scenario is discussed in Example 7.3.5. Therefore, studying the existence of optimal solutions without (explicit) regularization is also a question of interest.

Given a training set $\{(x_i, y_i)\}_{i=1}^P$, the problem of the existence of optimal solutions can be studied from the point of view of the set of functions implementable by the considered network architecture on the finite input domain $\Omega = \{x_i\}_{i=1}^P$. This is the case since the loss is usually of the form $\ell(\mathcal{R}_\theta^\Sigma(x_i), y_i)$ where $\mathcal{R}_\theta^\Sigma$ is the realization of the neural network with parameters θ and the sequence of activation functions Σ (see Equation (2.3)). Therefore, the loss involves directly the image of $\{x_i\}_{i=1}^P$ under the function $\mathcal{R}_\theta^\Sigma$. For theoretical purposes, we also study the function space on the domain $\Omega = [-B, B]^d, B > 0$. In particular, we investigate two topological properties of these function spaces, both w.r.t. the infinity norm $\|\cdot\|_\infty$: the best approximation property (BAP), which is the existence of an optimal solution θ^*, and the closedness, a necessary property for the BAP. These properties are studied in Section 7.3 and Section 7.4, respectively. Most of our analysis is dedicated to the case of *regression problems*. We do make some links to the case of classification problems in Section 7.3.

We particularly focus on analyzing the function space associated with *(structured) sparse ReLU neural networks*, which is motivated by recent advances in machine learning witnessing a compelling empirical success of sparsity based methods in NNs and deep learning techniques, such as pruning, fixed support sparse DNNs, or the lottery ticket hypothesis Section 2.3.4. Our approach exploits the notion of networks either with *fixed sparsity level* (by choosing $\mathcal{E}_i = \mathcal{M}_k^{\text{total}}$ in (2.2.SNNT)) or with *fixed sparsity pattern* (i.e., fixed support sparse DNN). This allows us to establish results covering both classical NNs (whose weights are not constrained to be sparse) and sparse NNs architectures. Our main contributions in this chapter are:

1. **To study the BAP (i.e., the existence of optimal solutions) in practical problems (finite Ω)**: we provide a necessary condition and a sufficient one on the architecture (embodied by a sparsity pattern) to guarantee such existence. As a particular consequence of our results, we show that: a) *for one-hidden-layer NNs with a fixed sparsity level*, the training problem on a finite data set *always admits an optimal solution* (cf. Theorem 7.3.7 and Corollary 7.3.8); b) however, practitioners should be

Works	Architecture	Activation functions	Ω	Function space	BAP
Theorem 7.3.7 Corollary 7.3.8	(Sparse) feed-forward network	ReLU	finite set	$(\mathbb{R}^{P\times d_o}, \|\cdot\|)$, arbitrary $\|\cdot\|$	✓
[Kůr95][KKV00]	Feed-forward network	Heavyside	$[0,1]^d$	$(L^p(\Omega), \|\cdot\|_{L^p})$, $\forall p \in [1,\infty)$	✓
[DK23][DJK23]	(Residual) feed-forward network	ReLU	$\mathbb{R}^n$	$(L^p_\mu(\Omega), \|\cdot\|_{L^p})$, $p = 2^\diamond$, μ is a measure with compact support and is continuous w.r.t Lebesgue measure	✓ (if the target function is continuous)
[PRV21]	Feedforward network	ReLU, pReLU	$[-B,B]^d$	$(C^0(\Omega), \|\cdot\|_\infty)$	×
Corollary 7.4.3†	Feed-forward network	ReLU	$[-B,B]^d$	$(C^0(\Omega), \|\cdot\|_\infty)$	×
Corollary 7.4.4 Corollary 7.4.5	Sparse feed-forward network	ReLU	$[-B,B]^d$	$(C^0(\Omega), \|\cdot\|_\infty)$	×

Table 7.1: **Closedness** results. All results are established for *one-hidden-layer* architectures with *scalar-valued* output, except † (which is valid for one-hidden-layer architectures with vector-valued output). In $\diamond$, if the architecture is simply a feed-forward network, then the result is valid for any $p > 1$. Definition for activation functions can be found in Table A.1. The meaning of the notations $C^0(\Omega), L^p_\mu(\Omega), L^p(\Omega)$ can be found in Notation.

cautious since *there also exist fixed sparsity patterns that do not guarantee the existence of optimal solutions* (cf. Theorem 7.3.3 and Example 7.3.5). In the context of an emerging emphasis on *structured sparsity* (e.g. for GPU-friendliness), this highlights the importance of choosing adequate sparsity patterns.

2. **To study the closedness of the function space on** $\Omega = [-B,B]^d$. As in the finite case, we provide a necessary condition and a sufficient one for the closedness of the function space of ReLU NNs with a fixed sparsity pattern. In particular, our sufficient condition on one-hidden-layer networks generalizes the closedness results of [PRV21, Theorem 3.8] on "dense' one-hidden-layer ReLU NNs to the case of sparse ones, either with fixed sparsity pattern (cf. Theorem 7.4.2, Corollary 7.4.3 and Corollary 7.4.4) or fixed sparsity level (Corollary 7.4.5). Moreover, our necessary condition (Theorem 7.4.1), which is also applicable to deep architectures, exhibits sparsity structures failing the closedness property.

Table 7.1 and Table 7.2 summarize our results and their positioning with respect to existing ones. Somewhat surprisingly, the necessary conditions in both domains (Ω finite and $\Omega = [-B,B]^d$) are identical. Our necessary/sufficient conditions also suggest a relation between sparse ReLU neural networks and their linear counterparts.

Works	Architecture	Activation functions	Function space	Assumptions are valid for any L	Assumptions are valid for any N_{L-1}	Assumptions are valid for any N_L
[GP02]	Feedforward network	Sigmoid	$(C^0(\Omega), \|\cdot\|_\infty)$	× ($L = 2$)	× ($N_{L-1} \geq 2$)	× ($N_L = 1$)
[LMQ21]	Feedforward network	ReLU	$(\mathbb{R}^{2\times 6}, \|\cdot\|)$	× ($L = 2$)	× ($N_{L-1} = 2$)	× ($N_L = 2$)
[PRV21]	Feedforward network	sigmoid, tanh, arctan, ISRLU, ISRU	$(C^0(\Omega), \|\cdot\|_\infty)$	✓	× ($N_{L-1} \geq 2$)	× ($N_L = 1$)
		sigmoid, tanh, arctan, ISRLU, ISRU, ReLU, pReLU	$(L^p(\Omega), \|\cdot\|_{L^p})$			
[MKC21]	Feedforward network	ELU, softsign	$(W^{1,p}(\Omega), \|\cdot\|_{L^p})$ $\forall p \in [1, \infty]$			
		ISRLU	$(W^{2,p}(\Omega), \|\cdot\|_{L^p})$ $\forall p \in [1, \infty]$			
		ISRU, sigmoid, tanh, arctan	$(W^{k,p}(\Omega), \|\cdot\|_{L^p})$ $\forall k, \forall p \in [1, \infty]$			
Theorem 7.4.1‡	(Sparse) feedforward network	ReLU	$(C^0(\Omega), \|\cdot\|_\infty)$	✓	✓	✓
Theorem 7.3.3◇	(Sparse) feedforward network	ReLU	$(\mathbb{R}^{N_L \times P}, \|\cdot\|)$	✓	✓	✓

Table 7.2: **Non-closedness** results (notations in Section 7.2). Previous results consider $\Omega = [-B, B]^d$; ours cover: ◇ a finite Ω; ‡ a bounded Ω with non-empty interior (this includes $\Omega = [-B, B]^d$). Definition for activation functions can be found in Table A.1. The meaning of the notations $C^0(\Omega), L^p_\mu(\Omega), L^p(\Omega), W^{k,p}(\Omega)$ can be found in Notation.

The rest of this chapter is organized as follows: Section 7.2 discusses related works and remind some important notations; the two technical sections, Section 7.3 and Section 7.4, presents the results for the case Ω finite set and $\Omega = [-B, B]^d$ respectively.

7.2 Related works on the existence of optimal parameters of (sparse) DNNs

In this section, we will place emphasis on the existence of optimal parameters (or solutions) of the (sparse) DNNs training problem. The same phenomenon for other problems such as tensor decomposition, RCPA, sparse matrix factorization was already discussed in Chapter 3.

There is an active line of research on the best approximation property and closedness of function spaces of neural networks. Existing results can be classified into two categories: *negative* results, which demonstrate the non-closedness and *positive* results for those showing the closedness or best approximation property of function spaces of NNs. Negative results can notably be found in [GP02, PRV21, MKC21], showing that the set of functions implemented as conventional multilayer perceptrons with various activation functions such as Inverse Square Root Linear Unit (ISRLU), Inverse Square Root Unit (ISRU), parametric ReLU (pReLU), Exponential Linear Unit (ELU) (see Table A.1) is not a closed subset of classical function spaces (e.g., the Lebesgue spaces L^p, the set of continuous functions

C^0 equipped with the sup-norm, or Sobolev spaces $W^{k,p}$). In a more practical setting, [LMQ21] hand-crafts a dataset of six points which makes the training problem of a dense one-hidden-layer neural network not admit any solution. Positive results are proved in [PRV21, Kůs95, KKV00], which establish both the closedness and/or the BAP. The BAP implies closedness [GP02, Proposition 3.1][PRV21, Section 3] (but the converse is not true, see Example B.2.1) hence the BAP can be more difficult to prove than closedness. So far, the only architecture proved to admit the best approximation property (and thus, also closedness) is *one-hidden-layer neural networks* with *heavyside activation* function and *scalar-valued output* (i.e., output dimension equal to *one*) [KKV00] in $L^p(\Omega), \forall p \in [1, \infty]$. If one allows additional assumptions such as the target function f being continuous, then BAP is also established for one-hidden layer and residual one-hidden layer NNs with ReLU activation function [DK23, DJK23].

In all other settings, to the best of our knowledge, the only property proved in the literature is closedness, but the BAP remains elusive.We compare our results with existing works in Tables 7.1 and 7.2.

In machine learning, there is an ongoing endeavour to explore sparse deep neural networks, as a prominent approach to reduce memory and computation overheads inherent in deep learning. One of its most well-known methods is Iterative Magnitude Pruning (IMP), which iteratively trains and prunes connections/neurons to achieve a certain level of sparsity. This method is employed in various works [HPTD15, ZG17], and is related to the so-called Lottery Ticket Hypothesis (LTH) [FC19]. The main issue of IMP is its running time: one typically needs to perform many steps of pruning and retraining to achieve a good trade-off between sparsity and performance. To address this issue, many works attempt to identify the sparsity patterns of the network before training. Once they are found, it is sufficient to train the sparse neural networks once. These *pre-trained* sparsity patterns can be found through algorithms [TKYG20, WZG20, LAT19] or leveraging the sparse structure of well-known fast linear operators such as the Discrete Fourier Transform [DGE+19, DCS+22a, LRC+21, DSG+20, CDL+22]. Regardless of the approaches, these methods are bound to train a neural network with *fixed sparsity pattern* at some points. This is a particular motivation for our work and our study on the best approximation property of sparse ReLU neural networks with fixed sparsity pattern.

Notations Before diving into the technical parts, we remind several notations that were introduced in Chapter 2, which will appear frequently in this chapter.

An architecture with fixed sparsity pattern is specified via $\mathbf{I} = (I_L, \dots, I_1)$, a collection of binary masks $I_i \in \{0,1\}^{N_i \times N_{i-1}}, 1 \leq i \leq L$, where the tuple $(N_L, \dots, N_0)$ denotes the dimensions of the input layer $N_0 = d$, hidden layers $(N_{L-1}, \dots, N_1)$ and output layer (N_L), respectively.

The space of parameters on the sparse architecture $\mathbf{I}$ is denoted $\mathcal{N}_\mathbf{I}$, and for each $\theta \in \mathcal{N}_\mathbf{I}$, $\mathcal{R}_\theta : \mathbb{R}^{N_0} \mapsto \mathbb{R}^{N_L}$ is the function implemented by the ReLU network with parameter θ:

$$\mathcal{R}_\theta : x \in \mathbb{R}^{N_0} \mapsto \mathcal{R}_\theta(x) := \mathbf{W}_L \sigma(\dots \sigma(\mathbf{W}_1 x + \mathbf{b}_1) \dots + \mathbf{b}_{L-1}) + \mathbf{b}_L \in \mathbb{R}^{N_L} \tag{7.1}$$

where $\sigma(x) = \max(0, x)$ is the ReLU activation.

Finally, for a given architecture $\mathbf{I}$, we define

$$\mathcal{L}_\mathbf{I} = \{\mathbf{W}_L \dots \mathbf{W}_1 \mid \mathrm{supp}(\mathbf{W}_i) \subseteq I_i, i \in [\![L]\!]\} \subseteq \mathbb{R}^{N_L \times N_0} \tag{7.2}$$

the set of matrices factorized into L factors respecting the support constraints $I_i, i \in [\![L]\!]$. In fact, $\mathcal{L}_{\mathbf{I}}$ is the set of linear operators implementable as *linear* neural networks (i.e., with $\sigma = \texttt{id}$ instead of the ReLU in (7.1), and no biases) with parameters $\theta \in \mathcal{N}_{\mathbf{I}}$. This definition of $\mathcal{L}_{\mathbf{I}}$ is identical to ones in Question 3.1.11 and Equation (2.11).

To ease the notation, we write $\mathbf{A}[i,:]v$ to denote the scalar product between $\mathbf{A}[i,:]$ and a vector $v \in \mathbb{R}^n$. This notation will be used regularly when we decompose the functions of one-hidden-neural networks into sum of functions corresponding to hidden neurons.

For a vector $v \in \mathbb{R}^d$, $v[I] \in \mathbb{R}^{|I|}$ is the vector v restricted to coefficients in $I \subseteq [\![d]\!]$. If $I = \{i\}$ a singleton, $v[i] \in \mathbb{R}$ is the ith coefficient of v.

The notation $\mathcal{F}_{\mathbf{I}}^{\Sigma}$ (defined in Equation (2.10)) denotes the function space associated to a sparse architecture $\mathbf{I}$ and a sequence of activation function Σ. When $\Sigma = \{\mathrm{Id}, \sigma, \ldots, \sigma\}$ where σ is the ReLU activation function, we use the shorthand $\mathcal{F}_{\mathbf{I}}$ as introduced in Chapter 2.

7.3 Analysis of fixed support ReLU neural networks for finite Ω

The setting of a finite set $\Omega = \{x_i\}_{i=1}^P$ is common in many practical machine learning tasks: models such as (sparse) neural networks are trained on often large (but finite) annotated dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^P$. The optimization/training problem usually takes the form:

$$\underset{\theta}{\text{Minimize}} \qquad \mathcal{L}(\theta) = \sum_{i=1}^{P} \ell(\mathcal{R}_\theta(x_i), y_i), \qquad \text{under sparsity constraints on } \theta \tag{7.3}$$

where ℓ is a loss function measuring the similarity between $\mathcal{R}_\theta(x_i)$ and y_i. A natural question that we would like to address for this task is:

Question 7.3.1. Under which conditions on $\mathbf{I}$, the prescribed sparsity pattern for θ, does the training problem of sparse neural networks admit an optimal solution for any finite data set $\mathcal{D}$?

We investigate this question both for parameters θ constrained to satisfy a *fixed* sparsity pattern $\mathbf{I}$, and in the case of a fixed sparsity level, see e.g. Corollary 7.4.5.

After showing in Section 7.3.1 that the answer to Question 7.3.1 is intimately connected with the closedness of the function space of neural networks with architecture $\mathbf{I}$, we establish in Section 7.3.2 that this closedness implies the closedness of the matrix set $\mathcal{L}_{\mathbf{I}}$ (a property that can be checked using algorithms from real algebraic geometry, see Section 3.2). We also provide concrete examples of support patterns $\mathbf{I}$ where closedness provably fails, and neural network training can diverge. Section 7.3.3 presents sufficient conditions for closedness that enable us to show that an optimal solution always exists on scalar-valued one-hidden-layer networks under a constraint on the sparsity level of each layer.

7.3.1 Equivalence between closedness and best approximation property

To answer Question 7.3.1, it is convenient to view Ω as the matrix $[x_1, \ldots, x_P] \in \mathbb{R}^{d \times P}$ and to consider the function space implemented by neural networks with the given architecture

$\mathbf{I}$ on the input domain Ω in dimension $d = N_0$, with output dimension N_L, defined as the set

$$\mathcal{F}_{\mathbf{I}}(\Omega) := \{\mathcal{R}_\theta(\Omega) \mid \theta \in \mathcal{N}_{\mathbf{I}}\} \subseteq \mathbb{R}^{N_L \times P} \tag{7.4}$$

where the matrix $\mathcal{R}_\theta(\Omega) := \big[\mathcal{R}_\theta(x_1), \dots, \mathcal{R}_\theta(x_P)\big] \in \mathbb{R}^{N_L \times P}$ is the image under $\mathcal{R}_\theta$ of Ω.

We study the closedness of $\mathcal{F}_{\mathbf{I}}(\Omega)$ under the usual topology induced by any norm $\|\cdot\|$ of $\mathbb{R}^{N_L \times P}$. This property is interesting because if $\mathcal{F}_{\mathbf{I}}(\Omega)$ is closed for any $\Omega = \{x_i\}_{i=1}^P$, then an optimal solution is guaranteed to exist for any $\mathcal{D}$ under classical assumptions of $\ell(\cdot, \cdot)$.

Proposition 7.3.2. *Assume that, for any fixed $y \in \mathbb{R}^{N_L}$, $\ell(\cdot, y) : \mathbb{R}^{N_L} \mapsto \mathbb{R}$ is continuous, coercive and that $y = \operatorname{argmin}_{y'} \ell(y', y)$. For any sparsity pattern $\mathbf{I}$ with input dimension $N_0 = d$ the following properties are equivalent:*

1. *irrespective of the training set, problem (7.3) under the constraint $\theta \in \mathcal{N}_{\mathbf{I}}$ has an optimal solution;*
2. *for every P and every $\Omega \in \mathbb{R}^{d \times P}$, the function space $\mathcal{F}_{\mathbf{I}}(\Omega)$ is a closed subspace of $\mathbb{R}^{N_L \times P}$.*

Proof. First, we remind the problem of the training of a sparse neural network on a finite data set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^P$:

$$\underset{\theta \in \mathcal{N}_{\mathbf{I}}}{\text{Minimize}} \qquad \mathcal{L}(\theta) := \sum_{i=1}^{P} \ell(\mathcal{R}_\theta(x_i), y_i), \tag{7.5}$$

which shares the same optimal value as the following optimization problem:

$$\underset{\mathbf{D} \in \mathcal{F}_{\mathbf{I}}(\Omega)}{\text{Minimize}} \qquad \mathcal{L}(\mathbf{D}) := \sum_{i=1}^{P} \ell(\mathbf{D}[:, i], y_i) \tag{7.6}$$

where $\Omega = \{x_i\}_{i=1}^P$. This is simply a change of variables: from $\mathcal{R}_\theta(x_i)$ to the ith column of $\mathbf{D} = \mathcal{R}_\theta(\Omega)$. We prove two implications as follows:

1. *Assume the closedness of $\mathcal{F}_{\mathbf{I}}(\Omega)$ for every finite Ω. Then an optimal solution of the optimization problem (7.5) exists for every finite data set $\{(x_i, y_i)\}_{i=1}^P$.* Consider a training set $\{(x_i, y_i)\}_{i=1}^P$ and $\Omega \coloneqq \{x_i\}_{i=1}^P$. Since $\mathbf{D} \coloneqq \mathbf{0}_{P \times N_L} \in \mathcal{F}_{\mathbf{I}}(\Omega)$ (by setting all parameters in θ equal to zero), the set $\mathcal{F}_{\mathbf{I}}(\Omega)$ is non-empty. The optimal value of (7.6) is thus upper bounded by $\mathcal{L}(\mathbf{0})$. Since the function $\ell(\cdot, y_i)$ is coercive for every y_i in the training set, there exists a constant C (dependent on the training set and the loss) such that minimizing (7.6) on $\mathcal{F}_{\mathbf{I}}(\Omega)$ or on $\mathcal{F}_{\mathbf{I}}(\Omega) \cap \mathcal{B}(0, C)$ (with $\mathcal{B}(\mathbf{0}, C)$ the L^2 ball of radius C centered at zero) yields the same infimum. The function $\mathcal{L}$ is continuous, since each $\ell(\cdot, y_i)$ is continuous by assumption, and the set $\mathcal{F}_{\mathbf{I}}(\Omega) \cap \mathcal{B}(0, C)$ is compact, since it is closed (as an intersection of two closed sets) and bounded (since $\mathcal{B}(0, C)$ is bounded). As a result there exists a matrix $\mathbf{D} \in \mathcal{F}_{\mathbf{I}}(\Omega) \cap \mathcal{B}(0, C)$ yielding the optimal value for (7.6). Thus, the parameters θ such that $\mathcal{R}_\theta(\Omega) = \mathbf{D}$ is an optimal solution of (7.5).

2. *Assume that an optimal solution of problem 7.5 exists for every finite data set* $\{(x_i, y_i)\}_{i=1}^P$. *Then* $\mathcal{F}_{\mathbf{I}}(\Omega)$ *is closed for every* Ω *finite.* We prove the contraposition of this claim. Assume there exists a finite set $\Omega = \{x_i\}_{i=1}^P$ such that $\mathcal{F}_{\mathbf{I}}(\Omega)$ is not closed. Then, there exists a matrix $\mathbf{D} \in \mathbb{R}^{N_L \times P}$ such that $\mathbf{D} \in \overline{\mathcal{F}_{\mathbf{I}}(\Omega)} \setminus \mathcal{F}_{\mathbf{I}}(\Omega)$. Consider the dataset $\{(x_i, y_i)\}_{i=1}^P$ where $y_i \in \mathbb{R}^{N_L}$ is the ith column of $\mathbf{D}$. We prove that the infimum value of (7.5) is $V := \sum_{i=1}^P \ell(y_i, y_i)$. Indeed, since $\mathbf{D} \in \overline{\mathcal{F}_{\mathbf{I}}(\Omega)}$, there exists a sequence $\{\theta_k\}_{k\in\mathbb{N}}$ such that $\lim_{k\to\infty} \mathcal{R}_{\theta_k}(\Omega) = \mathbf{D}$. Therefore, by continuity of $\ell(\cdot, y_i)$, we have:

$$\lim_{k\to\infty} \mathcal{L}(\theta_k) = \sum_{i=1}^P \lim_{k\to\infty} \ell(\mathcal{R}_{\theta_k}(x_i), y_i) = \sum_{i=1}^P \ell(y_i, y_i) = V.$$

Moreover, the infimum cannot be smaller than V because the ith summand is at least $\ell(y_i, y_i)$ (due to the assumption on ℓ in Proposition 7.3.2). Therefore, the infimum value is indeed V. Since we assume that y is the *only* minimizer of $y' \mapsto \ell(y', y)$, this value can be achieved only if there exists a parameter $\theta \in \mathbf{I}$ such that $\mathcal{R}_\theta(\Omega) = \mathbf{D}$. This is impossible due to our choice of $\mathbf{D}$ which *does not* belong to $\mathcal{F}_{\mathbf{I}}(\Omega)$. We conclude that with our constructed data set $\mathcal{D}$, an optimal solution *does not* exist for (7.5). □

Proposition 7.3.2 is somewhat related to Proposition 3.1.6 in Chapter 3 on Equation (2.4.FSMF). They both show that the existence of an optimal solution of the considered optimization problem is equivalent to the closedness of a certain set. The assumption on the loss function ℓ is also natural and realistic in regression problems: any loss function based on any norm on $\mathbb{R}^d$ (e.g. $\ell(y', y) = \|y' - y\|$), such as the quadratic loss, satisfies this assumption. In the classification case, using the soft-max after the last layer together with the cross-entropy loss function indeed leads to an optimization problem with no optimum (regardless of the architecture) when given a *single* training pair . This is due to the fact that changing either the bias or the scales of the last layer can lead the output of the soft-max arbitrarily close to an ideal Dirac mass. It is an interesting challenge to identify whether sufficiently many and diverse training samples (as in concrete learning scenarios) make the problem better posed, and amenable to a relevant closedness analysis.

In light of Proposition 7.3.2 we investigate next the closedness of $\mathcal{F}_{\mathbf{I}}(\Omega)$ for finite Ω.

7.3.2 A necessary closedness condition for fixed support ReLU networks

Our next result reveals connections between the closedness of $\mathcal{F}_{\mathbf{I}}(\Omega)$ for finite Ω and the closedness of $\mathcal{L}_{\mathbf{I}}$, the space of sparse matrix products with sparsity pattern $\mathbf{I}$.

Theorem 7.3.3. *If* $\mathcal{F}_{\mathbf{I}}(\Omega)$ *is closed for every finite* Ω *then* $\mathcal{L}_{\mathbf{I}}$ *is closed.*

Theorem 7.3.3 is a direct consequence of (and in fact logically equivalent to) the following lemma:

Lemma 7.3.4. *If* $\mathcal{L}_{\mathbf{I}}$ *is not closed then there exists a set* $\Omega \subset \mathbb{R}^d$, $d = N_0$, *of cardinality at most* $P := (3N_0 4^{\sum_{i=1}^{L-1} N_i} + 1)^{N_0}$ *such that* $\mathcal{F}_{\mathbf{I}}(\Omega)$ *is not closed.*

The proof is in deferred to Section 7.5.1 due to its involvement. In the following, we present only a sketch of proof to provide a high level description of our ideas.

Sketch of the proof. Since $\mathcal{L}_{\mathbf{I}}$ is not closed, there exists $\mathbf{A} \in \overline{\mathcal{L}_{\mathbf{I}}} \setminus \mathcal{L}_{\mathbf{I}}$ ($\overline{\mathcal{L}}$ is the closure of the set $\mathcal{L}$). Considering $f(x) \coloneqq \mathbf{A}x$, we construct a set $\Omega = \{x_i\}_{i=1}^P$ such that $[f(x_1), \dots, f(x_P)] \in \overline{\mathcal{F}_{\mathbf{I}}(\Omega)} \setminus \mathcal{F}_{\mathbf{I}}(\Omega)$. Therefore, $\mathcal{F}_{\mathbf{I}}(\Omega)$ is not closed. □

Besides showing a topological connection between $\mathcal{F}_{\mathbf{I}}$ (NNs with ReLU activation) and $\mathcal{L}_{\mathbf{I}}$ (linear NNs), Theorem 7.3.3 leads to a simple example where $\mathcal{F}_{\mathbf{I}}$ is not closed.

Example 7.3.5 (**LU** architecture)**.** Consider $\mathbf{I} = (I_2, I_1) \in \{0,1\}^{d\times d} \times \{0,1\}^{d\times d}$ where $I_1 = \{(i,j) \mid 1 \le i \le j \le d\}$ and $I_2 = \{(i,j) \mid 1 \le j \le i \le d\}$. Any pair of matrices $\mathbf{X}_2, \mathbf{X}_1 \in \mathbb{R}^{d\times d}$ such that $\operatorname{supp}(\mathbf{X}_i) \subseteq I_i, i = 1,2$ are respectively lower and upper triangular matrices. Therefore, $\mathcal{L}_{\mathbf{I}}$ is the set of matrices that admit an *exact* lower - upper (LU) factorization/decomposition. That explains its name: **LU** architecture. This set is shown to be not closed in Section 3.1. Therefore, $\mathcal{L}_{\mathbf{I}}$ is not closed and by the contraposition of Theorem 7.3.3 we conclude that there exists a finite set Ω such that $\mathcal{F}_{\mathbf{I}}(\Omega)$ is not closed.

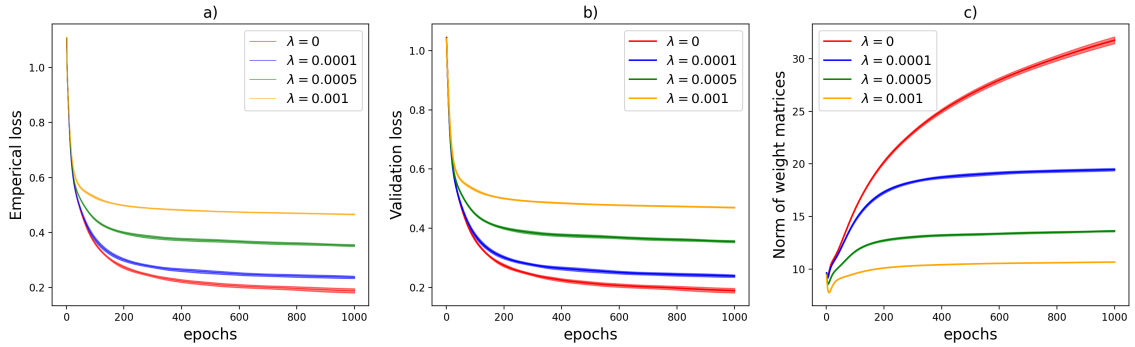


Figure 7.1: Training an one-hidden-layer fixed support (LU architecture) neural network with different regularization hyperparameters λ (we use L^2 regularizer). Subfigures a)-b) show the relative loss (the lower, the better) for training (empirical loss) and testing (validation loss) respectively. Subfigure c) shows the norm of two weight matrices. The experiments are conducted 10 times to produce the error bars in all figures.

Let us illustrate the impact of the non-closedness in Example 7.3.5 via the behavior during the training of a fixed support one-hidden-layer neural network with the LU support constraint **I**. This network is trained to learn the linear function $f(x) \coloneqq \mathbf{A}x$ where $\mathbf{A} \in \mathbb{R}^{d\times d}$ is an anti-diagonal *identity* matrix. Using the necessary and sufficient condition of LU decomposition existence [OJ05, Theorem 1], we have that $\mathbf{A} \in \overline{\mathcal{L}_{\mathbf{I}}} \setminus \mathcal{L}_{\mathbf{I}}$ as in the sketch proof of Lemma 7.3.4. Given network parameters θ and a training set, approximation quality can be measured by the relative loss: $\frac{1}{P}(\sum_{i=1}^P \|\mathcal{R}_\theta(x_i) - y_i\|_2^2 / \|y_i\|_2^2)$.

Figure 7.1 illustrates the behavior of the relative errors of the training set, validation set and the sum of weight matrices norm along epochs, using Stochastic Gradient Descent (SGD) with batch size 3000, learning rate 0.1, momentum 0.9 and four different weight decay (the hyperparameter controlling the L^2 regularizer) $\lambda \in \{0, 10^{-4}, 5\times 10^{-4}, 10^{-3}\}$. The case $\lambda = 0$ corresponds to the *unregularized* case. Our training and testing sets contain each $P = 10^5$ samples (x_i, y_i) were generated independently as $x_i \sim \mathcal{U}([-1,1]^d)$ ($d = 100$) and $y_i \coloneqq \mathbf{A}x_i$.

Example 7.3.5 and Figure 7.1 also lead to two interesting remarks: while the L^2 regularizer ($\lambda > 0$) does prevent the parameter divergence phenomenon, the larger is λ, the larger are the validation and empirical loss. This is the situation where adding a regularization term might be detrimental for the data fitting terms, as stated earlier. More interestingly, the size of the dataset is 10^5, which is much smaller than the theoretical P in Lemma 7.3.4. It is thus interesting to see if we can reduce the theoretical value of P, which is currently exponential w.r.t. to the input dimension.

Another interesting application of Theorem 7.3.3 is to allow us answer the following question:

Question 7.3.6. If the supports of the weight matrices are randomly sampled from a distribution, what is the probability that the corresponding training problem potentially admits no optimal solutions?

While simple, this setting does happen in practice since random supports/binary masks are considered a strong and common baseline for sparse DNNs training [LCC+22].

In fact, thanks to Theorem 7.3.3, if $\mathcal{L}_{\mathbf{I}}$ is not closed then the support is "bad". Thus, to have an estimation of a *lower bound* on the probability of "bad" supports, we could sample the supports from the given distribution and use the algorithm in Section 3.2 to *decide* if $\mathcal{L}_{\mathbf{I}}$ is closed. Unfortunately, this algorithm has doubly exponential complexity, thus hinders its practical use. However, for one-hidden-layer NNs, there is a *polynomial* algorithm to *detect* non-closedness: if the support constraint is "locally similar" to the **LU** structure (precisely, if it satisfies the condition of Theorem 4.3.1), then $\mathcal{L}_{\mathbf{I}}$ is not closed. This fact is elaborated in Section 7.5.3. The resulting detection algorithm can have false positives (i.e., it can fail to detect more complex configurations where $\mathcal{L}_{\mathbf{I}}$ is not closed) but no false negative.

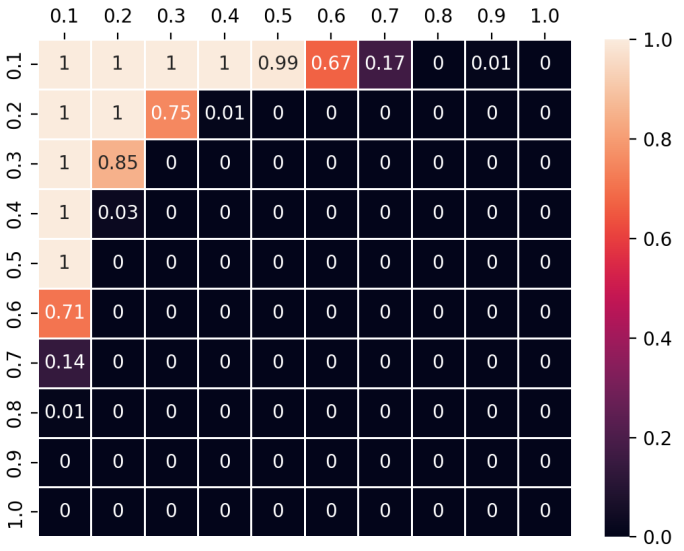


Figure 7.2: Probability of *detectable* "bad" support constraints sampled from uniform distribution over 100 samples.

We test this algorithm on a one-hidden layer ReLU network with two 100×100 weight matrices. We randomly choose their supports whose cardinality are $p_1 100^2$ and $p_2 100^2$ respectively, with $(p_1, p_2) \in \{0.1, 0.2, \ldots, 1.0\}^2$. For each pair (p_1, p_2), we sample 100 instances. Using the detection algorithm, we obtain Figure 7.2. The numbers in Figure 7.2 indicate the probability that a random support constraint (I, J) has $\mathcal{E}_{I,J}$ non-closed (as detected by the algorithm). This figure shows two things: 1) "Bad" architectures such as **LU** are not rare and one can (randomly) generate plenty of them. 2) At a sparse regime $(a_1, a_2 \leq 0.2)$, most of the random supports might lead to training problems without optimal solutions. We remind that the detection algorithm may give some false positives. Thus, for less sparse regimes, it is possible that our heuristic fails to detect the non-closedness. The algorithm indeed gives a lower bound on the probability of finding non-closed instances.

7.3.3 Best approximation property of scalar-valued one-hidden-layer sparse networks

So far, we introduced a necessary condition for the closedness (and thus, by Proposition 7.3.2, the best approximation property) of sparse ReLU networks, and we provided an example of an architecture **I** whose training problem might not admit any optimal solution. One might wonder if there are architectures **I** that *avoid* the issue caused by the non-closedness of $\mathcal{F}_{\mathbf{I}}$. Indeed, we show that for one-hidden-layer sparse ReLU neural networks with scalar output dimension (i.e., $L = 2, N_2 = 1$), the existence of optimal solutions is guaranteed, *regardless of the sparsity pattern.*

Theorem 7.3.7. *Consider scalar-valued, one-hidden-layer ReLU neural networks (i.e., $L = 2, N_2 = 1$). For any support pairs $\mathbf{I} = (I_2, I_1)$ and any finite set $\Omega := \{x_1, \dots, x_P\}$, $\mathcal{F}_{\mathbf{I}}(\Omega)$ is closed.*

The proof is given in Section 7.5.2. As a sanity check, observe that when $L = 2, N_2 = 1$, the necessary condition in Theorem 7.3.3 is satisfied. Indeed, since $N_2 = 1$, $\mathcal{L}_{\mathbf{I}} \subseteq \mathbb{R}^{1\times N_0}$ can be thought as a subset of $\mathbb{R}^{N_0}$. Any $\mathbf{X} \in \mathcal{L}_{\mathbf{I}}$ can be written as a sum: $\mathbf{X} = \sum_{i\in I_2} \mathbf{W}_2[i]\mathbf{W}_1[i,:]$, a decomposition of the product $\mathbf{W}_2\mathbf{W}_1$, where $\mathbf{W}_2[i] \in \mathbb{R}, \mathbf{W}_1[i,:] \in \mathbb{R}^{N_0}, \mathrm{supp}(\mathbf{W}_1[i,:]) \subseteq I_1[i,:]$. Define $\mathcal{H} := \cup_{i\in I_2} I_1[i,:] \subseteq [\![N_0]\!]$ the union of row supports of the first weight matrix. It is easy to verify that $\mathcal{L}_{\mathbf{I}}$ is isomorphic to $\mathbb{R}^{|\mathcal{H}|}$, which is closed. In fact, this argument only works for scalar-valued output, $N_2 = 1$. Thus, there is no conflict between Theorem 7.3.3 and Theorem 7.3.7.

In practice, many approaches search for the best support $\mathbf{I}$ *among a collection of possible supports*, for example, the approach of pruning and training [HPTD15, ZG17] or the lottery ticket hypothesis [FC19]. Our result for fixed support in Theorem 7.3.7 can be also applied in this case and is stated in Corollary 7.3.8. In particular, we consider a set of supports such that the support sizes (or sparsity ratios) of the layers are kept below a certain threshold $K_i, i = 1, \dots, L$. This constraint on the sparsity level of each layer is widely used in many works on sparse neural networks [HPTD15, HMD16, ZG17, FC19].

Corollary 7.3.8. *Consider scalar-valued, one-hidden-layer ReLU neural networks with arbitrary hidden-layer dimension. For any finite data set*[1] $\mathcal{D} = (x_i, y_i)_{i=1}^P$*, problem* (7.3) *under the constraints $\|\mathbf{W}_i\|_0 \le K_i, i = 1, 2$ has a minimizer.*

Proof. Denote $\mathcal{I}$ the collection of sparsity patterns satisfying $\|I_i\|_0 \le K_i, i = 1, 2$, so that a set of parameters satisfies the sparsity constraints $\|\mathbf{W}_i\|_0 \le K_i, i = 1, 2$ if and only if the supports of the weight matrices belong to $\mathcal{I}$. Therefore, to solve the optimization problem under sparsity constraints $\|\mathbf{W}_i\|_0 \le K_i, i = 1, 2$, it is sufficient to solve the same problem for every sparsity pattern in $\mathcal{I}$.

For each $\mathbf{I} \in \mathcal{I}$, we solve a training problem with architecture $\mathbf{I}$ on a given finite dataset $\mathcal{D}$. Thanks to Theorem 7.3.7 and Proposition 7.3.2, the infimum is attained. We take the optimal solution corresponding to $\mathbf{I}$ that yields the smallest value of the loss function $\mathcal{L}$. This is possible because the set $\mathcal{I}$ has a finite number of elements (the total number of possible sparsity patterns is finite). □

7.4 Analysis of fixed support ReLU networks on continuous domains

We now investigate closedness properties when the domain $\Omega \subseteq \mathbb{R}^d$ is no longer finite. Denoting $\mathcal{F}_{\mathbf{I}} = \{\mathcal{R}_\theta : \mathbb{R}^{N_0} \mapsto \mathbb{R}^{N_L} \mid \theta \in \mathcal{N}_{\mathbf{I}}\}$ (with $N_0 = d$) the functions that can be implemented on a given ReLU network architecture $\mathbf{I}$, we are interested in $\mathcal{F}_{\mathbf{I}}(\Omega) = \{f_{|\Omega} : f \in \mathcal{F}_{\mathbf{I}}\}$, the restriction of elements of $\mathcal{F}_{\mathbf{I}}$ to Ω. This is a natural extension of the set $\mathcal{F}_{\mathbf{I}}(\Omega)$ studied in the case of finite Ω.

Specifically, we investigate the closedness of $\mathcal{F}_{\mathbf{I}}(\Omega)$ in $(C^0(\Omega), \|\cdot\|_\infty)$ (the set of continuous functions on Ω equipped with the supremum norm $\|f\|_\infty := \sup_{x\in\Omega} \|f(x)\|_2$).

[1] Notice that $\mathcal{D}$ contains both input vectors x_i and targets y_i, unlike Ω which only contains the inputs.

Contrary to the previous section, we can no longer exploit Proposition 7.3.2 to deduce that the closedness property and the BAP are equivalent. Such a situation is presented in Appendix B.2.1.

The results in this section can be seen as a continuation (and also generalization) of the line of research on the topological property of function space of neural networks [PRV21, GP02, Kůr95, KKV00, MKC21] since we work with fixed support sparse DNNs (which also contains classical DNNs if the binary masks are all-ones), and the output dimension of our considered sparse DNNs can be arbitrary (unlike previous works focusing only on scalar output architectures). In Section 7.4.1 and Section 7.4.2, we provide respectively a necessary and a sufficient condition on $\mathbf{I}$ for the closedness of $\mathcal{F}_{\mathbf{I}}(\Omega)$ in $(C^0(\Omega, \|\cdot\|_\infty)$ respectively. Our necessary condition is applicable for any depth, while the sufficient one is only valid for one-hidden-layer networks ($L = 2$). These results are established under various assumptions on Ω (such as $\Omega = [-B, B]^d$, or Ω being bounded with non-empty interior) that will be specified in each result.

7.4.1 A necessary condition for closedness of fixed support ReLU networks

Theorem 7.4.1 states our result on the necessary condition for the closedness. Interestingly, observe that this result naturally generalizes Theorem 7.3.3. Again, closedness of $\mathcal{L}_{\mathbf{I}}$ in $\mathbb{R}^{N_L \times N_0}$ is with respect to the usual topology defined by any norm.

Theorem 7.4.1. *Consider $\Omega \subset \mathbb{R}^d$ a bounded set with non-empty interior, and $\mathbf{I}$ a sparse architecture with input dimension $N_0 = d$. If $\mathcal{F}_{\mathbf{I}}(\Omega)$ is closed in $(C^0(\Omega), \|\cdot\|_\infty)$ then $\mathcal{L}_{\mathbf{I}}$ is closed in $\mathbb{R}^{N_L \times N_0}$.*

The proof of Theorem 7.4.1 is deferred to Section 7.6.1. Theorem 7.4.1 applies for any Ω which is bounded and has non-empty interior. Thus, it encompasses not only the hypercubes $[-B, B]^d, B > 0$ but also many other domains such as closed or open $\mathbb{R}^d$ balls. Similar to Theorem 7.3.3, Theorem 7.4.1 is interesting in the sense that it allows us to check the non-closedness of the function space $\mathcal{F}_{\mathbf{I}}$ (a subset of the infinite-dimensional space $C^0(\Omega)$) by checking that of $\mathcal{L}_{\mathbf{I}} \subseteq \mathbb{R}^{N_L \times N_0}$ (a finite-dimensional space). The latter can be checked using the algorithm presented in Lemma 3.2.16. Moreover, the **LU** architecture presented in Example 7.3.5 is also an example of $\mathbf{I}$ whose function space is not closed in $(C^0(\Omega), \|\cdot\|_\infty)$.

7.4.2 A sufficient condition for closedness of fixed support ReLU network

The following theorem is the main result of this section. It provides a sufficient condition to verify the closedness of $\mathcal{F}_{\mathbf{I}}(\Omega)$ for $\Omega = [-B, B]^d$ (in the whole chapter we naturally assume $B > 0$) with one-hidden-layer sparse ReLU neural networks.

Theorem 7.4.2. *Consider $\Omega = [-B, B]^d$, $N_0 = d$ and a sparsity pattern $\mathbf{I} = (I_2, I_1)$ such that:*

1. *There is no support constraint for the weight matrix of the second layer, $\mathbf{W}_2$: $I_2 = \mathbf{1}_{N_2 \times N_1}$;*

2. *For each non-empty set of hidden neurons, $S \subseteq [\![N_1]\!]$, $\mathcal{L}_{\mathbf{I}_S}$ is closed in $\mathbb{R}^{N_2 \times N_1}$, where $\mathbf{I}_S \coloneqq (I_2[:, S], I_1[S, :])$ is the support constraint restricted to the sub-network with hidden neurons in S.*

Then the set $\mathcal{F}_{\mathbf{I}}(\Omega)$ is closed in $(C^0(\Omega), \|\cdot\|_\infty)$.

Given the involvement of the proof of Theorem 7.4.2, we dedicate Section 7.6.2 to present its complete proof.

Both conditions in Theorem 7.4.2 can be verified algorithmically: while the first one is trivial to check, the second one requires us to check the closedness of 2^{N_1} sets $\mathcal{L}_{\mathbf{I}_S}$ since it is literally equivalent to the statement: for each fixed binary diagonal matrix $\mathbf{D}$, the set $\{\mathbf{W}_2\mathbf{D}\mathbf{W}_1 \mid \operatorname{supp}(\mathbf{W}_1) \subseteq I_1, \operatorname{supp}(\mathbf{W}_2) \subseteq I_2\}$ is closed. There are at most 2^{N_1} different binary diagonal matrices $\mathbf{D}$, hence the number of support constraints to check. It is still algorithmically possible (although perhaps practically intractable) with the algorithm of Lemma 3.2.16. Apart from its algorithmic aspect, we present two interesting corollaries of Theorem 7.4.2. The first one, Corollary 7.4.3, is about the closedness of the function space of fully connected (i.e., with no sparsity constraint) one-hidden-layer neural networks.

Corollary 7.4.3 (Closedness of fully connected one-hidden-layer ReLU networks *of any output dimension*)**.** *Given $\mathbf{I} = (\mathbf{1}_{N_2 \times N_1}, \mathbf{1}_{N_1 \times N_0})$, the set $\mathcal{F}_{\mathbf{I}}$ is closed in $(C^0([-B, B]^d), \|\cdot\|_\infty)$ where $d = N_0$.*

Proof. The result follows from Theorem 7.4.2 once we check if its assumptions hold. The first one is trivial. To check the second, observe that for every non-empty set of hidden neurons $S \subseteq [\![N_1]\!]$, the set $\mathcal{L}_{\mathbf{I}_S} \subseteq \mathbb{R}^{N_2 \times N_0}$ is simply the set of matrices of rank at most $|S|$, which is closed for any S. □

Corollary 7.4.4 states the closedness of scalar-valued, one-hidden-layer sparse ReLU NNs. In a way, it can be seen as the analog of Theorem 7.3.7 for $\Omega = [-B, B]^d$.

Corollary 7.4.4 (Closedness of *fixed support* one-hidden-layer ReLU networks with scalar output)**.** *Given any input dimension $d = N_0 \geq 1$, any number of hidden neurons $N_1 \geq 1$, scalar output dimension $N_2 = 1$, and any prescribed supports $\mathbf{I} = (I_2, I_1)$, the set $\mathcal{F}_{\mathbf{I}}$ is closed in $(C^0([-B, B]^d), \|\cdot\|_\infty)$.*

Proof. The proof is inductive on the number of hidden neurons N_1:

1. Basic case $N_1 = 1$: Consider $\theta \coloneqq \{(\mathbf{W}_i, \mathbf{b}_i)_{i=1}^2\} \in \mathcal{N}_{\mathbf{I}}$, the function $\mathcal{R}_\theta$ has the form:

$$\mathcal{R}_\theta(x) = \mathbf{w}_2 \sigma(\mathbf{w}_1^\top x + \mathbf{b}_1) + \mathbf{b}_2$$

where $\mathbf{w}_1 = \mathbf{W}_1[1, :] \in \mathbb{R}^{N_0}, \mathbf{w}_2 = \mathbf{W}_2[1, 1] \in \mathbb{R}$. There are two possibilities:

   (a) $I_2 = \emptyset$: then $\mathbf{w}_2 = 0$, $\mathcal{F}_I$ is simply a set of constant functions on Ω, which is closed.

   (b) $I_2 = \{(1, 1)\}$: We have $I_2 = \mathbf{1}_{1 \times N_1}$, which makes the first assumption of Theorem 7.4.2 satisfied. To check that the second assumption of Theorem 7.4.2 also holds, we consider all the possible non-empty subsets S of $[\![1]\!]$: there is only one non-empty subset of I_2, which is $S = [\![1]\!]$. In that case, $\mathcal{L}_{\mathbf{I}_S} = \{\mathbf{W} \in \mathbb{R}^{1 \times N_0} \mid \operatorname{supp}(\mathbf{W}) \subseteq I_1\}$, which is closed (since $\mathcal{L}_{\mathbf{I}_S}$ is isomorphic to $\mathbb{R}^{|I_1|}$). The result thus follows using Theorem 7.4.2.

2. Assume the conclusion of the theorem holds for all $1 \leq N_1 \leq k$ (and any $N_0 \geq 1$). We need to prove the result for $N_1 = k+1$. Define $H = \{i \mid I_2[1,i] = 1\}$ the set of hidden neurons that are allowed to be connected to the output via a nonzero weight. Consider two cases:

(a) If $|H| \leq k$, we have $\mathcal{F}_{\mathbf{I}} = \mathcal{F}_{\mathbf{I}_H}$, which is closed due to the induction hypothesis.

(b) If $H = [\![k+1]\!]$, we can apply Theorem 7.4.2. Indeed, since $I_2 = \mathbf{1}_{1\times N_1}$, the first condition of Theorem 7.4.2 is satisfied. In addition, for any non-empty $S \subseteq [\![N_1]\!]$, define $\mathcal{H} := \cup_{i\in S} I[i,:] \subseteq [\![N_0]\!]$ the union of row supports of $I_1[S,:]$. It is easy to verify that $\mathcal{L}_{\mathbf{I}_S}$ is isomorphic to $\mathbb{R}^{|\mathcal{H}|}$, which is closed. As such, Theorem 7.4.2 can be applied. □

In fact, both Corollary 7.4.3 and Corollary 7.4.4 generalize [PRV21, Theorem 3.8], which proves the closedness of $\mathcal{F}_{\mathbf{I}}([-B,B]^d)$ when $I_2 = \mathbf{1}_{1\times N_1}, I_1 = \mathbf{1}_{N_1\times N_0}$ (classical fully connected one-hidden-layer ReLU networks with ouput dimension equal to one).

To conclude, let us consider the analog to Corollary 7.3.8: we study the function space implementable as a sparse one-hidden-layer network with constraints on the *sparsity level* of each layer (i.e., $\|\mathbf{W}_i\|_0 \leq K_i, i = 1, 2$.

Corollary 7.4.5. *Consider scalar-valued, one-hidden-layer ReLU networks ($L = 2, N_2 = 1, N_1, N_0$) with ℓ^0 constraints $\|\mathbf{W}_1\|_0 \leq K_1, \|\mathbf{W}_2\|_0 \leq K_2$ for some constants $K_1, K_2 \in \mathbb{N}$. The function space $\mathcal{F}([-B,B]^d)$ associated with this architecture is closed in $(C^0([-B,B]^{N_0}), \|\cdot\|_\infty)$.*

Proof. Denote $\mathcal{I} := \{(I_2, I_1) \mid I_2 \subseteq [\![1]\!] \times [\![N_1]\!], I_1 \subseteq [\![N_1]\!] \times [\![N_0]\!], |I_1| \leq K_1, |I_2| \leq K_2\}$ the set of sparsity patterns respecting the ℓ^0 constraints, so that $\mathcal{F}([-B,B]^d) = \bigcup_{\mathbf{I}\in\mathcal{I}} \mathcal{F}_{\mathbf{I}}([-B,B]^d)$. Since $\mathcal{I}$ is finite and $\forall \mathbf{I} \in \mathcal{I}, \mathcal{F}_{\mathbf{I}}([-B,B]^d)$ is closed (Corollary 7.4.4), the result is proved. □

7.5 Complete proofs for Section 7.3

7.5.1 Complete proofs of Lemma 7.3.4 and Theorem 7.3.3

Since Theorem 7.3.3 is a direct corollary of Lemma 7.3.4, this section is devoted to prove Lemma 7.3.4. We first state several necessary technical theorems. Their proof are provided right after the proof of Lemma 7.3.4.

Lemma 7.5.1. *Consider $\Omega = \{x_i\}_{i=1}^P$ a finite subset of $\mathbb{R}^d$ and $\Omega' = [-B,B]^d$ such that $\Omega \subseteq \Omega'$. If $f \in \overline{\mathcal{F}_{\mathbf{I}}(\Omega')}$ (under the topology induced by $\|\cdot\|_\infty$), then $\mathbf{D} := \begin{bmatrix} f(x_1) \dots f(x_P) \end{bmatrix} \in \overline{\mathcal{F}_{\mathbf{I}}(\Omega)}$.*

Lemma 7.5.2. *Consider $\mathcal{R}_\theta$, the realization of a ReLU neural network with parameter $\theta \in \mathbf{I}$. This function is continuous and piecewise linear. On the interior of each piece, its Jacobian matrix is constant and satisfies $\mathbf{J} \in \mathcal{L}_{\mathbf{I}}$.*

Lemma 7.5.3. *For $p, N \in \mathbb{N}$, consider the following set of points (a discretized grid for $[0,1]^N$):*

$$\Omega = \Omega_p^N = \left\{ \left(\frac{i_1}{p}, \dots, \frac{i_N}{p} \right) \mid 0 \leq i_j \leq p, i_j \in \mathbb{N}, \forall 1 \leq j \leq N \right\}.$$

If $H \in \mathbb{N}$ satisfies $p \geq 3NH$, then for any collection of H hyperplanes, there exists $x \in \Omega_p^N$ such that the elementary hypercube whose vertices are of the form

$$\left\{ x + \left(\frac{i_1}{p}, \dots, \frac{i_N}{p} \right) \mid i_j \in \{0, 1\} \ \forall 1 \leq j \leq N \right\} \subseteq \Omega_p^N$$

lies entirely inside a polytope delimited by these hyperplanes.

We are now ready to prove Lemma 7.3.4.

Proof of Lemma 7.3.4. Since $\mathcal{L}_{\mathbf{I}}$ is not closed, there exists a matrix $\mathbf{A} \in \overline{\mathcal{L}_{\mathbf{I}}} \setminus \mathcal{L}_{\mathbf{I}}$, and we consider $f(x) \coloneqq \mathbf{A}x$. Setting $p \coloneqq 3N_0 4^{\sum_{i=1}^{L-1} N_i}$ we construct Ω as the grid:

$$\Omega = \left\{ \left(\frac{i_1}{p}, \dots, \frac{i_{N_0}}{p} \right) \mid 0 \leq i_j \leq p, i_j \in \mathbb{N}, \ \forall 1 \leq j \leq N_0 \right\},$$

so that the cardinality of $\Omega = \{x_i\}_{i=1}^P$ is $P := (p+1)^{N_0}$. Similar to the sketch proof, consider $\mathbf{D} := \big[f(x_1), f(x_2), \dots, f(x_P) \big]$. Our goal is to prove that $\mathbf{D} \in \overline{\mathcal{F}_{\mathbf{I}}(\Omega)} \setminus \mathcal{F}_{\mathbf{I}}(\Omega)$.

First, notice that $\mathbf{D} \in \overline{\mathcal{F}_{\mathbf{I}}(\Omega)}$ as an immediate consequence of Lemma 7.5.1 and Lemma 7.6.1.

It remains to show that $\mathbf{D} \notin \mathcal{F}_{\mathbf{I}}(\Omega)$. We proceed by contradiction, assuming that there exists $\theta \in \mathcal{N}_{\mathbf{I}}$ such that $\mathcal{R}_\theta(\Omega) = \mathbf{D}$.

To show the contradiction, we start by showing that, as a consequence of Lemma 7.5.3 there exists $x \in \Omega$ such that the hypercube whose vertices are the 2^{N_0} points

$$\left\{ x + \left(\frac{i_1}{p}, \dots, \frac{i_{N_0}}{p} \right) \mid i_j \in \{0, 1\}, \forall 1 \leq j \leq N_0 \right\} \subseteq \Omega, \tag{7.7}$$

lies entirely inside a linear region $\mathcal{P}$ of the continuous piecewise linear function $\mathcal{R}_\theta$ [ABMM18]. Denote $K = 2^{\sum_{i=1}^{L} N_i}$ a bound on the number of such linear regions, see e.g. [MPCB14]. Each frontier between a pair of linear regions can be completed into a hyperplane, leading to at most $H = K^2$ hyperplanes. Since $p = 3N_0K^2 \geq 3N_0H$, by Lemma 7.5.3 there exists $x \in \Omega$ such that the claimed hypercube lies entirely inside a polytope delimited by these hyperplanes. As this polytope is itself included in some linear region $\mathcal{P}$ of $\mathcal{R}_\theta$, this establishes our intermediate claim.

Now, define $v_0 \coloneqq x$ and $v_i \coloneqq x + (1/p)\mathbf{e}_i, i \in [\![N_0]\!]$ where $\mathbf{e}_i$ is the ith canonical vector. Denote $\mathbf{P} \in \mathbb{R}^{N_L \times N_0}$ the matrix such that the restriction of $\mathcal{R}_\theta$ to the piece $\mathcal{P}$ is $f_{\mathcal{P}}(x) = \mathbf{P}x + \mathbf{b}$. Since $\mathbf{P}$ is the Jacobian matrix of $\mathcal{R}_\theta$ in the linear region $\mathcal{P}$, we deduce from Lemma 7.5.2 that $\mathbf{P} \in \mathcal{L}_{\mathbf{I}}$. Since the points v_i belong to the hypercube which is both included in $\mathcal{P}$ and in Ω we have for each i:

$$\begin{aligned} \mathbf{P}(v_0 - v_i) &= f_{\mathcal{P}}(v_0) - f_{\mathcal{P}}(v_i) \\ &= \mathcal{R}_\theta(v_0) - \mathcal{R}_\theta(v_i) \\ &= f(v_0) - f(v_i) \\ &= \mathbf{A}(v_0 - v_i). \end{aligned}$$

where the third equality follows from the definition of $\mathbf{D}$ and the fact that we assume $\mathcal{R}_\theta(\Omega) = \mathbf{D}$. Since $v_0 - v_i = e_i/p, i = 1, \dots, n$ are linearly independent, we conclude that $\mathbf{P} = \mathbf{A}$. This implies $\mathbf{A} \in \mathcal{L}_{\mathbf{I}}$, hence the contradiction. This concludes the proof. □

We now prove the intermediate technical lemmas.

Proof of Lemma 7.5.1. Since $f \in \overline{\mathcal{F}_{\mathbf{I}}(\Omega')}$, there exists a sequence $\{\theta_k\}_{k\in\mathbb{N}}$ such that:

$$\lim_{k\to\infty} \sup_{x\in\Omega'} \|f(x) - \mathcal{R}_{\theta_k}(x)\| = 0$$

Denoting $\mathbf{D}_k \coloneqq \left[\mathcal{R}_{\theta_k}(x_1)\dots\mathcal{R}_{\theta_k}(x_r)\right]$, since $x_i \in \Omega \subseteq \Omega'$, $i = 1,\dots,P$, it follows that $\mathbf{D}_k$ converges to $\mathbf{D}$. Since $\mathbf{D}_k \in \mathcal{F}_{\mathbf{I}}(\Omega)$ by construction, we get that $\mathbf{D} \in \overline{\mathcal{F}_{\mathbf{I}}(\Omega)}$. □

Proof of Lemma 7.5.2. For any $\theta \in \mathbf{I}$, $\mathcal{R}_\theta$ is a continuous piecewise linear function since it is the realization of a ReLU neural network [ABMM18]. Consider $\mathcal{P}$ a linear region of $\mathcal{R}_\theta$ with non-empty interior. The Jacobian matrix of $\mathcal{P}$ has the following form [SG22, Lemma 9]:

$$\mathbf{J} = \mathbf{W}_L\mathbf{D}_{L-1}\mathbf{W}_{L-1}\mathbf{D}_{L-2}\dots\mathbf{D}_1\mathbf{W}_1$$

where $\mathbf{D}_i$ is a binary diagonal matrix (diagonal matrix whose coefficients are either one or zero). Since $\operatorname{supp}(\mathbf{D}_i\mathbf{W}_i) \subseteq \operatorname{supp}(\mathbf{W}_i) \subseteq I_i$, we have: $\mathbf{J} = \mathbf{W}_L \prod_{i=1}^{L-1}(\mathbf{D}_i\mathbf{W}_i) \in \mathcal{L}_I$. □

Proof of Lemma 7.5.3. Every edge of an elementary hypercube can be written as:

$$\left(x, x + \frac{1}{p}\mathbf{e}_i\right), x \in \Omega_p^N$$

where $\mathbf{e}_i$ is the ith canonical vector, $1 \le i \le N$. The points x and $x + (1/p)\mathbf{e}_i$ are two *endpoints*. Note that in this proof we use the notation (a, b) to denote the line segment whose endpoints are a and b. By construction, Ω_p^N contains p^N such elementary hypercubes. Given a collection of H hyperplanes, we say that an elementary hypercube is an *intersecting hypercube* if it does not lie entirely inside a polytope generated by the hyperplanes, meaning that there exists a hyperplane that *intersects* at least one of its edges. More specifically, an edge and a hyperplane intersect if they have *exactly* one common point. We exclude the case where there are more than two common points since that implies that the edge lies completely in the hyperplane. The edges that are intersected by at least one hyperplane are called *intersecting edges*. Note that a hypercube can have intersecting edges, but it may not be an intersecting one. A visual illustration of this idea is presented in Figure 7.3.

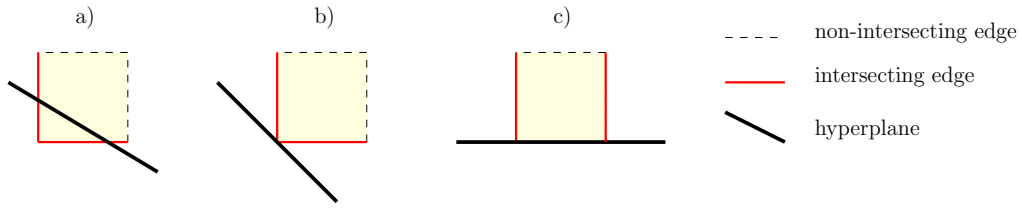


Figure 7.3: Illustration of definitions in $\mathbb{R}^2$: a) an intersecting hypercube with two intersecting edges; b) *not* an intersecting hypercube, but it has two intersecting edges; c) *not* an intersecting hypercube and it only has two intersecting edges (not three according to our definitions: the bottom edge is *not* intersecting).

Formally, a hyperplane $\{w^\top x + b = 0\}$ for $w \in \mathbb{R}^N$ and $b \in \mathbb{R}$ intersects an edge $(x, x + \frac{1}{p}\mathbf{e}_i)$ if:

$$\begin{cases} (w^\top x + b)\left[w^\top (x + \frac{1}{p}\mathbf{e}_i) + b\right] \leq 0 \\ \text{and} \\ w^\top x + b \neq 0 \text{ or } w^\top (x + \frac{1}{p}\mathbf{e}_i) + b \neq 0 \end{cases} \tag{7.8}$$

We further illustrate these notions in Figure 7.4. We emphasize that according to Equation (7.8), ℓ_3 in Figure 7.4 does not intersect any edge *along its direction*.

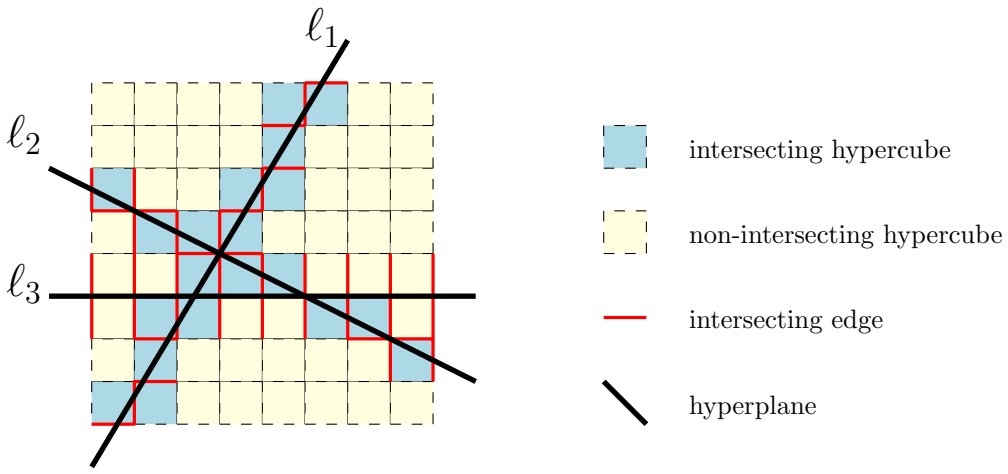


Figure 7.4: Illustration of intersecting hypercubes and hyperplanes in $\mathbb{R}^2$.

Clearly, the number of intersecting hypercubes is upper bounded by the number of intersecting edges. The rest of the proof is devoted to showing that this number is strictly smaller than p^N *if* $p \geq 3NH$, as this will imply the existence of at least one non-intersecting hypercube.

To estimate the *maximum* number of intersecting edges, we analyze the *maximum* number of edges that a given hyperplane can intersect. For a fixed index $1 \leq i \leq N$, we count the number of edges of the form $(x, x + \frac{1}{p}\mathbf{e}_i)$ intersected by a single hyperplane. The key observation is: if we fix all the coordinates of x except the ith one, then the edges $(x, x + \frac{1}{p}\mathbf{e}_i)$ form a line in the ambient space. Among those edges, there are at most *two* intersecting edges with respect to the given hyperplane. This happens only when the hyperplane intersects an edge at one of its endpoints (e.g., the hyperplane ℓ_2 and the second vertical line in Figure 7.4). In total, for each $1 \leq i \leq N$ and each given hyperplane, there are at most $2(p+1)^{N-1}$ intersecting edges of the form $(x, x+\frac{1}{p}\mathbf{e}_i)$. For a given hyperplane, there are thus at most $2N(p+1)^{N-1}$ intersecting edges in total (since $i \in [\![N]\!]$). Since the number of hyperplanes is at most H, there are at most $2NH(p+1)^{N-1}$ intersecting edges, and this quantity also bounds the number of intersecting cubes as we have seen. With the assumption $p \geq 3NH$, we conclude by proving that $p^N > 2NH(p+1)^{N-1}$. Indeed, we

have:

$$\frac{2NH(p+1)^{N-1}}{p^N} = \frac{2NH}{p}\left(\frac{p+1}{p}\right)^{N-1} = \frac{2NH}{p}\left(1+\frac{1}{p}\right)^{N-1} < \frac{2NH}{p}\left(1+\frac{1}{p}\right)^{NH}$$
$$\leq \frac{2NH}{3NH}\left(1+\frac{1}{3NH}\right)^{NH} \leq \frac{2e^{1/3}}{3} \approx 0.93 < 1$$

where we used that $(1+1/n)^n \leq e \approx 2.71828$, the Euler number. □

7.5.2 Proof of Theorem 7.3.7

Proof. We denote $\mathbf{X} = \big[x_1, \ldots, x_P\big] \in \mathbb{R}^{N_0 \times P}$, the matrix representation of Ω. Our proof has three main steps:

Step 1: We show that we can reduce the study of the closedness of $\mathcal{F}_{\mathbf{I}}(\Omega)$ to that of the closedness of a union of subsets of $\mathbb{R}^P$, associated to the vectors $\mathbf{W}_2$. To do this, we prove that for any element $f \in \mathcal{F}_{\mathbf{I}}(\Omega)$, there exists a set of parameters $\theta \in \mathcal{N}_{\mathbf{I}}$ such that the matrix of the second layer $\mathbf{W}_2$ belongs to $\{-1,0,1\}^{1\times N_1}$ (since we assume $N_2 = 1$). This idea is reused from the proof of [ABMM18, Theorem 4.1].

For $\theta := \{(\mathbf{W}_i, \mathbf{b}_i)_{i=1}^2\} \in \mathcal{N}_{\mathbf{I}}$, the function $\mathcal{R}(\theta)$ has the form:

$$\mathcal{R}_\theta(x) = \mathbf{W}_2\sigma(\mathbf{W}_1 x + \mathbf{b}_1) + \mathbf{b}_2 = \sum_{i=1}^{N_1} \mathbf{w}_{2,i}\sigma(\mathbf{w}_{1,i}x + \mathbf{b}_{1,i}) + \mathbf{b}_2$$

where $\mathbf{w}_{1,i} = \mathbf{W}_1[i,:] \in \mathbb{R}^{1\times N_0}, \mathbf{w}_{2,i} = \mathbf{W}_2[i] \in \mathbb{R}, \mathbf{b}_{1,i} = \mathbf{b}[i] \in \mathbb{R}$. Moreover, if $w_{2,i}$ is different from zero, we have:

$$\mathbf{w}_{2,i}\sigma(\mathbf{w}_{1,i}x + \mathbf{b}_1) = \frac{\mathbf{w}_{2,i}}{|\mathbf{w}_{2,i}|}\sigma(|\mathbf{w}_{2,i}|\mathbf{w}_{1,i}x + |\mathbf{w}_{2,i}|\mathbf{b}_{1,i}).$$

In that case, one can assume that $\mathbf{w}_{2,i}$ can be equal to either -1 or 1. Thus, we can assume $\mathbf{w}_{2,i} \in \{\pm 1, 0\}$. For a vector $\mathbf{v} \in \{-1,0,1\}^{1\times N_1}$, we define:

$$F_{\mathbf{v}} = \{[\mathcal{R}_\theta(x_1), \ldots, \mathcal{R}_\theta(x_P)] \mid \theta \in \mathcal{N}_{\mathbf{I},\mathbf{v}}\} \tag{7.9}$$

where $\mathcal{N}_{\mathbf{I},\mathbf{v}} \subseteq \mathcal{N}_{\mathbf{I}}$ is the set of $\theta = \{(\mathbf{W}_i, \mathbf{b}_i)_{i=1}^2\}$ with $\mathbf{W}_2 = \mathbf{v} \in \{0,1\}^{1\times N_1}$, i.e., in words, $F_{\mathbf{v}}$ is the image of Ω through the function $\mathcal{R}_\theta, \theta \in \mathcal{N}_{\mathbf{I},\mathbf{v}}$.

Define $\mathbb{V} := \{\mathbf{v} \mid \text{supp}(\mathbf{v}) \subseteq I_2\} \cap \{0,\pm 1\}^{1\times N_1}$. Clearly, for $\mathbf{v} \in \mathbb{V}$, $F_{\mathbf{v}} \subseteq \mathcal{F}_{\mathbf{I}}(\Omega)$. Therefore,

$$\bigcup_{\mathbf{v}\in\mathbb{V}} F_{\mathbf{v}} \subseteq \mathcal{F}_{\mathbf{I}}(\Omega).$$

Moreover, by our previous argument, we also have:

$$\mathcal{F}_{\mathbf{I}}(\Omega) \subseteq \bigcup_{\mathbf{v}\in\mathbb{V}} F_{\mathbf{v}}.$$

Therefore,

$$\mathcal{F}_{\mathbf{I}}(\Omega) = \bigcup_{\mathbf{v}\in\mathbb{V}} F_{\mathbf{v}}.$$

Step 2: Using the first step, to prove that $\mathcal{F}_{\mathbf{I}}(\Omega)$ is closed, it is sufficient to prove that $F_{\mathbf{v}}$ is closed, $\forall \mathbf{v} \in \mathbb{V}$. This can be accomplished by further decomposing $F_{\mathbf{v}}$ into smaller closed sets. We denote θ' the set of parameters $\mathbf{W}_1, \mathbf{b}_1$ and $\mathbf{b}_2$. In the following, only the parameters of θ' are varied since $\mathbf{W}_2$ is now fixed to $\mathbf{v}$.

Due to the activation function σ, for a given data point $x_j \in \Omega$, we have:

$$\sigma(\mathbf{W}x_j + \mathbf{b}_1) = \mathbf{D}_j(\mathbf{W}x_j + \mathbf{b}_1) \tag{7.10}$$

where $\mathbf{D}_j \in \mathcal{D}$, the set of binary diagonal matrices, and its diagonal coefficients $\mathbf{D}_j[i,i]$ are determined by:

$$\mathbf{D}_j[i,i] = \begin{cases} 0 & \text{if } \mathbf{W}[i,:]x_j + \mathbf{b}_1[i] \leq 0 \\ 1 & \text{if } \mathbf{W}[i,:]x_j + \mathbf{b}_1[i] \geq 0 \end{cases}. \tag{7.11}$$

Note that $\mathbf{D}_j[i,i]$ can take both values 0 or 1 if $\mathbf{W}[i,:]x_j + \mathbf{b}_1[i] = 0$. We call the matrix $\mathbf{D}_j$ the activation matrix of x_j. Therefore, for (7.10) to hold, the N_1 constraints of the form (7.11) must hold simultaneously. It is important to notice that all these constraints are linear w.r.t. θ'. We denote $\mathbf{z}$ a vectorized version of θ' (i.e., we concatenate all coefficients whose indices are in I_1 of $\mathbf{W}$ and $\mathbf{b}_1, \mathbf{b}_2$ into a long vector), and we write all the constraints in (7.10) in a compact form:

$$\mathcal{A}(\mathbf{D}_j, x_j)\mathbf{z} \leq \mathbf{0}_{N_1}$$

where $\mathcal{A}(\mathbf{D}_j, x_j)$ is a constant matrix that depend on $\mathbf{D}_j$ and x_j.

Set $\theta = (\mathbf{v}, \mathbf{z})$. Given that (7.10) holds, we deduce that:

$$\mathcal{R}_\theta(x_j) = \mathbf{v}\sigma(\mathbf{W}x_j + \mathbf{b}_1) + \mathbf{b}_2 = \mathbf{v}\mathbf{D}_j(\mathbf{W}x_j + \mathbf{b}_1) + \mathbf{b}_2 = \mathcal{V}(\mathbf{D}_j, x_j, \mathbf{v})\mathbf{z}$$

where $\mathcal{V}(\mathbf{D}_j, x_j, \mathbf{v})$ is a constant matrix that depends on $\mathbf{D}_j, \mathbf{v}, x_j$. In particular, $\mathcal{R}_\theta(x_j)$ is also a linear function w.r.t the parameters $\mathbf{z}$. Assume that the activation matrices of $(x_1, \ldots, x_P)$ are $(\mathbf{D}_1, \ldots, \mathbf{D}_P)$, then we have:

$$\mathcal{R}_\theta(\Omega) = \big(\mathcal{V}(\mathbf{D}_1, x_1, \mathbf{v})\mathbf{z}, \ldots, \mathcal{V}(\mathbf{D}_P, x_P, \mathbf{v})\mathbf{z}\big) \in \mathbb{R}^{1\times P}.$$

To emphasize that $\mathcal{R}_\theta(\Omega)$ depends linearly on $\mathbf{z}$, for the rest of the proof, we will write $\mathcal{R}_\theta(\Omega)$ as a vector of size P (instead of a row matrix $1 \times P$) as follows:

$$\mathcal{R}_\theta(\Omega) = \mathcal{V}(\mathbf{D}_1, \ldots, \mathbf{D}_P)\mathbf{z} \quad \text{where} \quad \mathcal{V}(\mathbf{D}_1, \ldots, \mathbf{D}_P) = \begin{pmatrix} \mathcal{V}(\mathbf{D}_1, x_1, \mathbf{v}) \\ \vdots \\ \mathcal{V}(\mathbf{D}_P, x_P, \mathbf{v}) \end{pmatrix}.$$

Moreover, to have $(\mathbf{D}_1, \ldots, \mathbf{D}_P)$ activation matrices, the parameters $\mathbf{z}$ need to satisfy:

$$\mathcal{A}(\mathbf{D}_1, \ldots, \mathbf{D}_P)\mathbf{z} \leq \mathbf{0}_Q$$

where $Q = PN_1$ and

$$\mathcal{A}(\mathbf{D}_1, \ldots, \mathbf{D}_P) = \begin{pmatrix} \mathcal{A}(\mathbf{D}_1, x_1) \\ \vdots \\ \mathcal{A}(\mathbf{D}_P, x_P) \end{pmatrix}.$$

Thus, the set of $\mathcal{R}_\theta(\Omega)$ *given* the activation matrices $(\mathbf{D}_1, \ldots, \mathbf{D}_P)$ has the following compact form:

$$F_{\mathbf{v}}^{(\mathbf{D}_1,\ldots,\mathbf{D}_P)} := \{\mathcal{V}(\mathbf{D}_1, \ldots, \mathbf{D}_P)\mathbf{z} \mid \mathcal{A}(\mathbf{D}_1, \ldots, \mathbf{D}_P)\mathbf{z} \leq \mathbf{0}\}.$$

Clearly, $F_{\mathbf{v}}^{(\mathbf{D}_1,\ldots,\mathbf{D}_P)} \subseteq F_{\mathbf{v}}$ since each element is equal to $\mathcal{R}_\theta(\Omega)$ with $\theta = (\mathbf{v}, \mathbf{z})$ for some $\mathbf{z}$. On the other hand, each element of $F_{\mathbf{v}}$ is an element of $F_{\mathbf{v}}^{(\mathbf{D}_1,\ldots,\mathbf{D}_P)}$ for some $(\mathbf{D}_1, \ldots, \mathbf{D}_P) \in \mathcal{D}^P$ since the set of activation matrices corresponding to any θ is in $\mathcal{D}^P$. Therefore,

$$F_{\mathbf{v}} = \bigcup_{(\mathbf{D}_1,\ldots,\mathbf{D}_P)\in\mathcal{D}^P} F_{\mathbf{v}}^{(\mathbf{D}_1,\ldots,\mathbf{D}_P)}.$$

Step 3: Using the previous step, it is sufficient to prove that $F_{\mathbf{v}}^{(\mathbf{D}_1,\ldots,\mathbf{D}_P)}$ is closed, for any $\mathbf{v}, (\mathbf{D}_1, \ldots, \mathbf{D}_P) \in \mathcal{D}^P$. To do so, we write $F_{\mathbf{v}}^{(\mathbf{D}_1,\ldots,\mathbf{D}_P)}$ in a more general form:

$$\{\mathbf{Az} \mid \mathbf{Cz} \leq \mathbf{y}\}. \tag{7.12}$$

Therefore, it is sufficient to prove that a set as in Equation (7.12) is closed. These sets are linear transformations of an intersection of a finite number of half-spaces. Since the intersection of a finite number of halfspaces is *stable* under linear transformations (cf. Lemma 7.5.4 below), and the intersection of a finite number of half-spaces is a closed set itself, the proof can be concluded. □

Lemma 7.5.4 (Closure of intersection of half-spaces under linear transformations)**.** *For any* $\mathbf{A} \in \mathbb{R}^{m\times n}, \mathbf{C} \in \mathbb{R}^{\ell\times n}, \mathbf{y} \in \mathbb{R}^\ell$*, there exists* $\mathbf{C}' \in \mathbf{R}^{k\times m}, \mathbf{b}' \in \mathbf{R}^k$ *such that:*

$$\{\mathbf{Ax} \mid \mathbf{Cx} \leq \mathbf{y}\} = \{\mathbf{C}'\mathbf{z} \leq \mathbf{b}'\}.$$

Proof. The proof uses Fourier–Motzkin elimination [2]. This method is a quantifier elimination algorithm for linear functions [3]. In fact, the LHS can be written as: $\{\mathbf{t} \mid \mathbf{t} = \mathbf{Ax}, \mathbf{Cx} \leq \mathbf{y}\}$, or more generally,

$$\left\{\mathbf{t} \mid \exists \mathbf{x} \in \mathbb{R}^n \text{ s.t. } \mathbf{B}\begin{pmatrix}\mathbf{x}\\ \mathbf{t}\end{pmatrix} \leq \mathbf{v}\right\} \subseteq \mathbb{R}^m$$

where $\begin{pmatrix}\mathbf{x}\\ \mathbf{t}\end{pmatrix}$ is the concatenation of two vectors $(\mathbf{x}, \mathbf{t})$ and the linear constraints imposed by $\mathbf{B}\begin{pmatrix}\mathbf{x}\\ \mathbf{t}\end{pmatrix} \leq \mathbf{v}$ replace the two linear constraints $\mathbf{Cx} \leq \mathbf{y}$ and $\mathbf{t} = \mathbf{Ax}$. The idea is to show that:

$$\left\{\mathbf{t} \mid \exists \mathbf{x} \in \mathbb{R}^n \text{ s.t. } \mathbf{B}\begin{pmatrix}\mathbf{x}\\ \mathbf{t}\end{pmatrix} \leq \mathbf{v}\right\} = \left\{\mathbf{t} \mid \exists \mathbf{x}' \in \mathbb{R}^{n-1} \text{ s.t. } \mathbf{B}'\begin{pmatrix}\mathbf{x}'\\ \mathbf{t}\end{pmatrix} \leq \mathbf{v}'\right\} \tag{7.13}$$

for some matrix $\mathbf{B}'$ and vector $\mathbf{v}'$. By doing so, we reduce the dimension of the quantified parameter $\mathbf{x}$ by one. By repeating this procedure until there is no more quantifier, we prove the lemma. The rest of this proof is thus devoted to show that $\mathbf{B}', \mathbf{v}'$ as in (7.13) do exist.

We will show how to eliminate the first coordinate of $\mathbf{x}[1]$. First, we partition the set of linear constraints of LHS of (7.13) into three groups:

[2] More detail about this method can be found in this link

[3] In fact, the algorithm determining the closedness of $\mathcal{L}_\mathbf{I}$ is also a quantifier elimination one, but it can be used in a more general setting: polynomials

1. $S_0 := \{j \mid \mathbf{B}[j,1] = 0\}$: In this case, $\mathbf{x}[1]$ does not appear in this constraint, there is nothing to do.

2. $S_+ := \{j \mid \mathbf{B}[j,1] > 0\}$, for $j \in S_+$, we can rewrite the constraints $\mathbf{B}[j,:]\begin{pmatrix}\mathbf{x}\\ \mathbf{t}\end{pmatrix} \leq \mathbf{v}[j]$ as:
$$\mathbf{x}[1] \leq \gamma[j] + \sum_{i=2}^{n} \alpha[i]\mathbf{x}[i] + \sum_{i=1}^{m} \beta[i]\mathbf{t}[i] := B_j^+(\mathbf{x}', \mathbf{t})$$
for some suitable $\gamma[j], \alpha[i], \beta[i]$ where $\mathbf{x}'$ is the last $(n-1)$ coordinate of the vector $\mathbf{x}$.

3. $S_- := \{j \mid \mathbf{B}[j,1] < 0\}$: for $j \in S_-$, we can rewrite the constraints $\mathbf{B}[j,:]\begin{pmatrix}\mathbf{x}\\ \mathbf{t}\end{pmatrix} \leq \mathbf{v}_j$ as:
$$\mathbf{x}[1] \geq \gamma[j] + \sum_{i=2}^{n} \alpha[i]\mathbf{x}[i] + \sum_{i=1}^{m} \beta[i]\mathbf{t}[i] := B_j^-(\mathbf{x}', \mathbf{t}).$$

For the existence of such $\mathbf{x}[1]$, it is necessary and sufficient that:

$$B_k^+(\mathbf{x}', \mathbf{t}) \geq B_j^-(\mathbf{x}', \mathbf{t}), \qquad \forall k \in S_+, j \in S_-. \tag{7.14}$$

Thus, we form the matrix $\mathbf{B}'$ and the vector $\mathbf{v}'$ such that the linear constraints written in the following form:

$$\mathbf{B}' \begin{pmatrix}\mathbf{x}'\\ \mathbf{t}\end{pmatrix} \leq \mathbf{v}'$$

represent all the linear constraints in the set S_0 and those in the form of (7.14). Using this procedure recursively, one can eliminate all quantifiers and prove the lemma. □

7.5.3 Polynomial algorithm to detect a pair of support constraints (I, J) with non-closed $\mathcal{L}_{(I,J)}$.

Lemma 7.5.5. *Consider $\mathbf{I} = (I, J) \in \{0,1\}^{m\times r} \times \{0,1\}^{r\times n}$ support constraints for the weight matrices of one-hidden-layer neural network. If there exists four indices $1 \leq i_1, i_2 \leq m, 1 \leq j_1, j_2 \leq n$ and two indices $k \neq l, 1 \leq k, l \leq r$such that:*

1. *For each pair $(i,j) \in \{(i_1,j_1),(i_1,j_2),(i_2,j_1)\}$ $(i,j) \in \text{supp}(I[:,k]J[k,:])$ and $(i,j) \notin \text{supp}(I[:,\ell]J[\ell,:]), \forall \ell \neq k$.*

2. *The pair (i_2,j_2) belongs to $\text{supp}(I[:,k]J[k,:])$ and $\text{supp}(I[:,l]J[l,:])$.*

Proof. First, it is easy to see that the assumptions of this lemma are equivalent to those of Theorem 4.3.1 since $\text{supp}(I[:,k]J[k,:])$ is precisely the kth rank-one support of the pair (I,J). Without loss of generality, one can assume that $i_1, j_1 = 1, i_2, j_2 = 2$ and $k = 1, l = 2$. We will prove that the matrix:

$$\mathbf{A} = \begin{pmatrix}\mathbf{A}' & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{pmatrix} \in \mathbb{R}^{m\times n}, \text{ where } \mathbf{A}' = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix} \in \mathbb{R}^{2\times 2}$$

satisfy: $\mathbf{A} \in \overline{\mathcal{L}_\mathbf{I}} \setminus \mathcal{L}_\mathbf{I}$. This can be shown in two steps:

1. Proof that $\mathbf{A} \in \overline{\mathcal{L}_\mathbf{I}}$: For any $\epsilon > 0$, consider two factors:

$$\mathbf{X}_\epsilon = \begin{pmatrix} \mathbf{X}'_\epsilon & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}, \mathbf{Y}_\epsilon = \begin{pmatrix} \mathbf{Y}'_\epsilon & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$$

where $\mathbf{X}'_\epsilon, \mathbf{Y}'_\epsilon \in \mathbb{R}^{2\times 2}$ and respect the support constraints corresponding to **LU** architecture. It is not hard to see that such a construction of $(\mathbf{X}_\epsilon, \mathbf{Y}_\epsilon)$ satisfies the support constraints (I, J) (due to the assumption of the lemma and the value of indices). Moreover, we also have:

$$\|\mathbf{A} - \mathbf{X}_\epsilon \mathbf{Y}_\epsilon\|_F = \|\mathbf{A}' - \mathbf{X}'_\epsilon \mathbf{Y}'_\epsilon\|_F$$

Thus, to have $\|\mathbf{A} - \mathbf{X}_\epsilon \mathbf{Y}_\epsilon\|_F \leq \epsilon$, it is sufficient to choose a pair of factors $(\mathbf{X}'_\epsilon, \mathbf{Y}'_\epsilon)$ respecting the **LU** architecture of size 2×2 such that $\|\mathbf{A}' - \mathbf{X}'_\epsilon \mathbf{Y}'_\epsilon\|_F \leq \epsilon$. Such a pair exists, since the set of matrices admitting the exact LU decomposition is dense in $\mathbb{R}^{2\times 2}$. This holds for any $\epsilon > 0$. Therefore, $\mathbf{A} \in \overline{\mathcal{L}_\mathbf{I}}$.

2. Proof that $\mathbf{A} \notin \mathcal{L}_\mathbf{I}$: Assume there exist a pair of factors $(\mathbf{X}, \mathbf{Y})$ whose product $\mathbf{XY} = \mathbf{A}$ and supports are included in (I, J). Due to the assumptions on the pairs $(i_1, j_1), (i_1, j_2), (i_2, j_1)$, we must have:

$$\begin{cases} \mathbf{X}[1,1]\mathbf{Y}[1,1] & = \mathbf{A}[1,1] = 0 \\ \mathbf{X}[2,1]\mathbf{Y}[1,1] & = \mathbf{A}[2,1] = 1 \\ \mathbf{X}[1,1]\mathbf{Y}[1,2] & = \mathbf{A}[1,2] = 1 \end{cases}$$

It is easy to see that it is impossible. Therefore, $\mathbf{A} \notin \mathcal{L}_\mathbf{I}$. □

The conditions of Lemma 7.5.5 can be verified in polynomial time. A brutal force algorithm will have complexity $O(m^2n^2r)$. A more clever implementation with careful book-marking can reduce this complexity to $O(\min(m, n)mnr)$.

7.6 Complete proofs for Section 7.4

7.6.1 Proof for Theorem 7.4.1

In fact, Theorem 7.4.1 is a corollary of Lemma 7.6.1, which states as follow:

Lemma 7.6.1. *If $\mathbf{A} \in \overline{\mathcal{L}_\mathbf{I}} \backslash \mathcal{L}_\mathbf{I} \subseteq \mathbb{R}^{N_L \times N_0}$ then the function $f : x \mapsto f(x) := \mathbf{A}x$ satisfies $f \in \overline{\mathcal{F}_\mathbf{I}(\Omega)} \setminus \mathcal{F}_\mathbf{I}(\Omega)$ for every subset Ω of $\mathbb{R}^{N_0}$ that is bounded with non-empty interior.*

Thus, it is sufficient to give a proof for Lemma 7.6.1. It is presented in the following.

Proof of Lemma 7.6.1. Since $\mathbf{A} \in \overline{\mathcal{L}_\mathbf{I}} \backslash \mathcal{L}_\mathbf{I} \subseteq \mathbb{R}^{N_L \times N_0}$, we have:

1. $\mathbf{A} \notin \mathcal{L}_\mathbf{I}$.
2. There exists a sequence $\{(\mathbf{X}_i^k)_{i=1}^L\}_{k\in\mathbb{N}}$ such that $\lim_{k\to\infty} \|\mathbf{X}_L^k \dots \mathbf{X}_1^k - \mathbf{A}\| = 0$ for any norm defined on $\mathbb{R}^{N_0}$.

We will prove that the linear function: $f(x) := \mathbf{A}x$ satisfies $f \in \overline{\mathcal{F}_\mathbf{I}} \setminus \mathcal{F}_\mathbf{I}$ (where $\overline{\mathcal{F}_\mathbf{I}}$ is the closure of $\mathcal{F}_\mathbf{I}$ in $(C^0(\Omega), \|\cdot\|_\infty)$, that is to say f is not the realization of any neural network but it is the uniform limit of the realizations of a sequence of neural networks). Firstly, we prove that $f \notin \mathcal{F}_\mathbf{I}$. For the sake of contradiction, assume there exists $\theta = (\mathbf{W}_i, \mathbf{b}_i)_{i=1}^L \in \mathcal{N}_\mathbf{I}$ such that $\mathcal{R}_\theta = f$. Since $\mathcal{R}_\theta$ is the realization of a ReLU neural network, it is a continuous piecewise linear function. Therefore, since Ω has non-empty interior, there exist a non-empty open subset Ω' of $\mathbb{R}^d$ such that $\Omega' \subseteq \Omega$ and $\mathcal{R}_\theta$ is linear on Ω', i.e., there are $\mathbf{A}' \in \mathbb{R}^{N_L \times N_0}$, $\mathbf{b} \in \mathbb{R}^{N_L}$ such that $\mathcal{R}_\theta(x) = \mathbf{A}'x + \mathbf{b}', \forall x \in \Omega'$. Since $f = \mathcal{R}_\theta$, we have: $\mathbf{A}' = \mathbf{A}$ and also equal to the Jacobian matrix of $\mathcal{R}_\theta$ on Ω'. Using Lemma 7.5.2 and the fact that $\mathbf{A} \notin \mathcal{L}_\mathbf{I}$, we conclude that $f \notin \mathcal{F}_\mathbf{I}$.

There remains to construct a sequence $\{\theta^k\}_{k\in\mathbb{N}}, \theta^k = (\mathbf{W}_i^k, \mathbf{b}_i^k)_{i=1}^L \in \mathcal{N}_\mathbf{I}$ such that $\lim_{k\to\infty} \|\mathcal{R}_{\theta^k} - f\|_\infty = 0$. We will rely on the sequence $\{(\mathbf{X}_i^k)_{i=1}^L\}_{k\in\mathbb{N}}$ for our construction. Given $k \in \mathbb{N}$ we simply define the weight matrices as $\mathbf{W}_i^k = \mathbf{X}_i^k, 1 \leq i \leq L$. The biases are built recursively. Starting from $c_1^k := \sup_{x\in\Omega} \|\mathbf{W}_1^k x\|_\infty$ and $\mathbf{b}_1^k := c_1^k \mathbf{1}_{N_1}$, we iteratively define for $2 \leq i \leq L-1$:

$$\begin{aligned}
\gamma_{i-1}^k(x) &:= \mathbf{W}_{i-1}^k x + \mathbf{b}_{i-1} \\
c_i^k &:= \sup_{x\in\Omega} \|\gamma_{i-1}^k \circ \ldots \circ \gamma_1^k(x)\|_\infty \\
\mathbf{b}_i^k &:= c_i^k \mathbf{1}_{N_i}.
\end{aligned}$$

The boundedness of Ω ensures that c_i^k is well-defined with a finite supremum. For $i = L$ we define:

$$\mathbf{b}_L^k = -\sum_{i=1}^{L-1} \Big(\prod_{j=i+1}^{L} \mathbf{W}_j \Big) \mathbf{b}_i^k.$$

We will prove that $\mathcal{R}_{\theta^k}(x) = \left(\mathbf{X}_L^k \ldots \mathbf{X}_1^k\right) x, \forall x \in \Omega$. As a consequence, it is immediate that:

$$\begin{aligned}
\lim_{k\to\infty} \|\mathcal{R}_{\theta^k} - f\|_\infty &= \lim_{k\to\infty} \sup_{x\in\Omega} \|\mathcal{R}_{\theta^k}(x) - f(x)\|_2 \\
&\leq \lim_{k\to\infty} \|\mathbf{X}_L^k \ldots \mathbf{X}_1^k - \mathbf{A}\|_{2\to 2} \sup_{x\in\Omega} \|x\|_2 = 0
\end{aligned}$$

where we used that all matrix norms are equivalent and denoted $\|\cdot\|_{2\to 2}$ the operator norm associated to Euclidean vector norms. Back to the proof that $\mathcal{R}_{\theta^k}(x) = \left(\mathbf{X}_L^k \ldots \mathbf{X}_1^k\right) x, \forall x \in \Omega$, due to our choice of c_i^k, we have for $2 \leq i \leq L-1$:

$$\gamma_{i-1}^k \circ \ldots \circ \gamma_1^k(x) \geq 0, \forall x \in \Omega$$

where $\geq$ is taken in coordinate-wise manner. Therefore, an easy induction yields:

$$\begin{aligned}
\mathcal{R}_{\theta^k}(x) &= \gamma_L^k \circ \sigma \circ \gamma_{L-1}^k \circ \ldots \circ \sigma \circ \gamma_1^k(x) \\
&= \gamma_L^k \circ \gamma_{L-1}^k \ldots \circ \gamma_1^k(x) \\
&= \mathbf{W}_L^k(\ldots(\mathbf{W}_2^k(\mathbf{W}_1^k x + \mathbf{b}_1^k) + \mathbf{b}_2^k)\ldots) + \mathbf{b}_L^k \\
&= (\mathbf{X}_L^k \ldots \mathbf{X}_1^k)x + \sum_{i=1}^{L-1} \Big(\prod_{j=i+1}^{L} \mathbf{W}_j \Big) \mathbf{b}_i^k - \sum_{i=1}^{L-1} \Big(\prod_{j=i+1}^{L} \mathbf{W}_j \Big) \mathbf{b}_i^k \\
&= (\mathbf{X}_L^k \ldots \mathbf{X}_1^k)x.
\end{aligned}$$

□

7.6.2 Complete proof for Theorem 7.4.2

Given the involvement of Theorem 7.4.2, we decompose its proof and present it in two subsections: the first one establishes general results that do not use the assumption of Theorem 7.4.2. The second one combines the established results with the assumption of Theorem 7.4.2 to provide a full proof.

Properties of the limit function of fixed support one-hidden-layer NNs

The main results of this parts are summarized in Lemma 7.6.3 and Lemma 7.6.6. It is important to emphasize that all results in this section do *not* make any assumption on $\mathbf{I}$.

We first introduce the following technical results.

Lemma 7.6.2 (Normalization of the rows of the first layer [PRV21])**.** *Consider Ω a bounded subset of $\mathbb{R}^{N_0}$. Given any $\theta = \{(\mathbf{W}_i, \mathbf{b}_i)_{i=1}^2\} \in \mathcal{N}_\mathbf{I}$ and any norm $\|\cdot\|$ on $\mathbb{R}^{N_0}$, there exists $\tilde{\theta} := \{(\tilde{\mathbf{W}}_i, \tilde{\mathbf{b}}_i)_{i=1}^2\} \in \mathcal{N}_\mathbf{I}$ such that the matrix $\tilde{\mathbf{W}}_1$ has unit norm rows, $\|\tilde{\mathbf{b}}_1\|_\infty \leq C := \sup_{x\in\Omega} \sup_{\|u\|\leq 1} \langle u, x\rangle$ and $\mathcal{R}_\theta(x) = \mathcal{R}_{\tilde{\theta}}(x), \forall x \in \Omega$.*

Proof. We report this proof for self-completeness of this work. It is *not* a contribution, as it merely combines ideas from the proof of [PRV21, Lemma D.2] and [PRV21, Theorem 3.8, Steps 1-2].

We first show that for each set of weights $\theta \in \mathcal{N}_\mathbf{I}$ we can find another set of weights $\theta' = \{(\mathbf{W}'_i, \mathbf{b}'_i)_{i=1}^2\} \in \mathcal{N}_\mathbf{I}$ such that $\mathcal{R}_\theta = \mathcal{R}_{\theta'}$ on $\mathbb{R}^{N_0}$ and $\mathbf{W}'_1$ has unit norm rows. Note that $\|\mathbf{b}'_1\|_\infty$ can be larger than C. Indeed, given $\theta \in \mathcal{N}_\mathbf{I}$, the function $\mathcal{R}_\theta$ can be written as: $\mathcal{R}_\theta : x \in \mathbb{R}^{N_0} \mapsto \sum_{j=1}^{N_1} h_j(x) + \mathbf{b}_2$ where $h_j(x) = \mathbf{W}_2[:, j]\sigma(\mathbf{W}_1[j, :]x + \mathbf{b}_1[j])$ denotes the contribution of the jth hidden neuron. For hidden neurons corresponding to nonzero rows of $\mathbf{W}_1^k$, we can rescale the rows of $\mathbf{W}_1^k$, the columns of $\mathbf{W}_2^k$ and $\mathbf{b}_1^k$ such that the realization of h_j is invariant. This is due to the fact that $\mathbf{w}_2\sigma(\mathbf{w}_1^\top x + b) = \|\mathbf{w}_1\|\mathbf{w}_2\sigma((\mathbf{w}_1/\|\mathbf{w}_1\|)^\top x + (b/\|\mathbf{w}_1\|))$ for any $\mathbf{w}_1 \neq \mathbf{0} \in \mathbb{R}^{N_0}, \mathbf{w}_2 \in \mathbb{R}^{N_2}, b \in \mathbb{R}$. Neurons corresponding to null rows of $\mathbf{W}_1^k$ are handled similarly, in an iterative manner, by setting them to an arbitrary normalized row, setting the corresponding column of $\mathbf{W}_2^k$ to zero, and changing the bias $\mathbf{b}_2^k$ to keep the function $\mathcal{R}_\theta$ unchanged on $\mathbb{R}^{N_0}$, using that $\mathbf{w}_2\sigma(\mathbf{0}^\top x + b) + \mathbf{b}_2 = \mathbf{0}\sigma(\mathbf{v}^\top x + b) + (\mathbf{b}_2 + \mathbf{w}_2\sigma(b))$ for any normalized vector $\mathbf{v} \in \mathbb{R}^{N_0}$. Thus, we obtain θ' whose matrix of the first layer, $\mathbf{W}'_1$, has normalized rows and $\mathcal{R}_\theta = \mathcal{R}_{\theta'}$ on $\mathbb{R}^{N_0}$.

To construct $\tilde{\theta}$ with $\|\tilde{\mathbf{b}}_1\|_\infty \leq C$ we see that, by definition of C, if $\|\mathbf{w}_1\| = 1$ and $b \geq C$ then

$$\mathbf{w}_1^\top x + b \geq -C + b \geq 0, \qquad \forall x \in \Omega. \tag{7.15}$$

Thus, the function $\mathbf{w}_2\sigma(\mathbf{w}_1^\top x + b) = \mathbf{w}_2(\mathbf{w}_1^\top x + b)$ is linear on Ω and

$$\begin{aligned}\mathbf{w}_2\sigma(\mathbf{w}_1^\top x + b) + \mathbf{b}_2 &= \mathbf{w}_2(\mathbf{w}_1^\top x + C) + ((b - C)\mathbf{w}_2 + \mathbf{b}_2) \\ &= \mathbf{w}_2\sigma(\mathbf{w}_1^\top x + C) + ((b - C)\mathbf{w}_2 + \mathbf{b}_2)\end{aligned}$$

Thus, for any hidden neuron with a bias exceeding C, the bias can be saturated to C by changing accordingly the output bias $\mathbf{b}_2$, keeping the function $\mathcal{R}_\theta$ unchanged *on the bounded domain* Ω (but not on the whole space $\mathbb{R}^{N_0}$). Hidden neurons with a bias $b \leq -C$ can be similarly modified. Sequentially saturating each hidden neuron yields $\tilde{\theta}$ which satisfies all conditions of Lemma 7.6.2. □

Lemma 7.6.3. *Consider Ω a bounded subset of $\mathbb{R}^{N_0}$, for any $\mathbf{I} = (I_2, I_1)$, given a continuous function $f \in \overline{\mathcal{F}_{\mathbf{I}}(\Omega)}$, there exists a sequence $\{\theta^k\}_{k\in\mathbb{N}}, \theta^k = (\mathbf{W}_i^k, \mathbf{b}_i^k)_{i=1}^2 \in \mathcal{N}_{\mathbf{I}}$ such that:*

1. The sequence $\mathcal{R}_{\theta^k}$ admits f as its uniform limit, i.e., $\lim_{k\to\infty} \|\mathcal{R}_{\theta^k} - f\|_\infty = 0$.

2. The sequence $\{(\mathbf{W}_1^k, \mathbf{b}_1^k)\}_{k\in\mathbb{N}}$ has a finite limit $(\mathbf{W}_1^\star, \mathbf{b}_1^\star)$ where $\mathbf{W}_1^\star$ has unit norm rows and $\operatorname{supp}(\mathbf{W}_1^\star) \subseteq I_1$.

Proof. Given a function $f \in \overline{\mathcal{F}_{\mathbf{I}}(\Omega)}$, by definition, there exists a sequence $\{\theta^k\}_{k\in\mathbb{N}}$, $\theta^k \in \mathcal{N}_{\mathbf{I}}$ $\forall k \in \mathbb{N}$ such that $\lim_{k\to\infty} \|\mathcal{R}_{\theta^k} - f\|_\infty = 0$. We can assume that $\mathbf{W}_1^k$ has normalized rows and $\mathbf{b}_1^k$ is bounded using Lemma 7.6.2. We can also assume WLOG that the parameters of the first layer (i.e $\mathbf{W}_1^k, \mathbf{b}_1^k$) have finite limits $\mathbf{W}_1^\star$ and $\mathbf{b}_1^\star$. Indeed, since both $\mathbf{W}_1^k$ and $\mathbf{b}_1^k$ are bounded (by construction from Lemma 7.6.2), there exists a subsequence $\{\varphi_k\}_{k\in\mathbb{N}}$ such that $\mathbf{W}_1^{\varphi_k}$ and $\mathbf{b}_1^{\varphi_k}$ have finite limits and $\mathcal{R}_{\theta^{\varphi_k}} \to f$ as $\mathcal{R}_{\theta^k} \to f$. Replacing the sequence $\{\theta^k\}_{k\in\mathbb{N}}$ by $\{\theta^{\varphi_k}\}_{k\in\mathbb{N}}$ yields the desired sequence. Finally, since $\mathbf{W}_1^\star = \lim_{k\to\infty} \mathbf{W}_1^k$, $\mathbf{W}_1^\star$ obviously has normalized rows and $\operatorname{supp}(\mathbf{W}_1^\star) \subseteq I_1$. □

Definition 7.6.4. Consider Ω bounded subset of $\mathbb{R}^d$, a function $f \in \overline{\mathcal{F}_{\mathbf{I}}(\Omega)}$ and a sequence $\{\theta^k\}_{k\in\mathbb{N}}$ as given by Lemma 7.6.3. We define $(a_i, b_i) = (\mathbf{W}_1^\star[i,:], \mathbf{b}_1^\star[i])$ the limit parameters of the first layer corresponding to the ith neuron. We partition the set of neurons into two subsets (one of them may be empty):

1. Set of active neurons: $J := \{i \mid (\exists x \in \Omega, a_i x + b_i > 0) \wedge (\exists x \in \Omega, a_i x + b_i < 0)\}$.

2. Set of non-active neurons: $\bar{J} = [\![N_1]\!] \setminus J$.

For $i, j \in J$, we write $i \simeq j$ if $(\mathbf{W}_1^\star[j,:], \mathbf{b}_1^\star[j]) = \pm(\mathbf{W}_1^\star[i,:], \mathbf{b}_1^\star[i])$. The relation $\simeq$ is an equivalence relation.

We define $(J_\ell)_{\ell=1,\dots,r}$ the equivalence classes induced by $\simeq$ and we use $(\alpha_\ell, \beta_\ell) := (a_i, b_i)$ for some $i \in J_\ell$ as the representative limit of the ℓth equivalence class. For $i \in J_\ell$, we have: $(a_i, b_i) = \epsilon_i(\alpha_\ell, \beta_\ell), \epsilon_i \in \{\pm 1\}$. We define $J_\ell^+ = \{i \in J_\ell \mid \epsilon_i = 1\} \neq \emptyset$ and $J_\ell^- = J_\ell \setminus J_\ell^+$.

For each equivalence class J_ℓ, define $H_\ell := \{x \in \Omega \mid \alpha_\ell x + \beta_\ell = 0\}$ the boundary generated by neurons in J_ℓ and the positive (resp. negative) half-space partitioned by H_ℓ, $H_\ell^+ := \{x \in \Omega \mid \alpha_\ell x + \beta_\ell > 0\}$ (resp. $H_\ell^- := \{x \in \Omega \mid \alpha_\ell x + \beta_\ell < 0\}$). For any $\epsilon > 0$ we also define the open half-spaces $H_\ell^{(\epsilon,+)} := \{x \in \mathbb{R}^d \mid \alpha_\ell^\top x + \beta_\ell > \epsilon\}$ and $H_\ell^{(\epsilon,-)} := \{x \in \mathbb{R}^d \mid \alpha_\ell^\top x + \beta_\ell < -\epsilon\}$.

Definition 7.6.4 groups neurons sharing the same "linear boundary" (or "singular hyperplane" as in [PRV21]). This concept is related to "twin neurons" [SG22], which also groups neurons with the same active zone. This partition somehow allows us to treat classes independently. Observe also that

$$\operatorname{supp}(\alpha_\ell) \subseteq \bigcap_{i\in J_\ell} I_1[i,:], \forall 1 \leq \ell \leq r. \tag{7.16}$$

Definition 7.6.5 (Contribution of an equivalence class)**.** In the setting of Definition 7.6.4, we define the contribution of the ith neuron $1 \leq i \leq N_1$ (resp. of the ℓth $(1 \leq \ell \leq r)$ equivalence class) of θ^k as:

$$h_i^k : \mathbb{R}^{N_0} \mapsto \mathbb{R}^{N_2} : x \mapsto \mathbf{W}_2^k[:, i]\sigma(\mathbf{W}_1^k[i, :]x + \mathbf{b}_1^k[i]) \ ,$$
$$g_\ell^k : \mathbb{R}^{N_0} \mapsto \mathbb{R}^{N_2} : x \mapsto \sum_{i \in J_\ell} h_i^k(x) \ .$$

Lemma 7.6.6. *Consider $\Omega = [-B, B]^d$, $f \in \overline{\mathcal{F}_{\mathbf{I}}(\Omega)}$ and a sequence $\{\theta^k\}_{k\in\mathbb{N}}$ as given by Lemma 7.6.3, and $\alpha_\ell, \beta_\ell, 1 \leq \ell \leq r, \epsilon_i, i \in J$ as given by Definition 7.6.4. There exist some $\gamma_\ell, \mathbf{b} \in \mathbb{R}^{N_2}, \mathbf{A} \in \mathbb{R}^{N_2 \times N_0}$ such that:*

$$f(x) = \sum_{i=1}^{r} \gamma_\ell \sigma(\alpha_\ell x + \beta_\ell) + \mathbf{A}x + \mathbf{b} \quad \forall x \in \Omega \tag{7.17}$$

$$\lim_{k\to\infty} \sum_{i \in J_\ell} \epsilon_i \mathbf{W}_2^k[:, i]\mathbf{W}_1^k[i, :] = \gamma_\ell \alpha_\ell, \quad \forall 1 \leq \ell \leq r \tag{7.18}$$

$$\lim_{k\to\infty} \sum_{i \in J_\ell} \epsilon_i \mathbf{b}_1^k[i]\mathbf{W}_2^k[:, i] = \gamma_\ell \beta_\ell, \quad \forall 1 \leq \ell \leq r \tag{7.19}$$

$$\operatorname{supp}(\gamma_\ell) \subseteq \bigcup_{i \in J_\ell} I_2[:, i], \quad \forall 1 \leq \ell \leq r \tag{7.20}$$

Proof. The proof is divided into three parts: We first show that there exist $\gamma_\ell, \mathbf{b} \in \mathbb{R}^{N_2}$ and $\mathbf{A} \in \mathbb{R}^{N_2 \times N_0}$ such that Equation (7.17) holds. The last two parts will be devoted to prove that equations (7.18) - (7.20) hold.

1. **Proof of Equation (7.17):** Our proof is based on a result of [PRV21], which deals with the case of a scalar output (i.e, $N_2 = 1$). It is proved in [PRV21, Theorem 3.8, Steps $3, 6, 7$] and states the following:

Lemma 7.6.7 (Analytical form of a limit function with scalar output [PRV21])**.** *In case $N_2 = 1$ (i.e., output dimension equal to one), consider $\Omega = [-B, B]^d$, a scalar-valued function $f : \Omega \mapsto \mathbb{R}, f \in \overline{\mathcal{F}_{\mathbf{I}}(\Omega)}$ and a sequence as given by Lemma 7.6.3, there exist $\mu \in \mathbb{R}^{N_0}, \gamma_\ell, \nu \in \mathbb{R}$ such that:*

$$f(x) = \sum_{\ell=1}^{r} \gamma_\ell \sigma(\alpha_\ell x + \beta_\ell) + \mu^\top x + \nu, \quad \forall x \in \Omega \tag{7.21}$$

Back to our proof, one can write $f = (f_1, \ldots, f_{N_2})$ where $f_j : \Omega \subseteq \mathbb{R}^{N_0} \mapsto \mathbb{R}$ is the function f restricted to the jth coordinate. Clearly, f_j is also a uniform limit on Ω of $\{\mathcal{R}_{\tilde{\theta}^k}\}_{k\in\mathbb{N}}$ for a sequence $\{\tilde{\theta}^k\}_{k\in\mathbb{N}}$ which shares the same $\mathbf{W}_1^k$ with $\{\theta^k\}_{k\in\mathbb{N}}$ but $\tilde{\mathbf{W}}_2^k$ is the jth row of $\mathbf{W}_2^k$. Therefore, $\{\tilde{\theta}^k\}_{k\in\mathbb{N}}$ also satisfies the assumptions of Lemma 7.6.7, which gives us:

$$f_j(x) = \sum_{\ell=1}^{r} \gamma_{\ell,j} \sigma(\alpha_\ell x + \beta_\ell) + \mu_j^\top x + \nu_j, \quad \forall x \in \Omega$$

for some $\mu_j \in \mathbb{R}^{N_0}, \gamma_{i,j}, \nu_j \in \mathbb{R}$. Note that α_ℓ, β_ℓ and r are not dependent on the index j since they are defined directly from the considered sequence. Therefore, the function f (which is the concatenation of f_j coordinate by coordinate) is:

$$f(x) = \sum_{\ell=1}^{r} \gamma_\ell \sigma(\alpha_\ell x + \beta_\ell) + \mathbf{A}x + \mathbf{b}, \quad \forall x \in \Omega$$

with $\gamma_\ell = \begin{pmatrix} \gamma_{i,1} \\ \vdots \\ \gamma_{i,N_2} \end{pmatrix}, \mathbf{A} = \begin{pmatrix} \mu_1^\top \\ \vdots \\ \mu_{N_2}^\top \end{pmatrix}, \mathbf{b} = \begin{pmatrix} \nu_1 \\ \vdots \\ \nu_{N_2} \end{pmatrix}$.

2. **Proof for Equations** (7.18)-(7.19): With the construction of γ, we will prove Equation (7.18) and Equation (7.19). We consider an arbitrary $1 \leq \ell \leq r$. Denoting Ω° the interior of Ω and $H_\ell := \{x \in \Omega \mid \alpha_\ell x + \beta_\ell = 0\}$ the hyperplane defined by the input weights and bias of the ℓ-th class of neurons, we take a point $x' \in (\Omega^\circ \cap H_\ell) \setminus \bigcup_{p \neq \ell} H_p$ and a fixed scalar $r > 0$ such that the open ball $\mathcal{B}(x', r) \subseteq \Omega^\circ \setminus \bigcup_{p \neq \ell} H_p$. Notice that x' is well-defined due to the definition of J (Definition 7.6.4). In addition, r also exists because $\Omega^\circ \setminus \bigcup_{p \neq \ell} H_p$ is an open set. Thus, there exists two constants $0 < \delta < B$ and $\epsilon > 0$ such that:

(a) $\mathcal{B}(x', r) \subseteq [-(B - \delta), B - \delta]^d$.

(b) For each $p \neq \ell$, the ball $\mathcal{B}(x', r)$ is either included in the half-space $H_p^{(\epsilon,+)} := \{x \in \mathbb{R}^d \mid \alpha_p^\top x + \beta_p > \epsilon\}$ or in the half-space $H_p^{(\epsilon,-)} := \{x \in \mathbb{R}^d \mid \alpha_p^\top x + \beta_p < -\epsilon\}$.

(c) The intersection of $\mathcal{B}(x', r)$ with $H_\ell^{(\epsilon,+)}$ and $H_\ell^{(\epsilon,-)}$ are not empty.

For the remaining of the proof, we will use Lemma 7.6.8, another result taken from [PRV21]. We only state the lemma. Its formal proof can be found in the proof of [PRV21, Theorem 3.8, Steps 4-5].

Lemma 7.6.8 (Affine linear area [PRV21])**.** *Given a sequence $\{\theta^k\}_{k \in \mathbb{N}}$ satisfying the second condition of Lemma 7.6.3, we have:*

(a) For any $0 < \delta < B$, there exists a constant κ_δ such that $\forall i \in \bar{J}$, h_i^k are affine linear on $[-(B - \delta), B - \delta]^{N_0}$ for all $k \geq \kappa_\delta$.

(b) For any $\epsilon > 0$, there exists a constant κ_ϵ such that for each $1 \leq \ell \leq r$ and each $i \in J_\ell$ the function h_i^k is affine linear on $H_\ell^{(\epsilon,+)} \cup H_\ell^{(\epsilon,-)}$ for all $k \geq \kappa_\epsilon$.

The lemma implies the existence of $K = \max(\kappa_\delta, \kappa_\epsilon)$ such that for all $k \geq K$, we have:

$$\sum_{p \neq \ell} g_p^k(x) = \mathbf{B}^k x + \nu^k, \qquad \forall x \in \mathcal{B}(x', r),$$

for some $\mathbf{B}^k \in \mathbb{R}^{N_2 \times N_0}, \nu^k \in \mathbb{R}^{N_2}$. Therefore, for $k \geq K$, we have:

$$\mathcal{R}_{\theta^k}(x) = \mathbf{B}^k x + \nu^k + \sum_{i \in J_\ell^+} \mathbf{W}_2^k[:, i](\mathbf{W}_1^k[i, :]x + \mathbf{b}_1^k[i]), \forall x \in \mathcal{B}(x', r) \cap H_\ell^{(\epsilon,+)}$$

$$\mathcal{R}_{\theta^k}(x) = \mathbf{B}^k x + \nu^k + \sum_{i \in J_\ell^-} \mathbf{W}_2^k[:, i](\mathbf{W}_1^k[i, :]x + \mathbf{b}_1^k[i]), \forall x \in \mathcal{B}(x', r) \cap H_\ell^{(\epsilon,-)}.$$

Since we proved that f has the form Equation (7.17), there exist $\mathbf{C} \in \mathbb{R}^{N_2 \times N_0}, \mu \in \mathbb{R}^{N_2}$ such that

$$\begin{aligned} f(x) &= (\mathbf{C} + \gamma_\ell \alpha_\ell)x + (\mu + \gamma_\ell \beta_\ell), \quad \forall x \in \mathcal{B}(x', r) \cap H_\ell^{(\epsilon,+)} \\ f(x) &= \mathbf{C}x + \mu, \qquad\qquad\qquad\qquad \forall x \in \mathcal{B}(x', r) \cap H_\ell^{(\epsilon,-)} \end{aligned}$$

As both $\mathcal{B}(x', r) \cap H_\ell^{(\epsilon,+)}$ and $\mathcal{B}(x', r) \cap H_\ell^{(\epsilon,-)}$ are open sets, and given our hypothesis of uniform convergence of $\mathcal{R}_{\theta^k} \to f$, we obtain,

$$\begin{aligned} \lim_{k\to\infty} \mathbf{B}^k + \sum_{i \in J_\ell^+} \mathbf{W}_2^k[:, i]\mathbf{W}_1^k[i, :] &= \mathbf{C} + \gamma_\ell \alpha_\ell \\ \lim_{k\to\infty} \mathbf{B}^k + \sum_{i \in J_\ell^-} \mathbf{W}_2^k[:, i]\mathbf{W}_1^k[i, :] &= \mathbf{C} \\ \lim_{k\to\infty} \nu^k + \sum_{i \in J_\ell^+} \mathbf{b}_1^k[i]\mathbf{W}_2^k[:, i] &= \mu + \gamma_\ell \beta_\ell \\ \lim_{k\to\infty} \nu^k + \sum_{i \in J_\ell^-} \mathbf{b}_1^k[i]\mathbf{W}_2^k[:, i] &= \mu. \end{aligned} \tag{7.22}$$

Proof for Equation (7.22) can be found in Appendix B.2.2. Equations (7.18) and (7.19) follow directly from Equation (7.22).

3. **Proof of Equation (7.20)**: Since $\alpha_\ell \neq 0$ (remember that $\|\alpha_\ell\| = 1$), this is an immediate consequence of Equation (7.18) as each vector $\mathbf{W}_2^k[:, j]$, $j \in J_\ell$ is supported in $I_2[:, j] \subseteq \cup_{i \in J_\ell} I_2[:, i]$. □

We state an immediate corollary of Lemma 7.6.6, which characterizes the limit of the sequence of contributions $\{g_\ell^k\}_{k\in\mathbb{N}}$ of the ℓth equivalence class with $|J_\ell| = 1$.

Corollary 7.6.9. *Consider $f \in \overline{\mathcal{F}_{\mathbf{I}}([-B, B]^d)}$ that admits the analytical form in Equation (7.17), a sequence $\{\theta^k\}_{k\in\mathbb{N}}$ as given by Lemma 7.6.3, and Definition 7.6.4. For all singleton equivalence classes $J_\ell = \{i\}, 1 \leq \ell \leq r$, we have $\lim_{k\to\infty} \mathbf{W}_2^k[:, i] = \gamma_\ell$ and $\lim_{k\to\infty} \|h_\ell^k - \gamma_\ell \sigma(\alpha_\ell^\top x + \beta_\ell)\|_\infty = 0$.*

Proof. We first prove that $\mathbf{W}_2^k[:, i]$ has a finite limit. In fact, applying the second point of Lemma 7.6.6 for $J_\ell = \{i\}$, we have:

$$\lim_{k\to\infty} \mathbf{W}_2^k[:, i]\mathbf{W}_1^k[i, :] = \gamma_\ell \alpha_\ell$$

where γ_ℓ, α_ℓ are defined in Lemma 7.6.6. Because $\lim_{k\to\infty} \mathbf{W}_1^k[i, :] = \alpha_\ell$ and $\|\alpha_\ell\|_2 = 1$, it follows that $\gamma_\ell = \lim_{k\to\infty} \mathbf{W}_2^k[:, i]$. To conclude, since we also have $\beta_\ell = \lim_{k\to\infty} \mathbf{b}_1^k[i]$, we obtain $h_\ell^k(\cdot) = \mathbf{W}_2^k[\ell, :]\sigma(\mathbf{W}_1^k[\ell, :] \cdot + \mathbf{b}_1^k[\ell]) \to \gamma_\ell \sigma(\alpha_\ell x + \beta_\ell)$ as claimed. □

The nice thing about Corollary 7.6.9 is that the contribution $g_\ell^k = h_\ell^k$ admits a (uniform) limit if $J_\ell = \{i\}$. Moreover, this limit is even implementable by using only the ith neuron because $\text{supp}(\alpha_\ell) \subseteq I_1[i, :]$ and $\text{supp}(\gamma_\ell) \subseteq I_2[:, i]$.

It would be tempting to believe that, for each $P \in \{\bar{J}\} \cup \{J_\ell \mid \ell = 1, \ldots, r\}$ the sequence of functions $\sum_{i\in P} g_i^k(x)$ must admit a limit (when k tends to ∞) and that this

limit is implementable using only neurons in P. This would obviously imply that $\mathcal{F}_{\mathbf{I}}(\Omega)$ is closed. This intuition is however wrong. For non-singleton equivalence class (i.e., for cases *not* covered by Corollary 7.6.9), the limit function *does not necessarily exist* as we show in the following example.

Example 7.6.10. Consider the case where $\mathbf{N} = (1, 3, 1)$ and no support constraint, $\Omega = [-1, 1]$, take the sequence $\{\theta^k\}_{k\in\mathbb{N}}$ which satisfies:

$$\mathbf{W}_1^k = \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix}, \mathbf{b}_1^k = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \mathbf{W}_2^k = \begin{pmatrix} k & -k & -k \end{pmatrix}, \mathbf{b}_2^k = k$$

Then for $x \in \Omega$, it is easy to verify that $\mathcal{R}_{\theta^k} = 0$. Indeed,

$$\begin{aligned} \mathcal{R}_{\theta^k}(x) &= \sum_{i=1}^{3} \mathbf{W}_2^k[:, i]\sigma(\mathbf{W}_1^k[i, :] + \mathbf{b}_1^k[i]) + \mathbf{b}_2^k \\ &= k\sigma(x) - k\sigma(-x) - k\sigma(x + 1) + k \\ &= k(\sigma(x) - \sigma(-x)) - k(x + 1) + k \quad (\text{since } x + 1 \geq 0, \forall x \in \Omega) \\ &= kx - k(x + 1) + k = 0 \end{aligned}$$

Thus, this sequence converges (uniformly) to $f = 0$. Moreover, this sequence also satisfies the assumptions of Lemma 7.6.3. Using the classification in Definition 7.6.4, we have one class equivalence $J_1 = \{1, 2\}$ and $\bar{J} = \{3\}$. The function $g_1^k(x) = k\sigma(x) - k\sigma(-x) = kx$, however, does not have any limit.

Actual proof of Theorem 7.4.2

Therefore, our analysis cannot treat each equivalence class entirely separately. The last result in this section is about a property of the matrix $\mathbf{A}$ in Equation (7.17). This is one of our key technical contributions in this work.

Lemma 7.6.11. *Consider $\Omega = [-B, B]^d, f \in \overline{\mathcal{F}_{\mathbf{I}}(\Omega)}$ that admits the analytical form in Equation (7.17), a sequence $\{\theta^k\}_{k\in\mathbb{N}}$ as given by Lemma 7.6.3, then the matrix $\mathbf{A} \in \overline{\mathcal{L}_{\mathbf{I}'}}$ where $\mathbf{I}' = (I_2[:, S], I_1[S, :]), S = \bar{J} \cup (\cup_{1\leq\ell\leq r} J_\ell^-)$, $\bar{J}, J_\ell^\pm$ are defined as in Definition 7.6.4).*

Combining Lemma 7.6.11 and the assumptions of Theorem 7.4.2, we can prove Theorem 7.4.2 immediately as follow:

Proof of Theorem 7.4.2. Consider $f \in \overline{\mathcal{F}_{\mathbf{I}}(\Omega)}$, we deduce that there exists a sequence of $\{\theta^k\}_{k\in\mathbb{N}}$ that satisfies the properties of Lemma 7.6.3. This allows us to define $\bar{J}$ and equivalence classes $J_\ell, 1 \leq \ell \leq r$ as well as $(\alpha_\ell, \beta_\ell)$ as in Definition 7.6.4. Using Lemma 7.6.6, we can also deduce an analytical formula for f as in Equation (7.17):

$$f(x) = \sum_{i=1}^{r} \gamma_\ell \sigma(\alpha_\ell x + \beta_\ell) + \mathbf{A}x + \mathbf{b}, \quad \forall x \in \Omega.$$

Finally, Lemma 7.6.11 states that matrix $\mathbf{A}$ in Equation (7.17) satisfies: $\mathbf{A} \in \overline{\mathcal{L}_{\mathbf{I}'}}$ with $\mathbf{I}' = (I_2[:, S], I_1[S, :]])$, where $S = \bar{J} \cup (\cup_{\ell=1}^{r} J_\ell^-)$. To prove that $f \in \mathcal{F}_{\mathbf{I}}$, we construct the parameters $\theta = \{(\mathbf{W}_i, \mathbf{b}_i)_{i=1}^2\}$ of the limit network as follows:

1. For each $1 \leq \ell \leq r$, choose one index $j \in J_\ell^+$ (which is possible since J_ℓ^+ is non-empty). We set:

$$(\mathbf{W}_1[i,:], \mathbf{W}_2[:,i], \mathbf{b}_1[i]) = \begin{cases} (\alpha_\ell, \gamma_\ell, \beta_\ell) & \text{if } i = j \\ (\alpha_\ell, \mathbf{0}, \beta_\ell) & \text{otherwise} \end{cases} \tag{7.23}$$

This satisfies the support constraint because $\text{supp}(\alpha_\ell) \subseteq I_1[j,:]$ (by (7.16)) $\alpha_\ell = \lim_{k\to\infty} \mathbf{W}_1^k[j,:])$ and $I_2 = \mathbf{1}_{N_2 \times N_1}$. This is where we use the first assumption of Theorem 7.4.2. Without it, $\text{supp}(\gamma_\ell)$ might not be a subset of $I_2[:,j]$.

2. For $i \in S$: Since $\mathbf{A} \in \overline{\mathcal{L}_{\mathbf{I}'}}$ (cf Lemma 7.6.11) and $\mathcal{L}_{\mathbf{I}'}$ is closed (second assumptions of Theorem 7.4.2), there exist two matrices $\hat{\mathbf{W}}_1, \hat{\mathbf{W}}_2$ such that: $\text{supp}(\hat{\mathbf{W}}_1) \subseteq I_1[:,S], \text{supp}(\hat{\mathbf{W}}_2) \subseteq I_2[S,:]$, and $\mathbf{A} = \hat{\mathbf{W}}_2\hat{\mathbf{W}}_1$. We set:

$$(\mathbf{W}_1[i,:], \mathbf{W}_2[:,i], \mathbf{b}_1[i]) = (\hat{\mathbf{W}}_1[i,:], \hat{\mathbf{W}}_2[:,i], C) \tag{7.24}$$

where $C = \sup_{x\in\Omega} \|\hat{\mathbf{W}}_1 x\|_\infty$. This satisfies the support constraints $\mathbf{I}$ due to our choice of $\hat{\mathbf{W}}_1, \hat{\mathbf{W}}_2$. The choice of C ensures that the function $h_i(x) := \mathbf{W}_2[i,:]\sigma(\mathbf{W}_1[i,:]x + \mathbf{b}_1[i])$ is *linear* on Ω.

3. For $\mathbf{b}_2$: Let $\mathbf{b}_2 = \mathbf{b} - C\left(\sum_{i\in S} \hat{\mathbf{W}}_2[:,i]\right)$ ($\mathbf{b}$ is the bias in Equation (7.17)).

Verifying $\mathcal{R}_\theta = f$ on Ω is thus trivial since:

$$\begin{aligned}
\mathcal{R}_\theta(x) &= \sum_{i=1}^{N_1} \mathbf{W}_2[:,i]\sigma(\mathbf{W}_1[i,:]x + \mathbf{b}_1[i]) + \mathbf{b}_2 \\
&= \sum_{i\notin S} \mathbf{W}_2[:,i]\sigma(\mathbf{W}_1[i,:]x + \mathbf{b}_1[i]) + \sum_{i\in S} \mathbf{W}_2[:,i]\sigma(\mathbf{W}_1[i,:]x + \mathbf{b}_1[i]) + \mathbf{b}_2 \\
&= \sum_{\ell=1}^{r} \gamma_\ell\sigma(\alpha_\ell x + \beta_\ell) + \sum_{j\in S} \hat{\mathbf{W}}_2[:,i](\hat{\mathbf{W}}_1[i,:]x + C) + \mathbf{b} - C\left(\sum_{i\in S} \hat{\mathbf{W}}_2[:,i]\right) \\
&= \sum_{\ell=1}^{r} \gamma_\ell\sigma(\alpha_\ell x + \beta_\ell) + \hat{\mathbf{W}}_2\hat{\mathbf{W}}_1 x + \mathbf{b} = \sum_{\ell=1}^{r} \gamma_\ell\sigma(\alpha_\ell x + \beta_\ell) + \mathbf{A}x + \mathbf{b} = f. \quad \square
\end{aligned}$$

Proof of Lemma 7.6.11. In this proof, we define $\Omega_\delta^\circ = (-B + \delta, B - \delta)^{N_0}, 0 < \delta < B$. The choice of δ is not important in this proof (any $0 < \delta < B$ will do).

The proof of this lemma revolves around the following idea: We will construct a sequence of functions $\{f^k\}_{k\in\mathbb{N}}$ such that, for k large enough, f^k has the following analytical form:

$$f^k(x) = \sum_{\ell=1}^{r} \gamma_\ell\sigma(\alpha_\ell x + \beta_\ell) + \mathbf{A}^k x + \mathbf{b}^k, \forall x \in \Omega_\delta^\circ \tag{7.25}$$

and $\lim_{k\to\infty} f^k(x) = f(x)\ \forall x \in \Omega \setminus (\cup_{\ell=1}^r H_\ell)$ (or equivalently f^k converges pointwise to f on $\Omega \setminus (\cup_{\ell=1}^r H_\ell)$) and $\mathbf{A}^k$ admits a factorization into two factors $\mathbf{A}^k = \mathbf{X}^k\mathbf{Y}^k$ satisfying $\text{supp}(\mathbf{X}^k) \subseteq I_2[:,S], \text{supp}(\mathbf{Y}^k) \subseteq I_1[S,:]$, so that $\mathbf{A}^k \in \mathcal{L}_{\mathbf{I}'}$. Comparing Equation (7.17) and Equation (7.25), we deduce that the sequence of affine functions $\mathbf{A}^k x + \mathbf{b}^k$ converges pointwise to the affine function $\mathbf{A}x + \mathbf{b}$ on the open set $\Omega_\delta^\circ \setminus (\cup_{\ell=1}^r H_\ell)$. Therefore, $\lim_{k\to\infty} \mathbf{A}^k = \mathbf{A}$ by Lemma B.2.2 (whose statement and proof can be found in Appendix B.2.2), hence the conclusion.

The rest of this proof is devoted to the construction of $f^k = \mathcal{R}_{\tilde{\theta}^k}$ where $\tilde{\theta}^k \in \mathcal{N}_{\mathbf{N}}$ are parameters of a neural network of the same dimension as those in $\mathcal{N}_{\mathbf{I}}$ but only *partially* satisfying the support constraint **I**. To guarantee that f^k converges pointwise to f, we construct $\tilde{\theta}^k$ based on θ^k and harness their relation.

Choice of parameters. We set $\tilde{\theta}^k = \{(\tilde{\mathbf{W}}_i^k, \tilde{\mathbf{b}}_i^k)_{i=1}^2\} \in \mathcal{N}_{\mathbf{N}}$ where $\tilde{\mathbf{W}}_2^k \in \mathbb{R}^{N_2 \times N_1}$, $\tilde{\mathbf{W}}_1^k \in \mathbb{R}^{N_1 \times N_0}$ are defined as follows, where we use $C^k \coloneqq \sup_{x\in\Omega} \|\mathbf{W}_1^k x\|_\infty$:

- For inactive neurons $i \in \bar{J}$, we simply set $(\tilde{\mathbf{W}}_1^k[:,i], \tilde{\mathbf{W}}_2^k[i,:], \tilde{\mathbf{b}}_1^k[i]) = (\mathbf{W}_1^k[:,i], \mathbf{W}_2^k[i,:], \mathbf{b}_1^k[i])$.

- For each equivalence class of active neurons $1 \le \ell \le r$, we choose some $j_\ell \in J_\ell^+$ (note that J_ℓ^+ is non-empty due to Definition 7.6.4) and set the parameters $(\tilde{\mathbf{W}}_2^k[:,i], \tilde{\mathbf{W}}_1^k[i,:], \mathbf{b}_1[i]), i \in J_\ell$ as:

$$(\tilde{\mathbf{W}}_1^k[i,:], \tilde{\mathbf{W}}_2^k[:,i], \tilde{\mathbf{b}}_1^k[i]) = \begin{cases} (\mathbf{W}_1^k[i,:], \mathbf{W}_2^k[:,i], C^k), & \forall j \in J_\ell^- \\ (\mathbf{W}_1^k[i,:], \mathbf{0}, C^k), & \forall i \in J_\ell^+ \setminus \{j_\ell\} \\ (\alpha_\ell, \gamma_\ell, \beta_\ell), & i = j_\ell \end{cases} \tag{7.26}$$

For $i \in J_\ell \setminus \{j_\ell\}$, we clearly have: $\operatorname{supp}(\tilde{\mathbf{W}}_1^k[i,:]) \subseteq I_1[i,:]$ and $\operatorname{supp}(\tilde{\mathbf{W}}_2^k[:,i]) \subseteq I_2[:,i]$. The j_ℓ-th column of $\tilde{\mathbf{W}}_2^k$ is the only one that does not necessarily satisfy the support constraint, as $\operatorname{supp}(\gamma_\ell) \nsubseteq I_2[:,j_\ell]$ in general.

- Finally, the output bias $\mathbf{b}_2^k$ is set as:

$$\tilde{\mathbf{b}}_2^k \coloneqq \mathbf{b}_2^k + \sum_{\ell=1}^{r} \underbrace{\sum_{i\in J_\ell^-} (\mathbf{b}_1^k[i] - C^k)\mathbf{W}_2^k[:,i]}_{=:\xi_\ell^k} \tag{7.27}$$

Proof that $f^k \coloneqq R_{\tilde{\theta}^k}$ converges pointwise to f on $\Omega \setminus (\cup_{\ell=1}^r H_\ell)$. We introduce notations analog to Definition 7.6.5: for every $x \in \mathbb{R}^{N_0}$ we define:

$$\tilde{h}_i^k(x) = \tilde{\mathbf{W}}_2^k[:,i]\sigma(\tilde{\mathbf{W}}_1^k[i,:]x + \tilde{\mathbf{b}}_1^k[i]), \quad i = 1,\dots,N_1; \qquad \tilde{g}_\ell^k(x) = \sum_{i\in J_\ell} \tilde{h}_i^k(x), \quad \ell = 1,\dots,r$$

By construction

$$\tilde{h}_i^k = h_i^k, \quad \forall i \in \bar{J},\ \forall k, \tag{7.28}$$

and we further explicit the form of $\tilde{h}_i^k, i \in J_\ell$ for $x \in \Omega$ (but *not* on $\mathbb{R}^{N_0}$) as:

$$\tilde{h}_i^k(x) = \begin{cases} \mathbf{W}_2^k[:,i](\mathbf{W}_1^k[i,:]x + C^k), & \forall i \in J_\ell^- \\ 0, & \forall i \in J_\ell^+ \setminus \{j_\ell\}\ , \\ \gamma_\ell\sigma(\alpha_\ell x + \beta_\ell), & i = j_\ell \end{cases} \tag{7.29}$$

We justify our formula in Equation (7.29) as follow:

1. For $i \in J_\ell^-$: since $C^k = \sup_{x\in\Omega} \|\mathbf{W}_1^k x\|_\infty$ by construction, $\tilde{\mathbf{W}}_1^k[i,:]x + \mathbf{b}_1^k[i] = \mathbf{W}_1^k[i,:]x + \mathbf{b}_1^k[i] \ge 0$. The activation σ acts simply as an identity function.

2. For $i \in J_\ell^+$: Because we choose $\tilde{\mathbf{W}}_2^k[:,i] = \mathbf{0}$.

3. For $i = j_\ell$: Obvious due to the construction in Equation (7.26).

Given $x \in \Omega \setminus (\cup_{\ell=1}^r H_\ell)$, we now prove that this construction ensures that for each $\ell \in \{1, \dots, r\}$

$$\lim_{k\to\infty} (\tilde{g}_\ell^k(x) - g_\ell^k(x) + \xi_\ell^k) = 0. \tag{7.30}$$

This will imply the claimed poinwise convergence since

$$\begin{aligned}
\lim_{k\to\infty} f^k(x) = \lim_{k\to\infty} R_{\tilde{\theta}^k}(x) &= \lim_{k\to\infty} \left(\sum_{i\in\bar{J}} \tilde{h}_i^k(x) + \sum_{\ell=1}^r \tilde{g}_\ell^k(x) + \tilde{\mathbf{b}}_2^k \right) \\
&\overset{(7.28)\&(7.30)}{=} \lim_{k\to\infty} \left(\sum_{i\in\bar{J}} h_i^k(x) + \sum_{\ell=1}^r g_\ell^k(x) - \sum_{\ell=1}^r \xi_\ell^k + \tilde{\mathbf{b}}_2^k \right) \\
&\overset{(7.27)}{=} \lim_{k\to\infty} \left(\sum_{i\in\bar{J}} h_i^k(x) + \sum_{\ell=1}^r g_\ell^k(x) + \mathbf{b}_2^k \right) = \lim_{k\to\infty} R_{\theta^k}(x) = f(x).
\end{aligned}$$

To establish (7.30), observe that as $x \in \Omega \setminus (\cup_{\ell=1}^r H_\ell)$ we have $x \notin H_\ell$. We thus distinguish two cases:

Case $x \in H_\ell^-$.

Using (7.26) we show below that for k large enough and $x \in H_\ell^-$, we have

$$\tilde{h}_i^k(x) - h_i^k(x) = \begin{cases} (C^k - \mathbf{b}_1^k[i])\mathbf{W}_2^k[:,i], & i \in J_\ell^- \\ 0, & i \in J_\ell^+ \end{cases} \tag{7.31}$$

and thus

$$\tilde{g}_\ell^k(x) - g_\ell^k(x) + \xi_\ell^k = \sum_{i\in J_\ell} \left(\tilde{h}_i^k(x) - h_i^k(x) \right) + \xi_\ell^k = \sum_{i\in J_\ell^-} (C^k - \mathbf{b}_1^k[i])\mathbf{W}_2^k[:,i] + \xi_\ell^k = 0.$$

We indeed obtain (7.31) as follows. Since $x \in H_\ell^-$, $\alpha_\ell x + \beta_\ell < 0$, i.e., $-\alpha_\ell x - \beta_\ell > 0$. Therefore, given the definitions of $J_\ell^\pm$ (cf Definition 7.6.4) we have:

• For $i \in J_\ell^-$: $\lim_{k\to\infty}(\mathbf{W}_1^k[i,:], \mathbf{b}_1^k[i]) = -(\alpha_\ell, \beta_\ell)$, hence for k large enough, we have $\mathbf{W}_1^k[i,:]x + \mathbf{b}_1^k[i] > 0$ so that $\sigma(\mathbf{W}_1^k[i,:]x + \mathbf{b}_1^k[i]) = \mathbf{W}_1^k[i,:]x + \mathbf{b}_1^k[i]$ and, as expressed in (7.31):

$$\tilde{h}_i^k(x) - h_i^k(x) \overset{(7.29)}{=} \mathbf{W}_2^k[:,i](\mathbf{W}_1^k[i,:]x + C^k) - \mathbf{W}_2^k[:,i](\mathbf{W}_1^k[i,:]x + \mathbf{b}_1^k[i]) = (C^k - \mathbf{b}_1^k[i])\mathbf{W}_2^k[:,i].$$

• For $i \in J_\ell^+$: similarly, we have $\mathbf{W}_1^k[i,:]x + \mathbf{b}_1^k[i] < 0$ for k large enough. Therefore, $h_i^k(x) = 0$ for k large enough. The fact that we also have $\tilde{h}_i^k(x) = 0$ is immediate from Equation (7.29) if $i \neq j_\ell$, and for $i = j_\ell$ we also get from Equation (7.29) that $\tilde{h}_i^k(x) = \gamma_\ell \sigma(\alpha_\ell x + \beta_\ell) = 0$ since $\alpha_\ell x + \beta_\ell < 0$.

Case $x \in H_\ell^+$. An analog to Equation (7.31) for $x \in H_\ell^+$ is

$$\tilde{h}_i^k(x) - h_i^k(x) = \begin{cases} \mathbf{W}_2^k[:,i](\mathbf{W}_1^k[i,:]x + C^k), & i \in J_\ell^- \\ -\mathbf{W}_2^k[:,i](\mathbf{W}_1^k[i,:]x + \mathbf{b}_1^k[i]), & i \in J_\ell^+ \setminus \{j\} \\ \gamma_\ell(\alpha_\ell x + \beta_\ell) - \mathbf{W}_2^k[:,i](\mathbf{W}_1^k[i,:]x + \mathbf{b}_1^k[i]), & i = j \end{cases}. \tag{7.32}$$

We establish it before concluding for this case.

• For $i \in J_\ell^-$: by a reasoning analog to the case $x \in H_\ell^-$, we deduce that for k large enough

$$\tilde{h}_i^k(x) - h_i^k(x) \overset{(7.29)}{=} \mathbf{W}_2^k[:,i](\mathbf{W}_1^k[i,:]x + C^k).$$

• For $i \in J_\ell^+$: a similar reasoning yields $h_i^k(x) = \mathbf{W}_2^k[:,i](\mathbf{W}_1^k[i,:]x + \mathbf{b}_1^k[i])$ for k large enough, while Equation (7.29) yields $\tilde{h}_{j_\ell}^k(x) = \gamma_\ell\sigma(\alpha_\ell x+\beta_\ell) = \gamma_\ell(\alpha_\ell x+\beta_\ell)$ (since $\alpha_\ell x+\beta_\ell > 0$ as $x \in H_\ell^+$) and $\tilde{h}_i^k(x) = 0$ if $i \neq j_\ell$.

Using (7.32) we obtain for k large enough

$$\begin{aligned}
\tilde{g}_\ell^k(x) - g_\ell^k(x) + \xi_\ell^k &= \sum_{i\in J_\ell}\Big(\tilde{h}_i^k(x) - h_i^k(x)\Big) + \xi_\ell^k \\
&= \sum_{i\in J_\ell^-} \mathbf{W}_2^k[:,i](\mathbf{W}_1^k[i,:]x + C^k) - \sum_{i\in J_\ell^+} \mathbf{W}_2^k[:,i](\mathbf{W}_1^k[i,:]x + \mathbf{b}_1^k[i]) + \gamma_\ell(\alpha_\ell x + \beta_\ell) + \xi_\ell^k \\
&\overset{(7.27)}{=} \Big(\sum_{i\in J_\ell^-} \mathbf{W}_2^k[:,i]\mathbf{W}_1^k[i,:] - \sum_{j\in J_\ell^+} \mathbf{W}_2^k[:,i]\mathbf{W}_1^k[i,:] + \gamma_\ell\alpha_\ell\Big)x \\
&\quad + \Big(\underbrace{\xi_\ell^k + \sum_{i\in J_\ell^-} \mathbf{W}_2^k[:,i]C^k}_{\sum_{i\in J_\ell^-}\mathbf{W}_2^k[:,i]\mathbf{b}_1^k[i]} - \sum_{i\in J_\ell^+} \mathbf{W}_2^k[:,i]\mathbf{b}_1^k[i] + \gamma_\ell\beta_\ell\Big)
\end{aligned}$$

where in the last line we used the expression of ξ_ℓ^k from (7.27). Due to Equations (7.18) and (7.19) it follows that $\lim_{k\to\infty} \tilde{g}_\ell^k(x) - g_\ell^k(x) + \xi_\ell^k = 0$, $\forall x \in H_\ell^+$.

Thus combining both cases, we conclude that $\lim_{k\to\infty} \tilde{g}_\ell^k(x) - g_\ell^k(x) + \xi_\ell^k = 0, \forall x \notin H_\ell$, as desired.

Proof of the expression (7.25) with $\mathbf{A}^k \in \mathcal{L}_{\mathbf{I}'}$ for large enough k. From (7.29), we first deduce that

$$f^k(x) = \sum_{i=1}^{N_1} \tilde{h}_i^k(x) + \tilde{\mathbf{b}}_2^k = \sum_{\ell=1}^{r} \gamma_\ell\sigma(\alpha_\ell x + \beta_\ell) + \sum_{i\in S} \tilde{h}_i^k(x) + \tilde{\mathbf{b}}_2^k, \quad \forall x \in \mathbb{R}^{N_0}.$$

where we recall that $S \coloneqq \bar{J} \cup (\cup_{1\leq\ell\leq r} J_\ell^-)$. There only remains to show that, for k large enough, we have $\sum_{i\in S} \tilde{h}_i^k(x) = \mathbf{A}^k x + \mathbf{b}^k$ for every x *in the restricted domain* Ω_δ°, where $\mathbf{A}^k \in \mathcal{L}_{\mathbf{I}'}$ and $\mathbf{b}^k \in \mathbb{R}^{N_2}$. Note that for $i \in J_\ell$, our construction assures that $\tilde{h}_i^k$ is affine on Ω. Moreover, in the restricted domain Ω_δ°, for $k \geq \kappa_\delta$ large enough, $\tilde{h}_i^k, i \in \bar{J}$ also behave like affine functions (cf Lemma 7.6.8). Therefore,

$$\sum_{i\in S} \tilde{h}_i^k(x) = \Big(\sum_{i\in S} \delta_i^k \tilde{\mathbf{W}}_2^k[:,i]\tilde{\mathbf{W}}_1^k[i,:]\Big)x + \mathbf{c}^k, \quad \forall x \in \Omega_\delta^\circ, k \geq \kappa_\delta$$

for some vector $\mathbf{c}^k$ and binary scalars δ_i^k. In fact, $\delta_i^k = 0$ if $i \in \bar{J}^- \coloneqq \{j \in \bar{J} \mid \mathbf{W}_1^\star[j,:]x + \mathbf{b}_1^\star[j] \leq 0, \forall x \in \Omega\}$ and $\delta_k^i = 1$ otherwise. Thus, one chooses $\mathbf{A}^k = \sum_{i\in S} \delta_i^k \tilde{\mathbf{W}}_2^k[:,i]\tilde{\mathbf{W}}_1^k[i,:], \mathbf{b}^k = \mathbf{c}^k$ and the construction is complete. This construction allows us to write $\mathbf{A}^k = \hat{\mathbf{W}}_2^k\hat{\mathbf{W}}_1^k$ with:

$$\begin{aligned}
\hat{\mathbf{W}}_1^k &= \tilde{\mathbf{W}}_1^k[S,:] \\
\hat{\mathbf{W}}_2^k &= \tilde{\mathbf{W}}_2^k[:,S]\texttt{diag}(\{\nu_i^k \mid i = 1,\ldots,N_1\})
\end{aligned}$$

where $\texttt{diag}(\{\nu_i^k \mid i = 1, \ldots, N_1\}) \in \mathbb{R}^{N_1 \times N_1}$ is a diagonal matrix, $\nu_i^k = \delta_i^k$ for $i \in S$ and 0 otherwise. It is also evident that $\text{supp}(\hat{\mathbf{W}}_2^k[:, S]) \subseteq I_2[:, S], \text{supp}(\hat{\mathbf{W}}_1^k[S, :]) \subseteq I_1[S, :]$. (since the multiplication with a diagonal matrix does not increase the support of a matrix). This concludes the proof. □

Conclusion

In this chapter, we study the somewhat overlooked question of the existence of an optimal solution to sparse neural networks training problems. The study is accomplished by adopting the point of view of topological properties of the function spaces of such networks on two types of domains: a finite domain Ω, or (typically) a hypercube. On the one hand, our investigation of the BAP and the closedness of these function spaces reveals the existence of *pathological* sparsity patterns that fail to have optimal solutions on some instances (cf Theorem 7.3.3 and Theorem 7.4.1) and thus possibly cause instabilities in optimization algorithms (see Example 7.3.5 and Figure 7.1). On the other hand, we also prove several positive results on the BAP and closedness, notably for sparse one-hidden-layer ReLU neural networks (cf. Theorem 7.3.7 and Theorem 7.4.2). These results provide new instances of network architectures where the BAP is proved (cf Theorem 7.3.7) and substantially generalize existing ones (cf. Theorem 7.4.2).

In the future, a particular theoretical challenge is to propose necessary and sufficient conditions for the BAP and closedness of $\mathcal{F}_{\mathbf{I}}(\Omega)$, if possible covering in a single framework both types of domains Ω considered here. The fact that the conditions established on these two types of domains are very similar (cf. the similarity between Theorem 7.3.3 and Theorem 7.4.1, as well as between Theorem 7.3.7 and Corollary 7.4.4) is encouraging. Similar to the remark in Chapter 3, an interesting algorithmic challenge is to substantially reduce the complexity of the algorithm to decide the closedness of $\mathcal{L}_{\mathbf{I}}$ in Lemma 3.2.16, which is currently doubly exponential. It calls for a more efficient algorithm to make this check more practical. Achieving a practically tractable algorithm would for instance allow to check if a support selected e.g. by IMP is pathological or not. This would certainly consolidate the algorithmic robustness and theoretical foundations of pruning techniques to sparsity deep neural networks. From a more theoretical perspective, the existence of an optimum solution in the context of classical linear inverse problems has been widely used to analyze the desirable properties of certain cost functions, e.g. ℓ^1 minimization for sparse recovery. Knowing that an optimal solution exists for a given sparse neural network training problem is thus likely to open the door to further fruitful insights.

Conclusion

This thesis proposed an approach for a theoretical study of sparse deep neural networks: First, we studied a simpler problem named "Fixed support matrix factorization" (cf. (2.4.FSMF)), which is also a particular case of the sparse matrix factorization problem (cf. Section 2.3). The second step was to apply the results of (2.4.FSMF) to address certain questions with regards to sparse deep neural networks. This approach was motivated and justified in Chapter 2, in which we exhibited the similarity between sparse deep neural networks and sparse matrix factorization as well as the importance of the study of (2.4.FSMF) to have a better understanding of the problem of sparse matrix factorization.

A summary of main contributions

Fixed support matrix factorization We devoted three consecutive chapters (Chapters 3 to 5) to present our study of (2.4.FSMF).

In Chapter 3 - **Ill-posedness/Non-closedness** and **NP-hardness**, we introduced the first challenge when it comes to (2.4.FSMF): for certain inputs $(\mathbf{A}, I, J)$, the corresponding instance of (2.4.FSMF) does not have any optimal solution. After that, we proceeded further with a related question: under which conditions of (I, J) does the corresponding instances of (2.4.FSMF) always admit an optimal solution, regardless of target matrix $\mathbf{A}$? Using a simple topological argument, that question is equivalent to: for which (I, J) is the set $\mathcal{E}_{I,J} := \{\mathbf{X}\mathbf{Y}^\top \mid \text{supp}(\mathbf{X}) \subseteq I, \text{supp}(\mathbf{Y}) \subseteq J\} \subseteq \mathbb{R}^{m\times n}$ closed? Using the quantifier elimination algorithm [BPR06, Chapter 14], we proposed an algorithm whose complexity is doubly exponential to resolve this problem. Another challenging aspect of (2.4.FSMF) that is also described in Chapter 3 is the NP-hardness (cf. Theorem 3.3.4). This result implies that even if one has the knowledge of the factor supports, the problem of sparse matrix factorization remains challenging.

Chapter 4 - **Landscape**, is dedicated to study the landscape of the objective function of (2.4.FSMF). Two important notions are presented in Chapter 4: spurious local minima and spurious local valleys. These objects are conceptually "bad" for first-order optimization methods in the sense that functions possessing these spurious objects are difficult to globally optimize using first-order optimization methods. Then, we showed that in general, the landscape of (2.4.FSMF) possesses these undesirable objects. This might intuitively justify why (2.4.FSMF) is NP-hard to solve, as proved in Chapter 3.

While the message of Chapters 3 and 4 is somewhat negative, Chapter 5 - **Tractability**, in contrast, presents several families of support constraints (I, J) of (2.4.FSMF), whose corresponding instances are both tractable and of practical interest. These families are introduced in Theorems 5.2.3 and 5.2.9. Many popular support constraints of

(2.4.FSMF) such as Low Rank Matrix Approximation (LRMA) and Hierarchical Off-diagonal Low-rank (HODLR) matrices (cf. Section 2.4.1) belong to these families. Moreover, different from the general landscape of (2.4.FSMF) studied Chapter 4, those of instances with tractable support constraints introduced in this chapter are guaranteed to be free of spurious objects.

The two remaining chapters discuss applications of the results of (2.4.FSMF) to address certain questions in the world of sparse deep neural networks.

Butterfly parameterization The focus of Chapter 6 is the butterfly-parameterization, a successful approach in scientific computing and sparse deep neural networks consisting of replacing dense linear operators by products of extremely sparse and structured ones. Using the results of Chapter 5, we show that the usage of butterfly-parameterization is equivalent to a modification of the hypothesis classes of the linear operators in DNNs: instead of optimizing on plain $\mathbb{R}^{m\times n}$ (where m, n is the size of a weight matrix), one learns a matrix over a set of deformable butterfly matrices (cf. Definition 6.3.6). This set composes of matrices whose certain submatrices are of low rank (cf Corollary 6.5.12). In addition, we also show that there exists an approximate algorithm for the problem of projection onto a set of deformable butterfly matrices.

Existence of optimal solutions in sparse ReLU neural networks training The last technical chapter, Chapter 7, is devoted to answer the following question: under which conditions of the support constraints $\mathbf{I} = (I_L, \dots I_1)$ that the corresponding sparse deep neural networks training problems always admits an optimum, regardless of finite training data set? Using the results of Chapter 3, on the one hand, we show the following necessary condition: the existence of optimal solutions is only possible if $\mathcal{E}_{\mathbf{I}}$ is closed (cf. Theorem 7.3.3). On the other hand, we also provide a sufficient condition for the existence of optima: all $\mathbf{I}$ of one-hidden-layer NNs with scalar output dimension guarantee such an existence (cf. Theorem 7.3.7). A similar study is also conducted for the case of function space of sparse deep neural networks (cf. Theorems 7.4.1 and 7.4.2).

Open questions and perspectives

The following questions are mostly selected from the conclusions of all chapters. Those are questions which give an overview of this thesis and suggest potentially interesting research questions for certain communities (such that those working on matrix factorization and/or sparse DNNs) in general, and my future self in particular.

With regards to sparse DNNs training There are certain important questions that remain open:

1. In this thesis, we put an emphasis on the training/optimizing with fixed-support constraints on the weight matrices/linear operators. Our approach is complementary to a line of research (notably [MYSsS20, OHR20, PRN+20]) that considers only the pruning operation. In practice, proposed algorithms are often composed of both operations (pruning and retraining with fixed support constraints) and various types of regularizations. A unified theoretical framework that takes care of all the elements

is very much desired. Hopefully, the tools that I have developed during three-year Ph.D. can show us a way to that goal.

2. In Chapter 4-Chapter 5, we discussed simultaneously the tractability and landscapes of Equation (2.4.FSMF). A natural question is to extend these analysis to the setting of sparse fixed support NNs, especially the analysis of the landscape because of its impact to the performance of first-order optimization methods, the bread and butter of nowadays (sparse) deep learning. However, such a question is quite challenging, even from the modelling phase: finding a meaningful/practical setting in which the question becomes mathematically analyzable is non-trivial.

3. There is a "small" problem that we ignored in Chapter 6 - **butterfly parameterization**: In case training/inference can use GPUs, the speed-up expected for butterfly parameterization is nearly non-existent. The reason is simple: linear operators, even in large deep learning models are designed to get the most benefits out of GPUs parallelization. Therefore, even with structured sparse operators as in the case of butterfly parameterization, a careful implementation of sparse DNNs is required to get a faster training/inference than their dense counterparts. With several members in the laboratory, I work on such a project. The goal of the project is: for a given size of a linear operator, find a sequence of DB parameters Θ that: 1) is faster (than dense matrices) to compute; 2) can be used as a linear operator in deep learning architectures (such as Vision Transformers - ViTs [DBK+21]) with "good" performances.

4. In Chapter 7, we (partly) responded to an important question with regards to sparse DNNs training: whether its corresponding optimization problems have optimal solutions. However, all of our "positive" results (i.e., the existence of optimal solutions is guaranteed) are restrictive to one-hidden-layer neural networks with scalar output. This result is sharp, in the sense that if the output dimension is at least two, such existence is *not* guaranteed [LMQ21]. The result in [LMQ21] pretty much convinced me to believe that for most of deeper ReLU NN architectures which have more than one hidden layer and with/without sparsity patterns, optimal solutions do not exist for carefully crafted training data. This conjecture includes also dense ReLU NNs with more than one hidden layer as well. We have not been able to prove this conjecture with our current tool yet (cf. Theorem 7.3.3).

With regards to sparse matrix factorization

1. Is there a fast (i.e. polynomial) algorithm deciding the closedness of the set $\mathcal{E}_{I,J}$ (cf. (3.1))? We do not have any intuition for this question, at the moment. However, an anonymous reviewer of [LRG23] suggested that the problem of deciding the closedness of $\mathcal{E}_{I,J}$ might be ER-complete due to certain similarities between our problem and other proved ER-complete problems in the literature (for example, [AKM21] and [BHJ+22]). We have not verified this ER-completeness conjecture yet. Nevertheless, I wish it is not true because a polynomial algorithm for this problem might open up many possibilities to enlarge the set of polynomially tractable instances of (2.4.FSMF) in Chapter 5.

2. In this thesis, we show that projecting a target matrix to the set of matrices admitting a representation as products of sparse factors can be polynomially possible (up to an approximation ratio in the sense of (6.12)): for the set of HODLR matrices (cf. Lemma 5.2.4) and for the set of deformable butterfly matrices with chainable Θ (cf. Theorems 6.5.9 and 6.5.14). Another popular matrix sets belongs to this category is the hierarchical semi-separable matrices (HSS) [Mar11]. I think that we can reuse the normalization technique in Section 6.5 to prove an error bound similar to (6.12) for the set of HSS matrices.

3. Can we improve the constant C_N in Theorems 6.5.9 and 6.5.14 for a projection algorithm (possibly different from Algorithm 8) of the set of deformable butterfly matrices (cf. Definition 6.3.6)? Intuitively, I think our bound is not sharp enough since the empirical performance in Section 6.6 indicates a much smaller C_N. Therefore, the analysis of Algorithm 8 in the proof is likely to be enhanced. Another direction, which might be more fun to consider, is to rethink about the hierarchical factorization algorithm. We can, in fact, factorize a matrix into *three* intermediate factors, instead of *two* factors, in each iteration in polynomial time if the DB parameters $\theta_\ell, \ell = 1, \ldots, N$ all have small values of (b_ℓ, c_ℓ). If a relation as in (6.34) can be obtained, we likely to improve C_N significantly because there are only $(N-1)/2$ factorization to perform. The approach can be extended to any q-factor factorization.

Appendix A

List of activation functions

Acronym	Full name	Given by
ReLU	Rectified Linear Unit	$\max(x, 0)$
pReLU	parametric Rectified Linear Unit	$\max(ax, x)$ for some $a \geq 0$
heavyside	heavyside	$\mathbf{1}_{x\geq 0}$
sigmoid	sigmoid	$\frac{1}{1+\exp(-x)}$
tanh	tanh	$\frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$
arctan	arctan	$\arctan(x)$
ISRLU	Inverse Square Root Linear Unit	$x\mathbf{1}_{x\geq 0} + \frac{x}{\sqrt{1+ax^2}}\mathbf{1}_{x<0}$ for some $a > 0$
ISRU	Inverse Square Root Unit	$\frac{x}{\sqrt{1+ax^2}}$ for some $a > 0$
ELU	Exponential Linear Unit	$x\mathbf{1}_{x\geq 0} + (\exp(x) - 1)\mathbf{1}_{x<0}$
softsign	softplus	$\ln(1 + \exp(x))$

Table A.1: List of commonly used activation functions and their definitions.

Appendix B

Deferred proofs

B.1 Proof for results in Chapter 3

Proof for Proposition 3.1.6

Proof. We prove the two implications one by one:

1) $\mathcal{E}_{I,J}$ is closed $\implies$ (I, J) is well-posed: Note that one can rewrite (2.4.FSMF) equivalently as:

$$d_{I,J}(\mathbf{A}) = \inf_{\mathbf{B}\in\mathcal{E}_{I,J}} \|\mathbf{A}-\mathbf{B}\|^2$$

In words, (2.4.FSMF) amounts to finding the projection of a matrix $\mathbf{A}$ onto $\mathcal{E}_{I,J}$. Since $\mathbf{0}_{m\times n} \in \mathcal{E}_{I,J}$, the set $\mathcal{E}_{I,J}$ is not empty. Therefore, $0 \leq d_{I,J}(\mathbf{A}) \leq \|\mathbf{A}\|^2$. Thus, one can define $d_{I,J}(\mathbf{A})$ equivalently as:

$$d_{I,J}(\mathbf{A}) = \inf \|\mathbf{A}-\mathbf{B}\|^2 \quad \text{subject to: } \mathbf{B} \in \mathcal{E}_{I,J} \cap \mathcal{B}(\mathbf{A}, \|\mathbf{A}\|)$$

where $\mathcal{B}(\mathbf{A}, \|\mathbf{A}\|)$ is the ball centered at $\mathbf{A}$ with radius $\|\mathbf{A}\|$. Note that $\mathcal{E}_{I,J} \cap \mathcal{B}(\mathbf{A}, \|\mathbf{A}\|)$ is closed (since it is an intersection of two closed sets) and bounded (since $\mathcal{B}(\mathbf{A}, \|\mathbf{A}\|)$ is bounded). Therefore, $\mathcal{E}_{I,J} \cap \mathcal{B}(\mathbf{A}, \|\mathbf{A}\|)$ is compact. Since the function $f(\cdot) = \|\mathbf{A}-\cdot\|^2$ is continuous, the min is attained. That implies there exists $\mathbf{C} \in \mathcal{E}_{I,J}$ such that $\|\mathbf{A}-\mathbf{C}\|^2 = d_{I,J}(\mathbf{A})$. Thus, (I, J) is well-posed.

2) $\mathcal{E}_{I,J}$ is not closed $\implies$ (I, J) is ill-posed: If $\mathcal{E}_{I,J}$ is not closed, take $\mathbf{A} \in \overline{\mathcal{E}_{I,J}} \setminus \mathcal{E}_{I,J}$. The optimal value of (2.4.FSMF) is trivially zero. Moreover, it will never be attained since we choose $\mathbf{A} \notin \mathcal{E}_{I,J}$. Therefore, (I, J) is ill-posed.

□

B.2 Proof for results in Chapter 5

B.2.1 An example where the best approximation property and closedness are different

Example B.2.1. Consider $C^0([-1, 1])$ the set of continuous functions on the interval $[-1, 1]$, equipped with the norm sup $\|f\|_\infty = \max_{x\in[-1,1]} |f(x)|$, and define S, the subset

of all functions $f \in C^0([-1,1])$ such that:

$$\int_0^1 f\,dx - \int_{-1}^0 f\,dx = 1$$

It is easy to verify that S is closed. We show that the constant function $f = 0$ does not have a projection in S (i.e., a function $g \in S$ such that $\|f-g\|_\infty = \inf_{h\in S}\|f-h\|_\infty$).

First we observe that since $f = 0$, we have $\|f-h\|_\infty = \|h\|_\infty$ for each $h \in S$, and we show that $\inf_{h\in S}\|f-h\|_\infty \geq 1/2$. Indeed, for $h \in S$ we have:

$$1 = \int_0^1 h\,dx - \int_{-1}^0 h\,dx \leq \left|\int_0^1 h\,dx\right| + \left|\int_{-1}^0 h\,dx\right| \leq 2\|h\|_\infty = 2\|f-h\|_\infty. \tag{B.1}$$

Secondly, we show a sequence of $\{h_n\}_{k\in\mathbb{N}}$ such that $h_n \in S$ and $\lim_{n\to\infty}\|h_n\|_\infty = 1/2$. Consider the odd function h_n (i.e. $h_n(x) = -h_n(-x)$) such that:

$$h_n(x) = \begin{cases} c_n, & x \in [1/n, 1] \\ nc_n x & x \in [0, 1/n) \end{cases}$$

where $c_n = n/(2n-1)$. It is evident that $h_n \in S$ because:

$$\begin{aligned}\int_0^1 h_n\,dx - \int_{-1}^0 h_n\,dx = 2\int_0^1 h_n\,dx &= 2\left(\int_0^{1/n} h_n\,dx + \int_{1/n}^1 h_n\,dx\right) \\ &= 2\left(\frac{c_n}{2n} + \frac{c_n(n-1)}{n}\right) = \frac{c_n(2n-1)}{n} = 1\end{aligned}$$

Moreover, we also have $\lim_{n\to\infty}\|h_n\|_\infty = \lim_{n\to\infty} c_n = 1/2$.

Finally, we show that $1/2$ cannot be attained. By contradiction, assume that there exists $g \in S$ such that $\|f-g\|_\infty = 1/2$, i.e., as we have seen, $\|g\|_\infty = 1/2$. Using Equation (B.1), the equality will only hold if $g(x) = 1/2$ in $[0,1]$ and $g(x) = -1/2$ in $[-1,0]$. However, g is not continuous, a contradiction.

B.2.2 Statement and proof of Lemma B.2.2

Lemma B.2.2 (Convergence of affine function)**.** *Let Ω be a non-empty interior subset $\mathbb{R}^n$. If the sequence $\{f^k\}_{k\in\mathbb{N}}$, $f^k : \mathbb{R}^n \mapsto \mathbb{R}^m : x \mapsto \mathbf{A}^k x + \mathbf{b}^k$ where $\mathbf{A}^k \in \mathbb{R}^{m\times n}, \mathbf{b}^k \in \mathbb{R}^m$ converges pointwise to a function f on Ω, then f is affine (i.e., $f = \mathbf{A}x + \mathbf{b}$ for some $\mathbf{A} \in \mathbb{R}^{m\times n}, \mathbf{b} \in \mathbb{R}^m$). Moreover, $\lim_{k\to\infty}\mathbf{A}^k = \mathbf{A}$ and $\lim_{k\to\infty}\mathbf{b}_k = \mathbf{b}$.*

Proof. Consider $x_0 \in \Omega'$, an open subset of Ω (Ω' exists since Ω is a non-empty interior subset of $\mathbb{R}^n$). Define $g^k(x) = f^k(x) - f^k(x_0)$ and $g(x) = f(x) - g(x_0)$. The function g^k is linear and g^k converges pointwise to g on Ω (and thus, on Ω'). We first prove that g is linear. Indeed, for any $x, y \in \Omega, \alpha, \beta \in \mathbb{R}$ such that $\alpha x + \beta y \in \Omega$, we have:

$$\begin{aligned} g(\alpha x + \beta y) &= \lim_{k\to\infty} g^k(\alpha x + \beta y) \\ &= \lim_{k\to\infty} \alpha g^k(x) + \beta g^k(y) \\ &= \alpha \lim_{k\to\infty} g^k(x) + \beta \lim_{k\to\infty} g^k(y) \\ &= \alpha g(x) + \beta g(y) \end{aligned}$$

Therefore, there must exist $\mathbf{A} \in \mathbb{R}^{m\times n}$ such that $g(x) = \mathbf{A}x$. Choosing $\mathbf{b} := g(x_0)$, we have $f(x) = g(x) + g(x_0) = \mathbf{A}x + \mathbf{b}$.

Moreover, since Ω' is open, there exists a positive r such that the ball $\mathcal{B}(x, r) \subseteq \Omega'$. Choosing $x_i = x_0 + (r/2)\mathbf{e}_i$ with $\mathbf{e}_i$ the ith canonical vector, we have:

$$\lim_{k\to\infty} g^k(x_i) = \lim_{k\to\infty} (r/2)\mathbf{A}^k\mathbf{e}_i = (r/2)\mathbf{A}e_i,$$

or, equivalently, the ith column of $\mathbf{A}$ is the limit of the sequence generated by the ith column of $\mathbf{A}^k$. Repeating this argument for all $1 \leq i \leq n$, we have $\lim_{k\to\infty} \mathbf{A}^k = \mathbf{A}$. This also implies $\lim_{k\to\infty} \mathbf{b}^k = \mathbf{b}$ immediately. □

Bibliography

[AANR17] Alireza Aghasi, Afshin Abdi, Nam Nguyen, and Justin Romberg. Net-trim: Convex pruning of deep neural networks with performance guarantee. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 3180–3189, Red Hook, NY, USA, 2017. Curran Associates Inc. 34

[ABMM18] Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. Understanding deep neural networks with rectified linear units. In *International Conference on Learning Representations*, 2018. URL: https://openreview.net/forum?id=B1J_rgWRW. 174, 175, 177

[AKM21] Mikkel Abrahamsen, Linda Kleist, and Tillmann Miltzow. Training neural networks is er-complete. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 18293–18306. Curran Associates, Inc., 2021. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/9813b270ed0288e7c0388f0fd4ec68f5-Paper.pdf. 70, 196

[ARS18] Alessandro Achille, Matteo Rovere, and Stefano Soatto. Critical learning periods in deep networks. In *International Conference on Learning Representations*, 2018. 47

[AZLS19] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 242–252. PMLR, 09–15 Jun 2019. URL: https://proceedings.mlr.press/v97/allen-zhu19a.html. 15

[BD08] Thomas Blumensath and Mike Davies. Iterative thresholding for sparse approximations. *Journal of Fourier Analysis and Applications*, 14:629–654, 12 2008. doi:10.1007/s00041-008-9035-z. 32

[BHJ+22] Daniel Bertschinger, Christoph Hertrich, Paul Jungeblut, Tillmann Miltzow, and Simon Weber. Training fully connected neural networks is $\exists\mathbb{R}$-complete, 2022. arXiv:2204.01368. 70, 196

[BHMM19] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-

off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019. URL: https://www.pnas.org/doi/abs/10.1073/pnas.1903070116, arXiv:https://www.pnas.org/doi/pdf/10.1073/pnas.1903070116, doi:10.1073/pnas.1903070116. 15

[BK16] J. Ballani and D. Kressner. *Matrices with Hierarchical Low-Rank Structures*, volume 2173. 01 2016. doi:10.1007/978-3-319-49887-4_3. 20, 48

[BM03] Samuel Burer and Renato Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming, Series B*, 95:329–357, 02 2003. doi:10.1007/s10107-002-0352-8. 42, 47, 59

[BNS16] S. Bhojanapalli, B. Neyshabur, and N. Srebro. Global optimality of local search for low rank matrix recovery. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL: https://proceedings.neurips.cc/paper/2016/file/b139e104214a08ae3f2ebcce149cdf6e-Paper.pdf. 74, 75

[BPR06] Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in Real Algebraic Geometry (Algorithms and Computation in Mathematics)*. Springer-Verlag, Berlin, Heidelberg, 2006. 52, 57, 58, 59, 60, 68, 194

[BR92] Avrim L. Blum and Ronald L. Rivest. Training a 3-node neural network is np-complete. *Neural Networks*, 5(1):117–127, 1992. URL: https://www.sciencedirect.com/science/article/pii/S0893608005800103, doi:https://doi.org/10.1016/S0893-6080(05)80010-3. 70

[BST14] J. Bolte, S. Sabach, and M. Teboulle. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Mathematical Programming*, 146(1-2):459–494, 2014. URL: https://hal.inria.fr/hal-00916090, doi:10.1007/s10107-013-0701-9. 37, 46, 47

[BT09] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresding algorithm for linear inverse problems. *SIAM J. Imaging Sciences*, 2:183–202, 01 2009. doi:10.1137/080716542. 16

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, USA, 2004. 72

[CB18] Lénaïc Chizat and Francis Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 3040–3050, Red Hook, NY, USA, 2018. Curran Associates Inc. 18

[CDL+22] B. Chen, T. Dao, K. Liang, J. Yang, Z. Song, A. Rudra, and C. Ré. Pixelated butterfly: Simple and efficient sparse training for neural network models. In

International Conference on Learning Representations, 2022. URL: https://openreview.net/forum?id=Nfl-iXa-y7R. 48, 163

[CDY09] E. Candès, L. Demanet, and L. Ying. A fast butterfly algorithm for the computation of fourier integral operators. *Multiscale Modeling & Simulation*, 7(4):1727–1750, 2009. arXiv:https://doi.org/10.1137/080734339, doi:10.1137/080734339. 48

[CK14] Maxwell D. Collins and Pushmeet Kohli. Memory bounded deep convolutional networks. *ArXiv*, abs/1412.1442, 2014. URL: https://api.semanticscholar.org/CorpusID:16153118. 34

[CL19] J. Chen and X. Li. Model-free nonconvex matrix completion: Local minima analysis and applications in memory-efficient kernel PCA. *Journal of Machine Learning Research*, 20(142):1–39, 2019. URL: http://jmlr.org/papers/v20/17-776.html. 74, 75

[CLC19] Y. Chi, Y. M. Lu, and Y. Chen. Nonconvex optimization meets low-rank matrix factorization: An overview. *Trans. Sig. Proc.*, 67(20):5239–5269, oct 2019. doi:10.1109/TSP.2019.2937282. 20, 75

[CLMW11] Emmanuel J. Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *J. ACM*, 58(3), jun 2011. doi:10.1145/1970392.1970395. 43

[CMDP19] Louizos Christos, Welling Max, and Kingma Diederik P. Learning sparse neural networks through l0 regularization. In *ICLR*, 2019. 34

[CSGR96] Tautvydas Cibas, Françoise Fogelman Soulié, Patrick Gallinari, and Sarunas Raudys. Variable selection with neural networks. *Neurocomputing*, 12(2):223–248, 1996. Current European Neurocomputing Research. URL: https://www.sciencedirect.com/science/article/pii/0925231295001212, doi:https://doi.org/10.1016/0925-2312(95)00121-2. 33

[CWXC20] Shih-Kang Chao, Zhanyu Wang, Yue Xing, and Guang Cheng. Directional pruning of deep neural networks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. 34

[DBK+21] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL: https://openreview.net/forum?id=YicbFdNTTy. 196

[DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding.

ArXiv, abs/1810.04805, 2019. URL: https://api.semanticscholar.org/CorpusID:52967399. 12, 14

[DCS+22a] Tri Dao, Beidi Chen, Nimit Sohoni, Arjun Desai, Michael Poli, Jessica Grogan, Alexander Liu, Aniruddh Rao, Atri Rudra, and Christopher Ré. Monarch: Expressive structured matrices for efficient and accurate training. In *39th International Conference on Machine Learning*, 2022. 163

[DCS+22b] Tri Dao, Beidi Chen, Nimit Sharad Sohoni, Arjun D. Desai, Michael Poli, Jessica Grogan, Alexander Liu, Aniruddh Rao, Atri Rudra, and Christopher Ré. Monarch: Expressive structured matrices for efficient and accurate training. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 4690–4721. PMLR, 2022. URL: https://proceedings.mlr.press/v162/dao22a.html. 20, 26, 27, 29, 93, 112, 115, 116, 119, 120, 123, 138

[DDS+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 11, 17

[DDZ+19a] Xiaohan Ding, Guiguang Ding, Xiangxin Zhou, Yuchen Guo, Jungong Han, and Ji Liu. *Global Sparse Momentum SGD for Pruning Very Deep Neural Networks*. Curran Associates Inc., Red Hook, NY, USA, 2019. 32

[DDZ+19b] Xiaohan Ding, Guiguang Ding, Xiangxin Zhou, Yuchen Guo, Jungong Han, and Ji Liu. Global sparse momentum SGD for pruning very deep neural networks. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 6379–6391, 2019. URL: http://papers.nips.cc/paper/8867-global-sparse-momentum-sgd-for-pruning-very-deep-neural-networks. 47

[DGE+19] Tri Dao, Albert Gu, Matthew Eichhorn, Atri Rudra, and Christopher Ré. Learning fast algorithms for linear transforms using butterfly factorizations. *CoRR*, abs/1903.05895, 2019. URL: http://arxiv.org/abs/1903.05895, arXiv:1903.05895. 20, 26, 27, 29, 39, 40, 48, 112, 114, 116, 119, 120, 123, 138, 163

[DJK23] Steffen Dereich, Arnulf Jentzen, and Sebastian Kassing. On the existence of minimizers in shallow residual relu neural network optimization landscapes, 2023. arXiv:2302.14690. 161, 163

[DK23] Steffen Dereich and Sebastian Kassing. On the existence of optimal shallow feedforward networks with relu activation, 2023. arXiv:2303.03950. 161, 163

[DM09] Wei Dai and Olgica Milenkovic. Subspace pursuit for compressive sensing signal reconstruction. *IEEE Transactions on Information Theory*, 55(5):2230–2249, 2009. doi:10.1109/TIT.2009.2016006. 45

[Don06] D.L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006. doi:10.1109/TIT.2006.871582. 32

[Dor23] Thomas A Dorfer. Ten years of ai in review, May, 2023. URL: https://towardsdatascience.com/ten-years-of-ai-in-review-85decdb2a540. 10

[DSG+20] T. Dao, N. Sohoni, A. Gu, M. Eichhorn, A. Blonder, M. Leszczynski, A. Rudra, and C. Ré. Kaleidoscope: An efficient, learnable representation for all structured linear maps. In *International Conference on Learning Representations*, 2020. URL: https://openreview.net/forum?id=BkgrBgSYDS. 20, 26, 27, 29, 48, 112, 114, 115, 119, 120, 123, 163

[DZDW18] Happiness Ugochi Dike, Yimin Zhou, Kranthi Kumar Deveerasetty, and Qingtian Wu. Unsupervised learning based on artificial neural network: A review. In *2018 IEEE International Conference on Cyborg and Bionic Systems (CBS)*, pages 322–327, 2018. doi:10.1109/CBS.2018.8612259. 12

[EC96] A.P. Engelbrecht and I. Cloete. A sensitivity analysis algorithm for pruning feedforward neural networks. In *Proceedings of International Conference on Neural Networks (ICNN'96)*, volume 2, pages 1274–1278 vol.2, 1996. doi:10.1109/ICNN.1996.549081. 32

[EGM+20] Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020. 18

[EKT21] Bryn Elesedy, Varun Kanade, and Yee Whye Teh. Lottery tickets in linear models: An analysis of iterative magnitude pruning, 2021. URL: https://arxiv.org/abs/2007.08243, doi:10.48550/ARXIV.2007.08243. 35

[EMRV92] N. Engheta, W.D. Murphy, V. Rokhlin, and M.S. Vassiliou. The fast multipole method (fmm) for electromagnetic scattering problems. *IEEE Transactions on Antennas and Propagation*, 40(6):634–641, 1992. doi:10.1109/8.144597. 113

[EY36] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936. doi:10.1007/BF02288367. 20, 42, 47, 54

[FC19] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representation*, 2019. 17, 26, 31, 32, 34, 47, 163, 170

[Fou11] Simon Foucart. Hard thresholding pursuit: An algorithm for compressive sensing. *SIAM J. Numerical Analysis*, 49:2543–2563, 01 2011. doi:10.2307/41582705. 32

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org. 18, 25, 29, 31, 93, 138

[GBGR23] Antoine Gonon, Nicolas Brisebarre, Rémi Gribonval, and Elisa Riccietti. Approximation speed of quantized vs. unquantized ReLU neural networks and beyond. *IEEE Transactions on Information Theory*, 69(6):3960–3977, June 2023. URL: https://hal.science/hal-03672166, doi:10.1109/TIT.2023.3240360. 12

[GEH19] Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks, 2019. arXiv:1902.09574. 31, 32

[GEN+17] A. Gordon, Elad Eban, Ofir Nachum, Bo Chen, Tien-Ju Yang, and E. Choi. Morphnet: Fast & simple resource-constrained structure learning of deep networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1586–1595, 2017. URL: https://api.semanticscholar.org/CorpusID:206596875. 34

[GG10] Nicolas Gillis and François Glineur. Low-rank matrix approximation with weights or missing data is np-hard. *SIAM Journal on Matrix Analysis and Applications*, 32, 12 2010. doi:10.1137/110820361. 20, 42, 63

[GJZ17] R. Ge, C. Jin, and Y. Zheng. No spurious local minima in nonconvex low rank problems: A unified geometric analysis. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1233–1242. JMLR.org, 2017. 74, 75

[GLM16] R. Ge, J. D. Lee, and T. Ma. Matrix completion has no spurious local minimum. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL: https://proceedings.neurips.cc/paper/2016/file/7fb8ceb3bd59c7956b1df66729296a4c-Paper.pdf. 74, 75

[GMR23] Rémi Gribonval, Theo Mary, and Elisa Riccietti. Scaling is all you need: quantization of butterfly matrix products via optimal rank-one quantization. working paper or preprint, June 2023. URL: https://inria.hal.science/hal-04146706. 12

[GP02] Federico Girosi and Tomaso Poggio. Networks and the best approximation property. *Biological Cybernetics*, 63, 08 2002. doi:10.1007/BF00195855. 162, 163, 171

[GV18] Nicolas Gillis and Stephen A. Vavasis. On the complexity of robust pca an l1-norm low-rank matrix approximation. *Math. Oper. Res.*, 43(4):1072–1084, nov 2018. 69

[GVL96] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, third edition, 1996. 41, 47, 48, 55

[GZL+23] Antoine Gonon, Léon Zheng, Clément Lalanne, Quoc-Tung Le, Guillaume Lauga, and Can Pouliquen. Sparsity in neural networks can improve their privacy. In *GRETSI 2023 - XXIXème Colloque Francophone de Traitement du Signal et des Images*, Grenoble, France, August 2023. URL: https://hal.science/hal-04062317. 17

[HABN+21] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. 22(1), 2021. 17, 33, 35, 47

[Hac99] Wolfgang Hackbusch. A sparse matrix arithmetic based on h-matrices. part i: Introduction to h-matrices. *Computing*, 62:89–108, 1999. 20, 48

[Had02] J. Hadamard. Sur les problèmes aux dérivés partielles et leur signification physique. *Princeton University Bulletin*, 13:49–52, 1902. 53

[Hag93] M. Hagiwara. Removal of hidden units and weights for back propagation networks. In *Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan)*, volume 1, pages 351–354 vol.1, 1993. doi:10.1109/IJCNN.1993.713929. 32

[Has90] Johan Hastad. Tensor rank is np-complete. *Journal of Algorithms*, 11(4):644–654, 1990. URL: https://www.sciencedirect.com/science/article/pii/0196677490900146, doi:https://doi.org/10.1016/0196-6774(90)90014-6. 43, 69

[HK00] Wolfgang Hackbusch and Boris N. Khoromskij. A sparse h-matrix arithmetic. part ii: Application to multi-dimensional problems. *Computing*, 64:21–47, 2000. 20, 48

[HMD16] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL: http://arxiv.org/abs/1510.00149. 12, 26, 31, 32, 33, 170

[HPTD15] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks, 2015. arXiv:1506.02626. 26, 31, 32, 33, 47, 163, 170

[HS93] Babak Hassibi and David Stork. Second order derivatives for network pruning: Optimal brain surgeon. In S. Hanson, J. Cowan, and C. Giles, editors,

Advances in Neural Information Processing Systems, volume 5. Morgan-Kaufmann, 1993. URL: https://proceedings.neurips.cc/paper/1992/file/303ed4c69846ab36c2904d3ba8573050-Paper.pdf. 33

[HSS+22] H. Hu, Z. Salcic, L. Sun, G. Dobbie, P. S. Yu, and X. Zhang. Membership inference attacks on machine learning: A survey. *ACM Computing Surveys*, 2022. 15

[HTF] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. New York, NY, USA. 25, 40

[HVD15] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531, 2015. URL: https://api.semanticscholar.org/CorpusID:7200347. 12

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. pages 770–778, 06 2016. doi:10.1109/CVPR.2016.90. 16, 17, 18, 30, 34

[JAS19] T. Jared, T. Andrew, and V. Simon. Matrix rigidity and the ill-posedness of robust pca and matrix completion. *SIAM Journal on Mathematics of Data Science*, 1(3):537–554, 2019. doi:10.1137/18M1227846. 54, 65, 67

[JC16] Ian Jolliffe and Jorge Cadima. Principal component analysis: A review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374:20150202, 04 2016. doi:10.1098/rsta.2015.0202. 42

[JHLL17] Jianxin Wu Jian-Hao Luo and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *ICCV*, pages 5058–5066, 2017. 32

[Kaw16] Kenji Kawaguchi. Deep learning without poor local minima. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 586–594. Curran Associates, Inc., 2016. URL: http://papers.nips.cc/paper/6112-deep-learning-without-poor-local-minima.pdf. 18, 36, 74, 75

[KBV09] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009. doi:10.1109/MC.2009.263. 42

[KKS16] N. Kishore Kumar and J. Shneider. Literature survey on low rank approximation of matrices. *ArXiv preprint 1606.06511*, 2016. 83

[KKV00] P. Kainen, V. Kůrková, and A. Vogt. Best approximation by heaviside perceptron networks. *Neural Networks*, 13(7):695–697, 2000. 161, 163, 171

[KRS+20] Aditya Kusupati, Vivek Ramanujan, Raghav Somani, Mitchell Wortsman, Prateek Jain, Sham Kakade, and Ali Farhadi. Soft threshold weight reparameterization for learnable sparsity. In *Proceedings of the International Conference on Machine Learning*, July 2020. 18

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf. 16, 22, 25, 26

[Kůr95] Věra Kůrková. Approximation of functions by perceptron networks with bounded number of hidden units. *Neural Networks*, 8(5):745–750, 1995. 161, 163, 171

[LAT19] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. In *ICLR*, 2019. 26, 30, 32, 163

[LB98] Yann LeCun and Yoshua Bengio. *Convolutional Networks for Images, Speech, and Time Series*, page 255–258. MIT Press, Cambridge, MA, USA, 1998. 14

[LBBH98] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi:10.1109/5.726791. 30, 34

[LCC+21] Shiwei Liu, Tianlong Chen, Xiaohan Chen, Zahra Atashgahi, Lu Yin, Huanyu Kou, Li Shen, Mykola Pechenizkiy, Zhangyang Wang, and Decebal Constantin Mocanu. Sparse training via boosting pruning plasticity with neuroregeneration. *Advances in Neural Information Processing Systems*, 2021. 18

[LCC+22] Shiwei Liu, Tianlong Chen, Xiaohan Chen, Li Shen, Decebal Constantin Mocanu, Zhangyang Wang, and Mykola Pechenizkiy. The unreasonable effectiveness of random pruning: Return of the most naive baseline for sparse training. *arXiv preprint arXiv:2202.02643*, 2022. 168

[LDS90] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1990. URL: https://proceedings.neurips.cc/paper/1989/file/6c9882bbac1c7093bd25041881277658-Paper.pdf. 33

[Le20] Quoc-Tung Le. Multilayer Sparse Matrix Factorization. Master's thesis, ENS Paris Saclay, September 2020. URL: https://inria.hal.science/hal-03130680. 19, 22

[LG15] L. Le Magoarou and R. Gribonval. Chasing butterflies: In search of efficient dictionaries. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3287–3291, 2015. doi:10.1109/ICASSP.2015.7178579. 40, 48

[LG21] Quoc-Tung Le and Rémi Gribonval. Structured Support Exploration For Multilayer Sparse Matrix Factorization. In *ICASSP 2021 - IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1–5, Toronto, Ontario, Canada, June 2021. IEEE. Copyright 2021 IEEE. Published in ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), scheduled for 6-11 June 2021 in Toronto, Ontario, Canada. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works, must be obtained from the IEEE. Contact: Manager, Copyrights and Permissions / IEEE Service Center / 445 Hoes Lane / P.O. Box 1331 / Piscataway, NJ 08855-1331, USA. Telephone: + Intl. 908-562-3966. URL: https://inria.hal.science/hal-03132013, doi:10.1109/ICASSP39728.2021.9414238. 19, 22, 40

[LGR22] Q.T Le, R. Gribonval, and E. Riccietti. Code for reproducible research - "Spurious Valleys, NP-hardness, and Tractability of Sparse Matrix Factorization With Fixed Support", May 2022. URL: https://hal.inria.fr/hal-03667186. 46, 94

[Lig73] James Lighthill. Artificial intelligence: A general survey, 1973. 11

[LMG16] Luc Le Magoarou and Rémi Gribonval. Flexible Multi-layer Sparse Approximations of Matrices and Applications. *IEEE Journal of Selected Topics in Signal Processing*, 10(4):688–700, June 2016. URL: https://inria.hal.science/hal-01167948, doi:10.1109/JSTSP.2016.2543461. 35, 36, 37, 38, 46, 47

[LMQ21] Lek-Heng Lim, Mateusz Michalek, and Yang Qi. Best k-layer neural network approximations. *Constructive Approximation*, June 2021. URL: https://inria.hal.science/hal-03088287. 18, 54, 162, 163, 196

[LRC+21] Rui Lin, Jie Ran, King Hung Chiu, Graziano Chesi, and Ngai Wong. Deformable butterfly: A highly structured and sparse linear transform. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 16145–16157. Curran Associates, Inc., 2021. URL: https://proceedings.neurips.cc/paper/2021/file/86b122d4358357d834a87ce618a55de0-Paper.pdf. 20, 26, 27, 29, 112, 113, 115, 116, 118, 119, 123, 138, 163

[LRG22] Quoc-Tung Le, Elisa Riccietti, and Rémi Gribonval. Spurious Valleys, NP-hardness, and Tractability of Sparse Matrix Factorization With Fixed Sup-

port. *SIAM Journal on Matrix Analysis and Applications*, 2022. URL: https://hal.archives-ouvertes.fr/hal-03364668. 19, 21, 22, 44, 52, 55, 71, 82

[LRG23] Quoc-Tung Le, Elisa Riccietti, and Rémi Gribonval. Does a sparse ReLU network training problem always admit an optimum? In *Thirty-seventh Conference on Neural Information Processing Systems*, Advances in Neural Information Processing Systems 36 (NeurIPS 2023), New Orleans (Lousiane), United States, December 2023. URL: https://inria.hal.science/hal-04108849. 19, 21, 22, 35, 52, 159, 196

[LWF+15] Baoyuan Liu, Min Wang, Hassan Foroosh, Marshall Tappen, and Marianna Penksy. Sparse convolutional neural networks. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 806–814, 2015. doi:10.1109/CVPR.2015.7298681. 34

[LWM+07] Edo Liberty, Franco Woolfe, Per-Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert. Randomized algorithms for the low-rank approximation of matrices. *Proceedings of the National Academy of Sciences*, 104(51):20167–20172, 2007. URL: https://www.pnas.org/doi/abs/10.1073/pnas.0709640104, arXiv:https://www.pnas.org/doi/pdf/10.1073/pnas.0709640104, doi:10.1073/pnas.0709640104. 41

[LY17] Yingzhou Li and Haizhao Yang. Interpolative butterfly factorization. *SIAM Journal on Scientific Computing*, 39(2):A503–A531, 2017. 157

[LYM+15] Yingzhou Li, Haizhao Yang, Eileen R Martin, Kenneth L Ho, and Lexing Ying. Butterfly factorization. *Multiscale Modeling & Simulation*, 13(2):714–732, 2015. 113, 117, 157

[LZRG22] Q.T Le, L. Zheng, E. Riccietti, and R. Gribonval. Fast learning of fast transforms, with guarantees. In *ICASSP 2022 - IEEE International Conference on Acoustics, Speech and Signal Processing*, Singapore, Singapore, May 2022. This paper is associated to code for reproducible research available at https://hal.inria.fr/hal-03552956. URL: https://hal.inria.fr/hal-03438881. 19, 21, 40, 48, 113, 114, 119, 120, 121, 126, 127, 128

[LZT18] Q. Li, Z. Zhu, and G. Tang. The non-convex geometry of low-rank matrix optimization. *Information and Inference: A Journal of the IMA*, 8(1):51–96, 03 2018. arXiv:https://academic.oup.com/imaiai/article-pdf/8/1/51/28053147/iay003.pdf, doi:10.1093/imaiai/iay003. 74, 75

[Mai10] Julien Mairal. *Sparse coding for machine learning, image processing and computer vision*. Theses, École normale supérieure de Cachan - ENS Cachan, November 2010. URL: https://theses.hal.science/tel-00595312. 17, 40

[Mar11] P. G. Martinsson. A fast randomized algorithm for computing a hierarchically semiseparable representation of a matrix. *SIAM Journal on*

Matrix Analysis and Applications, 32(4):1251–1274, 2011. arXiv:https://doi.org/10.1137/100786617, doi:10.1137/100786617. 197

[MBP+08] Julien Mairal, Francis Bach, J. Ponce, Guillermo Sapiro, and Andrew Zisserman. Supervised dictionary learning. *Advances in Neural Information Processing Systems 21 - Proceedings of the 2008 Conference*, 09 2008. 17, 40

[MBPS09] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 689–696, New York, NY, USA, 2009. Association for Computing Machinery. doi:10.1145/1553374.1553463. 17, 40

[MBPS10] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online Learning for Matrix Factorization and Sparse Coding. *Journal of Machine Learning Research*, 11(1):19–60, January 2010. revised version. URL: https://inria.hal.science/inria-00408716. 17, 40

[MDFFF17] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 16

[MKBO18] Marina Munkhoeva, Yermek Kapushev, Evgeny Burnaev, and Ivan Oseledets. Quadrature-based features for kernel approximation. *Advances in neural information processing systems*, 31, 2018. 114

[MKC21] Scott Mahan, Emily J. King, and Alex Cloninger. Nonclosedness of sets of neural networks in sobolev spaces. *Neural Networks*, 137:85–96, may 2021. 162, 171

[MLN19] Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper/2019/file/2c601ad9d2ff9bc8b282670cdd54f69f-Paper.pdf. 47

[MMT+19] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz. Importance estimation for neural network pruning. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11256–11264, Los Alamitos, CA, USA, jun 2019. IEEE Computer Society. URL: https://doi.ieeecomputersociety.org/10.1109/CVPR.2019.01152, doi:10.1109/CVPR.2019.01152. 32

[MP43] Warren Mcculloch and Walter Pitts. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:127–147, 1943. 10

[MPCB14] Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 2924–2932, Cambridge, MA, USA, 2014. MIT Press. 174

[MS89] Michael C. Mozer and Paul Smolensky. *Skeletonization: A Technique for Trimming the Fat from a Network via Relevance Assessment*, page 107–115. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1989. 31, 32

[MYSsS20] E. Malach, G. Yehudai, S. Shalev-shwartz, and O. Shamir. Proving the lottery ticket hypothesis: Pruning is all you need. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020. 35, 195

[Ngu19] Quynh Nguyen. On connected sublevel sets in deep learning, 2019. arXiv:1901.07417. 72, 73, 74, 75

[NH17] Q. Nguyen and M. Hein. The loss surface of deep and wide neural networks, 2017. arXiv:1704.08045. 74

[NSH18] M. Nasr, R. Shokri, and A. Houmansadr. Machine learning with membership privacy using adversarial regularization. In *SIGSAC*. ACM, 2018. 15

[NT09] D. Needell and J.A. Tropp. Cosamp: Iterative signal recovery from incomplete and inaccurate samples. *Applied and Computational Harmonic Analysis*, 26(3):301 – 321, 2009. URL: http://www.sciencedirect.com/science/article/pii/S1063520308000638, doi:https://doi.org/10.1016/j.acha.2008.07.002. 32, 45

[NW06] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, second edition, 2006. 33, 71, 73, 138

[OHR20] Laurent Orseau, Marcus Hutter, and Omar Rivasplata. Logarithmic pruning is all you need. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. 35, 195

[OJ05] Pavel Okunev and Charles R. Johnson. Necessary and sufficient conditions for existence of the lu factorization of an arbitrary matrix, 2005. arXiv:math/0506382. 55, 167

[O'N07] Michael Patrick O'Neil. *A new class of analysis-based fast transforms*. Yale University, 2007. 113, 114

[Par95] Douglass Stott Parker. *Random butterfly transformations with applications in computational linear algebra*. UCLA Computer Science Department, 1995. 114

[Pee00] R. Peeters. The maximum edge biclique problem is np-complete. *Discrete Appl Math*, 131, 02 2000. 63

[Per23] Billy Perrigo. Exclusive: Openai used kenyan workers on less than two dollars per hour to make chatgpt less toxic, 2023. URL: https://time.com/6247678/openai-chatgpt-kenya-workers/. 12

[PNI+18] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. 02 2018. 12, 14

[PRN+20] Ankit Pensia, Shashank Rajput, Alliot Nagle, Harit Vishwakarma, and Dimitris Papailiopoulos. Optimal lottery tickets via subsetsum: Logarithmic over-parameterization is sufficient. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. 35, 195

[PRV21] Philipp Petersen, Mones Raslan, and Felix Voigtlaender. Topological properties of the set of functions generated by neural networks of fixed size. *Found. Comput. Math.*, 21(2):375–444, apr 2021. doi:10.1007/s10208-020-09461-0. 21, 52, 161, 162, 163, 171, 173, 183, 184, 185, 186

[PST01] Daniel Potts, Gabriele Steifdl, and M. Tasche. Fast fourier transforms for nonequispaced data: a tutorial. *Mod. Sampl. theory*, 23:19–25, 01 2001. 48

[QML20] Y. Qi, M. Michałek, and L-H Lim. Complex best r-term approximations almost always exist in finite dimensions. *Applied and Computational Harmonic Analysis*, 49(1):180–207, 2020. URL: https://www.sciencedirect.com/science/article/pii/S1063520317301847, doi:https://doi.org/10.1016/j.acha.2018.12.003. 66

[RM87] David E. Rumelhart and James L. McClelland. *Learning Internal Representations by Error Propagation*, pages 318–362. 1987. 14

[RWC+19] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. URL: https://api.semanticscholar.org/CorpusID:160025533. 12, 14

[RWK+19] Vivek Ramanujan, Mitchell Wortsman, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari. What's hidden in a randomly weighted neural network? *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11890–11899, 2019. URL: https://api.semanticscholar.org/CorpusID:208512842. 34, 35

[SAF+23] S Shekofteh, Christian Alles, Holger Fröning, et al. Reducing memory requirements for the ipu using butterfly factorizations. *arXiv preprint arXiv:2309.08946*, 2023. 116

[SCHU17] Simone Scardapane, Danilo Comminiello, Amir Hussain, and Aurelio Uncini. Group sparse regularization for deep neural networks. *Neurocomputing*, 241:81–89, 2017. URL: https://www.sciencedirect.com/science/article/pii/S0925231217302990, doi:https://doi.org/10.1016/j.neucom.2017.02.029. 34

[SD88] Sietsma and Dow. Neural net pruning-why and how. In *IEEE 1988 International Conference on Neural Networks*, pages 325–333 vol.1, 1988. doi:10.1109/ICNN.1988.23864. 31, 32

[SG22] Pierre Stock and Rémi Gribonval. An Embedding of ReLU Networks and an Analysis of their Identifiability. *Constructive Approximation*, 2022. URL: https://hal.archives-ouvertes.fr/hal-03292203, doi:10.1007/s00365-022-09578-1. 175, 184

[SGM20] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for modern deep learning research. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(09):13693–13696, Apr. 2020. URL: https://ojs.aaai.org/index.php/AAAI/article/view/7123, doi:10.1609/aaai.v34i09.7123. 11, 12

[SJG+20] Pierre Stock, Armand Joulin, Rémi Gribonval, Benjamin Graham, and Hervé Jégou. And the bit goes down: Revisiting the quantization of neural networks. In *International Conference on Learning Representations*, 2020. URL: https://openreview.net/forum?id=rJehVyrKwH. 12

[SL06] V. Silva and L.-H. Lim. Tensor rank and the ill-posedness of the best low-rank approximation problem. *SIAM Journal on Matrix Analysis and Applications*, 30:1084–1127, 08 2006. doi:10.1137/06066518X. 52, 54, 65, 66

[SLJ+15] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, Los Alamitos, CA, USA, jun 2015. IEEE Computer Society. URL: https://doi.ieeecomputersociety.org/10.1109/CVPR.2015.7298594, doi:10.1109/CVPR.2015.7298594. 16

[SLL19] David R. So, Chen Liang, and Quoc V. Le. The evolved transformer. In *International Conference on Machine Learning*, 2019. URL: https://api.semanticscholar.org/CorpusID:59523610. 12

[SQW16] J. Sun, Q. Qu, and J. Wright. A geometric analysis of phase retrieval. In *2016 IEEE International Symposium on Information Theory (ISIT)*, pages 2379–2383, 2016. doi:10.1109/ISIT.2016.7541725. 74, 75

[SSB17] Suraj Srinivas, Akshayvarun Subramanya, and R. Venkatesh Babu. Training sparse neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 455–462, 2017. doi:10.1109/CVPRW.2017.61. 34

[SSSS] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *IEEE Symposium on Security and Privacy*. 15

[SZ15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015. 16, 30, 34

[SZS+13] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. 12 2013. 16

[TBI97] L. N. Trefethen and D. Bau III. *Numerical linear algebra*, volume 50. SIAM, 1997. 83

[Tib96] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996. URL: http://www.jstor.org/stable/2346178. 16

[TKTH18] L. Theis, I. Korshunova, A. Tejani, and F. Huszár. Faster gaze prediction with dense networks and fisher pruning. arXiv:1801.05787, 2018. URL: https://arxiv.org/abs/1801.05787. 33

[TKYG20] Hidenori Tanaka, Daniel Kunin, Daniel L. K. Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. 26, 30, 163

[TLFF18] Enzo Tartaglione, Skjalg Lepsøy, Attilio Fiandrotti, and Gianluca Francini. Learning sparse neural networks via sensitivity-driven regularization. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 3882–3892, Red Hook, NY, USA, 2018. Curran Associates Inc. 32

[Use23] Deci User. An overview of state of the art (sota) dnns, May, 2023. URL: https://deci.ai/blog/sota-dnns-overview/. 15

[VBB20] Luca Venturi, Afonso S. Bandeira, and Joan Bruna. Spurious valleys in two-layer neural network optimization landscapes, 2020. arXiv:1802.06384. 18, 72, 73, 74, 75, 103

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf. 11, 12, 14, 26, 30

[VV22] Antoine Vanderschueren and Christophe Vleeschouwer. Are straight-through gradients and soft-thresholding all you need for sparse training?, 12 2022. 18

[WFR19] Mitchell Wortsman, Ali Farhadi, and Mohammad Rastegari. *Discovering Neural Wirings*. Curran Associates Inc., Red Hook, NY, USA, 2019. 18

[WGFZ19] Chaoqi Wang, Roger Grosse, Sanja Fidler, and Guodong Zhang. EigenDamage: Structured pruning in the Kronecker-factored eigenbasis. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 6566–6575. PMLR, 2019. URL: http://proceedings.mlr.press/v97/wang19g.html. 33

[Whi63] B. W. White. *The American Journal of Psychology*, 76(4):705–707, 1963. URL: http://www.jstor.org/stable/1419730. 14, 19, 22

[WWW+16] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 2082–2090, Red Hook, NY, USA, 2016. Curran Associates Inc. 34

[WZG20] Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations*, 2020. URL: https://openreview.net/forum?id=SkgsACVKPH. 26, 30, 163

[XWR19] Xia Xiao, Zigeng Wang, and Sanguthevar Rajasekaran. Autoprune: Automatic network pruning by regularizing auxiliary parameters. In *Neural Information Processing Systems*, 2019. URL: https://api.semanticscholar.org/CorpusID:202787073. 32

[YSKX22] Xiangli Yang, Zixing Song, Irwin King, and Zenglin Xu. A survey on deep semi-supervised learning. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–20, 2022. doi:10.1109/TKDE.2022.3220219. 12

[YWL20] Huanrui Yang, Wei Wen, and Hai Li. Deephoyer: Learning sparser neural network with differentiable scale-invariant sparsity measures. In *International Conference on Learning Representations*, 2020. URL: https://openreview.net/forum?id=rylBK34FDS. 34

[ZBH+21] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Commun. ACM*, 64(3):107–115, feb 2021. doi:10.1145/3446776. 15

[ZG17] Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression, 2017. arXiv:1710.01878. 18, 26, 31, 32, 33, 47, 163, 170

[ZGR21] L. Zheng, R. Gribonval, and E. Riccietti. Identifiability in exact multilayer sparse matrix factorization. *CoRR*, abs/2110.01230, 2021. URL: https://arxiv.org/abs/2110.01230, arXiv:2110.01230. 48, 93

[ZPR+23] Léon Zheng, Gilles Puy, Elisa Riccietti, Patrick Pérez, and Rémi Gribonval. Butterfly factorization by algorithmic identification of rank-one blocks. *arXiv preprint arXiv:2307.00820*, 2023. 116

[ZRG23] Léon Zheng, Elisa Riccietti, and Rémi Gribonval. Efficient identification of butterfly sparse matrix factorizations. *SIAM Journal on Mathematics of Data Science*, 5(1):22–49, 2023. 113, 114, 119, 120, 121, 123, 128

[ZSEW18] Zhihui Zhu, Daniel Soudry, Yonina C. Eldar, and Michael B. Wakin. The global optimization geometry of shallow linear neural networks. *CoRR*, abs/1805.04938, 2018. URL: http://arxiv.org/abs/1805.04938, arXiv:1805.04938. 71, 73, 74, 75

[ZU19] Wenyuan Zeng and Raquel Urtasun. MLPrune: Multi-layer pruning for automated neural network compression, 2019. URL: https://openreview.net/forum?id=r1g5b2RcKm. 33

[ZY06] Xiaoqin Zeng and Daniel S. Yeung. Hidden neuron pruning of multilayer perceptrons using a quantified sensitivity measure. *Neurocomputing*, 69(7):825–837, 2006. New Issues in Neurocomputing: 13th European Symposium on Artificial Neural Networks. URL: https://www.sciencedirect.com/science/article/pii/S0925231205001852, doi:https://doi.org/10.1016/j.neucom.2005.04.010. 32

[ZZXZ21] Xiao Zhou, Weizhong Zhang, Hang Xu, and Tong Zhang. Effective sparsification of neural networks with global sparsity constraint. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3599–3608, 2021. 18