Learning to Discover Graphical Model Structures

Eugene Belilovsky, Kyle Kastner, Gaël Varoquaux, Matthew Blaschko

Learning to Discover Graphical Model Structures

Eugene Belilovsky
INRIA Galen
University of Paris-Saclay
Chatenay-Malabry, France
eugene.belilovsky@inria.fr

Kyle Kastner
MILA
University of Montreal
Montreal, Canada
kyle.kastner@umontreal.ca

Gael Varoquaux
INRIA Parietal
Saclay,France
gael.varoquaux@normalesup.org

Matthew Blaschko
KU Leuven
Leuven, Belgium
matthew.blaschko@esat.kuleuven.be

Abstract

We consider structure discovery of undirected graphical models from observational data. Inferring likely structures from few examples is a complex task often requiring the formulation of priors and sophisticated inference procedures. In the setting of Gaussian Graphical Models (GGMs) a popular estimator is a penalized maximum likelihood objective on the precision matrix. Adapting this objective to capture domain-specific knowledge as priors or a new data likelihood requires great effort. In addition, structure recovery is a very indirect consequence of the data-fit term. By contrast, it may be easier to generate training samples of data that arise from graphs with the desired properties. We propose here to leverage this latter source of information in order to learn a function that maps from empirical covariance matrices to estimated graph structures. Learning this function brings two benefits: it implicitly models the desired structure or sparsity properties to form suitable priors, and it can more directly be tailored to the specific problem of edge structure discovery. We apply this framework to several critical real world problems in structure discovery and show that it can be competitive to standard approaches such as graphical lasso, at a fraction of the execution speed. We use convolutional neural networks to parametrize our estimators due to the compositional block structure of matrix inversion. Experimentally, our learnable graph-discovery method trained on synthetic data generalizes well to different data: identifying relevant edges in real data, completely unknown at training time. We find that on genetics, brain imaging, and simulation data we obtain competitive (and often superior) performance, compared with analytical methods.

1 Introduction

Probabilistic graphical models provide a powerful framework for describing the dependencies between a set of variables. Many applications require inferring the structure of a probabilistic graphical model from data to elucidate the relationships between variables. These relationships are often represented by an undirected graphical model also known as a Markov Random Field (MRF). We focus on a common MRF model, Gaussian graphical models (GGMs). GGMs are used in structure-discovery settings for rich data such as neuroimaging, genetics, or finance [30, 26]. Although multivariate Gaussian distributions have many nice properties, when there are few samples and the data is high dimensional, determining likely structures from few examples is a complex task requiring strong priors, typically a sparsity assumption, or other restrictions on the structure of the graph, making the distribution hard to characterize.

A standard approach to estimating sparse GGMs in high dimensions is based on the classic result that the zeros of a precision matrix correspond to zero partial correlation, a necessary and sufficient condition for conditional independence [18]. Assuming only a few conditional dependencies corresponds to a sparsity constraint on the entries of the precision matrix, leading to a combinatorial problem. Many popular approaches to learning GGMs can be seen as leveraging the ℓ_1-norm to create convex surrogates to this problem. [23] use nodewise ℓ_1 penalized regressions. Other estimators penalize the precision matrix directly [4, 10, 29]. The most popular being the graphical lasso

$$f_{glasso}(\hat{\mathbf{\Sigma}}) = \arg\min_{\mathbf{\Theta} \succ 0} - \log |\mathbf{\Theta}| + \mathrm{Tr}\left(\hat{\mathbf{\Sigma}}\mathbf{\Theta}\right) + \lambda \|\mathbf{\Theta}\|_1, \tag{1}$$

which can be seen as a penalized maximum likelihood estimator. Here $\mathbf{\Theta}$ and $\hat{\mathbf{\Sigma}}$ are the precision and sample covariance matrices, respectively. A large variety of alternative regularization penalties, extend the priors of the graphical lasso [6, 30]. However, several problems arise in this approach. Constructing novel surrogates for structured sparsity assumptions on MRF structures is challenging, as a prior needs to be formulated and incorporated into a penalized maximum likelihood objective which then needs an efficient optimization algorithm to be developed, often within a separate research effort. Furthermore, model selection in a penalized maximum likelihood setting is difficult as regularization parameters are often unintuitive.

Rather than designing an estimator, we propose framing the problem of finding a graph estimation procedure as selecting a function from a large flexible function class by way of risk minimization. This allows us to construct a loss function that explicitly aims to recover the edge structure. We can often sample from a distribution of graphs and empirical covariances with our desired properties, even though this distribution may not be analytically tractable. As such we can perform empirical risk minimization to select an appropriate function to use for edge estimation. Such a framework allows us to more easily control the assumed level of sparsity (as opposed to graph lasso) or impose structure on the sampling to shape the expected distribution, while optimizing a desired performance metric.

For particular cases, we show the problem of interest can be solved with a polynomial function, which is learnable with a neural network [1]. Motivated by this fact and theoretical and empricial results on learning smooth functions approximating solutions to combinatorial problems [5, 33], we propose to use a convolutional neural network as the function class. We train it by sampling from small sample data, generated from graphs with the prescribed properties, with a primary focus on sparse graphical models. Small sample ($n < p$) covariance matrices are estimated and given to the neural network (Figure 1) as input training data, which are to be mapped to indicators of present and absent edges in the underlying GGM. The learned network can then be employed in various real-world structure discovery problems.

In Section 1.1 we review the the related work. In Section 2 we formulate the risk minimization view of graph structure inference and describe how it can apply to sparse GGMs. Section 2.3 describes and motivates the deep learning architecture we chose to use for the sparse GGMs problem in this work. In Section 3 we describe the details of how we train an edge estimator for sparse GGMs. We then evaluate its properties extensively on simulation data. Finally, we show that this edge estimator trained only on synthetic data can obtain state of the art performance at inference time on real neuroimaging and genetics problems, while being much faster to execute than other methods.

1.1 Related Work

[22] analyzes learning functions to identify the structure of directed graphical models in causal inference. The learned functions take in estimates of kernel-mean embeddings. As in our work they demonstrate the use of simulations for training while testing on real data. Unlike our work they primarily focus on finding the causal direction in two node graphs with many observations.

Our learning architecture is motivated by the recent literature on deep networks. In [33] it was shown that neural networks can learn approximate solutions to NP-hard combinatorial problems, and the problem of optimal edge recovery in MRFs can be seen as a combinatorial optimization problem. Several recent works have been proposed which show neural architectures for graph input data [15, 9, 21]. These are based on multi layer convolutional networks, as in our work, or multi-step recurrent neural networks. The input in our approach can be viewed as a complete graph, while the ouput a sparse graph, thus none of these are directly applicable. A related use of deep networks to approximate a posterior distribution can be found in [2]. Finally, [13] use deep networks to approximate steps of a known sparse recovery algorithm.

Bayesian approaches to structure learning rely on priors on the graph combined with sampling techniques to estimate the posterior of the graph structure. Some approaches make assumptions on the decomposability of the graph [24]. The G-Wishart distribution is a popular distribution which forms part of a framework for structure inference, and advances have been recently made in efficient sampling [25]. These methods can still be rather slow compared to competing methods, and in the setting of $p > n$ we find they are less powerful.

2 Methods

2.1 Learning an Approximate Edge Estimation Procedure

In this section we consider MRF edge estimation as a learnable function. Let $\boldsymbol{X} \in R^{n \times p}$ be a matrix whose n columns are i.i.d. samples $x \sim P(x)$ of dimension p. Let $G = (V, E)$ be an undirected and unweighted graph associated with the set of variables in x. Let $\mathcal{L} = \{0, 1\}$ and $N_e = \frac{p(p-1)}{2}$ the maximum possible edges in E. Let $Y \in \mathcal{L}^{N_e}$ indicate the presence or absence of edges in the edge set E of G, namely

$$Y^{ij} = \begin{cases} 0 & x_i \perp x_j | x_{V \setminus i,j} \\ 1 & x_i \not\perp x_j | x_{V \setminus i,j} \end{cases} \tag{2}$$

We define an approximate structure discovery method $g_w(\boldsymbol{X})$, which produces a prediction of the edge structure, $\hat{Y} = g_w(\boldsymbol{X})$, given a set of data $\boldsymbol{X}$. In the rest of this paper we focus on $\boldsymbol{X}$ arising from Gaussian variables for which it can be more straightforward to consider $\hat{\boldsymbol{\Sigma}}$, the empirical covariance matrix corresponding to $\boldsymbol{X}$. In this case we will denote $g_w(\boldsymbol{X}) := f_w(\hat{\boldsymbol{\Sigma}})$. f_w is parametrized by w and belongs to the function class $\mathcal{F}$. We note that the graphical lasso in Equation (1) is an f_w for an appropriate choice of $\mathcal{F}$.

This view on the edge estimator now allows us to bring the selection of f_w from the domain of human design to the domain of empirical risk minimization over $\mathcal{F}$. Defining a distribution $\mathbb{P}$ on $R^{n \times p} \times \mathcal{L}^{N_e}$ such that $(\hat{\boldsymbol{\Sigma}}, Y) \sim \mathbb{P}$, we would like our estimator, f_w, to minimize the expected risk

$$R(f) = \mathbb{E}_{(\hat{\boldsymbol{\Sigma}}, Y) \sim \mathbb{P}}[l(f(\hat{\boldsymbol{\Sigma}}), Y)] \tag{3}$$

Here $l : R^{n \times p} \times \mathcal{L}^{N_e} \to \mathbb{R}^+$ is the loss function. For graphical model selection the $0/1$ loss function, $l = \mathbb{I}(Y \neq \hat{Y})$, is the natural error metric to consider [34]. The estimator with minimum risk is generally not possible to compute as a closed form expression for most interesting choices of $\mathbb{P}$, such as those arising from sparse graphs. In this setting it has been shown that (1) achieves the information theoretic optimal recovery rate up to a constant for certain $\mathbb{P}$ corresponding to uniformly sparse graphs with a maximum degree, but only when the optimal λ is used [34, 29].

The design of the estimator in Equation (1) is not explicitly minimizing this risk functional. Thus modifying the estimator to fit a different class of graphs (e.g small-world networks) while minimizing $R(f)$ is not obvious. Furthermore in practical settings the optimal λ is unknown. We would prefer to directly minimize the risk functional. The desired structural assumptions on samples from $\mathbb{P}$, such as sparsity, on the underlying graph mean the distribution is not tractable for analytic solutions. Meanwhile, we can often devise a sampling procedure for $\mathbb{P}$ allowing us to select an appropriate function via empirical risk minimization. Thus it is sufficient to define a rich enough $\mathcal{F}$ over which we can minimize the empirical risk over the samples generated, giving us a learning objective over N samples $\{Y_k, \boldsymbol{\Sigma}_k\}_{k=1}^N$ drawn from $\mathbb{P}$: $\min_w \frac{1}{N} \sum_{k=1}^N l(f_w(\hat{\boldsymbol{\Sigma}}_k), Y_k)$. To maintain tractability, we use the standard cross-entropy loss as a convex surrogate:

$$l(f_w(\hat{\boldsymbol{\Sigma}}), Y) = \sum_{i \neq j} \left(Y^{ij} \log(f_w^{ij}(\hat{\boldsymbol{\Sigma}})) + (1 - Y^{ij}) \log(1 - f_w^{ij}(\hat{\boldsymbol{\Sigma}})) \right) \tag{4}$$

We now need to select a sufficiently rich function class for f_w and a method to produce appropriate $(Y, \hat{\boldsymbol{\Sigma}})$ which model our desired data priors. This will allow us to learn a f_w that explicitly attempts to minimize errors in edge discovery.

2.2 Discovering Sparse Gaussian Graphical Model and Beyond

We discuss how the described approach can be applied to recover sparse Gaussian graphical models. A typical assumption in many modalities is that the number of edges is sparse. A convenient property of these GGMs is that the precision matrix has a zero value in the (i, j)th entry precisely when variables i and j are independent conditioned on all others. Additionally, the precision matrix and partial correlation matrix have the same sparsity pattern, while the partial correlation matrix has normalized entries.

Algorithm 1 Training a GGM edge estimator

for i $\in \{1, .., N\}$ **do**
  Sample $G_i \sim P(G)$
  Sample $\mathbf{\Sigma}_i \sim P(\mathbf{\Sigma}|G = G_i)$
  $\boldsymbol{X}_i \leftarrow \{x_j \sim N(0, \mathbf{\Sigma})\}_{j=1}^{n}$
  Construct $(Y_i, \hat{\mathbf{\Sigma}}_i)$ pair from $(G_i, \boldsymbol{X_i})$
end for
Select Function Class $\mathcal{F}$ (e.g. CNN)
Optimize: $\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{k=1}^{N} l(f(\hat{\mathbf{\Sigma}}_k), Y_k))$

We propose to simulate our *a priori* assumptions of sparsity and Gaussianity and attempt to learn $f_w(\hat{\mathbf{\Sigma}})$, which can then produce predictions of edges from the input data. The empirical covariance $\hat{\mathbf{\Sigma}}$ is a natural input for this task as, for Gaussian distributions, the empirical covariance is a sufficient statistic for the population covariance and hence the conditional independence structure. We model $P(x|G)$ as arising from a sparse prior on the graph G and correspondingly the entries of the precision matrix $\mathbf{\Theta}$. To obtain a single sample of $\boldsymbol{X}$ corresponds to n i.i.d. samples from $\mathcal{N}(0, \mathbf{\Theta}^{-1})$. We can now train $f_w(\hat{\mathbf{\Sigma}})$ by generating sample pairs $(\hat{\mathbf{\Sigma}}, Y)$. At execution time we standardize the input data and compute the covariance matrix before evaluating $f_w(\hat{\mathbf{\Sigma}})$. The process of learning f_w for the sparse GGM is given in Algorithm 1. A rather generic sparsity prior is one where each edge is equally likely with small probability, versus structured sparsity where edges have specific configurations. For obtaining the training samples $(\hat{\mathbf{\Sigma}}, Y)$ in this case we would like to create a sparse precision matrix, $\mathbf{\Theta}$, with the desired number of zero entries distributed uniformly. One strategy to do this and assure the precision matrices lie in the positive definite cone is to first construct an upper triangular sparse matrix and then multiply it by its transpose. This process is described in detail in the experimental section. Alternatively, an MCMC based G-Wishart distribution sampler can be employed if specific structures of the graph are desired [20].

The sparsity patterns in real data are often not uniformly distributed. Many real world networks are known to form small-world networks: graphs that are sparse and yet have a comparatively short average distance between nodes. These transport properties often hinge on a small number of hubs: high-degree nodes. Normally, such structural patterns require sophisticated adaptation when applying estimators like Eq 1. Indeed, high-degree nodes break the small-sample, sparse-recovery properties of ℓ_1-penalized estimators [29]. In our framework such structure assumption appears as a prior that can be learned offline during training of the neural network. Similarly priors on other distributions such as general exponential families can be more easily integrated. As the structure discovery model can be trained offline even a slow sampling procedure can suffice.

2.3 Neural Network Graph Estimator

To motivate the form of f_w consider the case when $n \gg p$. Then $\hat{\Sigma} \approx \Sigma$ and thus zeros of $\hat{\Sigma}^{-1}$ give the edge structure. In this case it suffices for f_w to mimic the inverse operation. A well known relation from linear algebra is $\Sigma^{-1}_{i,j} = \frac{(-1)^{i+j}}{\det(\Sigma)} \det(\tilde{\Sigma}_{i,j})$ where $\det(\tilde{\Sigma}_{i,j})$ refers to the (i, j)th minor of Σ. Here $\det(\Sigma)$, common to all (i, j), acts as a scale factor, while $(-1)^{i+j}$ doesn't affect the magnitude. The key term therefore is $\det(\tilde{\Sigma}_{i,j})$, a polynomial of degree $p - 1$. Theorem 4.2 in [1] shows, using gradient descent, a neural network with only two layers learns a polynomial function of degree d to arbitrary precision given sufficient hidden units. Consequently, $f_w(\hat{\mathbf{\Sigma}})$ is well parametrized by a neural network and learnable with standard non-convex optimization methods.

There are many architectural decisions one could make when applying a neural network, especially on a positive semidefinite matrix as done here. In the $\hat{\Sigma} \approx \Sigma$ example, the polynomials we wish to learn have a specific structure dictated by $\det(\tilde{\Sigma}_{i,j})$ which we would like to exploit, furthermore the target operation $(\det)$ is the same for each (i, j), suggesting a shared computation. Compositionality in neural networks which exploits structural patterns in the data, can often improve performance. Indeed, a driving force in the improvement of deep learning methods has been the use of compositional structures such as convolutional and recurrent layers that allow the learning of parameters exploiting regularity in image, speech, or text data. Here the determinant has a compositional structure with repeated algebraic block operations at multiple scales, this motivates the application of a series of convolutional layer directly to $\hat{\Sigma}$. Convolutional layers allow the network to easily learn computations

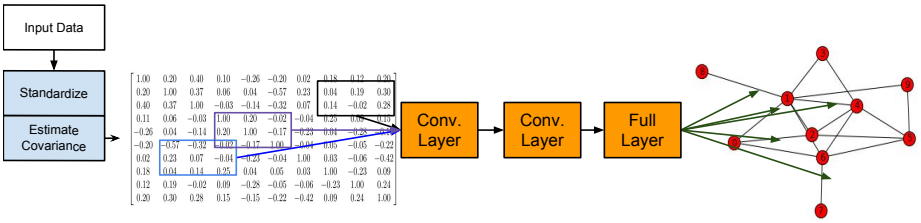

Figure 1: Diagram of the DeepGraph structure discovery architecture used in this work. The input is first standardized and then the sample covariance matrix is estimated. A neural network consisting of two convolutional and one fully connected layer is used to predict edges corresponding to non-zero entries in the precision matrix.

similar to determinants and share them amongst output variables. We found that this substantially improved generalization performance compared to using fully connected layers. Theoretical analysis of convolutional networks [5], has also shown they are very effective at learning smooth functions. Thus the form of $f_w(\hat{\mathbf{\Sigma}})$ we use in the experiments is as follows $h_1 = g(\text{conv}(\hat{\mathbf{\Sigma}}, W_1))$, $h_2 = g(\text{conv}(h_1, W_2))$, $h_3 = g(W_3 h_2)$, $f_w(\hat{\mathbf{\Sigma}}) = \sigma(W_4 h_3)$, where $g(\cdot)$ is an activation function, $\text{conv}(\cdot, \cdot)$ a convolutional layer, and $\sigma(\cdot)$ a sigmoid (Figure 1).

Returning to the scenario $n \gg p$, the sparsity and structure assumptions on $\hat{\Sigma}^{-1}$ generally considered in GGMs suggest the target function is simpler and can be made more efficient than the inverse. This motivates learning the function from data. As n becomes smaller the target function further changes and becomes intractable analytically, while an adaptive procedure can continue to find efficient computations to recover the graph structure.

3 Experiments

In our experiments we explore how well networks trained on parametric samples generalize, both to unseen synthetic data and to several real world problems. In order to highlight the generality of the learned networks, we apply the same network to multiple domains. We train two networks, DeepGraph-39 which takes in 39×39 covariance matrices, and DeepGraph-50 for 50×50. The sizes are chosen based on the real data we consider in subsequent sections. In both networks, we have 100 feature maps of 2×2 kernels in the first convolutional layer, and 50 feature maps of 2×2 kernels in the 2nd layer. The fully connected layer has 3000 hidden nodes. We use ReLU activations. The last layer is a sigmoid outputing a value of 0 to 1 for each edge.

Each network is trained continously with new samples generated until the validation error saturates. For a given precision matrix we generate 5 possible $\boldsymbol{X}$ samples to be used as training data. A total of approximately 1M training samples are used for each network. The networks are optimized using ADAM[17] coupled with cross-entropy loss as the objective function (cf. Sec. 2.1). We use batch normalization at each layer. In order to encourage the networks to converge to sparser solutions we add a small regularization term: $0.01\|f_w(\hat{\mathbf{\Sigma}})\|_2$, which improves convergence for our expectedly low outputs. Additionally, we found that using the absolute value of the true partial correlations as labels, instead of hard binary labels, improves results on the validation set.

We sample $P(X|G)$ with a sparse prior on $P(G)$ as follows. We first construct lower diagonal matrix, L, where each entry has α probability of being zero. Non-zero entries are set uniformly between -0.9 and 0.9. Multiplying LL^T gives a sparse positive definite precision matrix, Ω. This gives us our $P(\Omega|G)$ with a sparse prior on $P(G)$. We sample from the Gaussian $\mathcal{N}(0, \Omega^{-1})$ to obtain samples of $\boldsymbol{X}$. Here α corresponds approximately to a specific sparsity level in the final precision matrix, which we set to produce matrices $92 - 96\%$ sparse. Our experimental evaluations focus on the challenging high dimensional settings in which $p > n$ and consider both synthetic data and real data from genetics and neuroimaging.

Synthetic Data Evaluation To understand the properties of our learned networks, we evaluated them on different synthetic data than the ones they were trained on. More specifically, we used a completely different third party sampler so as to avoid any contamination. For all experiments DeepGraph-39 was used. The same trained network is utilized in the subsequent neuroimaging evaluations as well.

We used the `BDGraph` R-package to produce sparse precision matrices based on the G-Wishart distribution [25] as well as the R-package `rags2ridges` [28] to generate data from small-world networks corresponding to the Watts–Strogatz model. We compared our learned estimator against the `scikit-learn` [27] implementation of Graphical Lasso with regularizer chosen by cross-validation as well as the Birth-Death Rate MCMC (BDMCMC) method from [25].

We note the optimal graph structure is independent from the order of the input dimensions, a property inherent in existing structure recovery methods. However, the network must learn this invariance property from the data. Premuting the input data we can obtain another estimate of the output. Thus we also show the result of an average of 20 permutations of input to DeepGraph-39, this generally improves the result. In Table 1 we show evaluations for edge recovery in different scenarios.

For each scenario we repeat the experiment for 20 different graphs and small sample observations showing the average area under the ROC curve (AUC), precision@k corresponding to 2.5% of possible edges, and calibration error (CE) [25]. For graphical lasso we use the partial correlations to indicate confidence in edges; `BDGraph` automatically returns posterior probabilities as does our method.

For the case of random Gaussian graphs with n=35 (as in our training data), and graph sparsity of 95%, we have superior performance in terms of AUC and calibration error. Graphical Lasso performs better in terms of precision at the relevant levels. Our method is superior to the Birth-Death MCMC in all categories. Next we apply the method to a less straightforward synthetic data, with distributions typical of many applications. We found that, compared to baseline methods, our network performs particularly well in with high-degree nodes. In particular our method performs well on the relevant metrics with small-world networks, a very common family of graphs in real-world data, obtaining superior precision at the primary levels of interest. Figure 2 shows examples of random, Watts and Strogatz small-world graphs, and concentrated hub graphs used in these experiments.

Training a new network for each number of samples can pose difficulties with our proposed method. Thus we evaluted how robust the network DeepGraph-39 is to input covariances obtained from fewer or more samples. We find that overall the performance is quite good even when lowering the number of samples to $n = 15$, we obtain superior performance to the other approaches in a number of sample sizes and metrics (Table 1). We also applied DeepGraph-39 on data from a multivariate generalization of the Laplace distribution[11]. As in other experiments precision matrices were sampled from the G-Wishart at a sparsity of 95%. [11, Proposition 3.1] was applied to produce samples. We find that DeepGraph-39 still performs competitively, despite the discrepancy between train and test distributions. Experiments with variable sparsity are considered in the supplementary material, which find that for very sparse graphs, the networks remain robust in performance, while for increased density performance degrades but remains competitive.

Using the small-world network data generator [28], we demonstrate that we can update the generic sparse prior to a structured one. We re-train DeepGraph-39 using only 1000 examples mixed with 1000 examples from the original uniform sparsity model. We perform just one epoch of training and observe markedly improved performance on this test case as seen in the last row of Table 1. We compute the average execution time of our method compared to Graph Lasso and BDGraph on a CPU in Table 2. We note that we use a production quality version of graph lasso [27], whereas we have not optimized the network execution, for which known strategies may be applied [7].

Cancer Genome Data We perform experiments on a gene expression dataset described in [16]. The data come from a cancer genome atlas from 2360 subjects for various types of cancer. We used the first 50 genes from Appendix C.2 of [16] of commonly regulated genes in cancer. We evaluated on two groups of subjects, one with breast invasive carcinoma (BRCA) consisting of 590 subject samples and the other colon adenocarcinoma (CODA) consisting of 174 samples.

Evaluating edge selection in real world data is challenging. We use the following methodology: for each method we select the top-k ranked edges, and then fix all other edges, recomputing the maximum likelihood precision matrix with support given by the corresponding edge selection method. We then evaluate the likelihood on a held-out set of data. We repeat this procedure for a range of k. We rely on Algorithm 0 in [14] to compute the maximum likelihood precision given a support. The experiment is repeated for each of CODA and BRCA subject groups 50 times, results are shown in Figure 3. In all cases we use 40 samples for edge selection and precision estimation. We compare with graphical lasso as well as the Ledoit-Wolf shrinkage estimator [19]. We additionally consider

Experimental Setup	Method	Prec@5%	AUC	Calibration Error
Gaussian Random Graphs (n=35)	GLasso	**0.53**	0.709	0.07
	Bdgraph	0.48	0.716	0.19
	DeepGraph-39	0.5	0.752	0.07
	DeepGraph-39+Perm	*0.52*	***0.772***	*0.07*
Gaussian Random Graphs (n=100)	GLasso	0.6	0.757	0.06
	Bdgraph	**0.62**	0.777	0.09
	DeepGraph-39	0.53	0.787	0.06
	DeepGraph-39+Perm	*0.55*	***0.81***	*0.06*
Gaussian Random Graphs (n=15)	GLasso	0.34	0.642	**0.07**
	Bdgraph	0.16	0.608	0.38
	DeepGraph-39	0.32	0.688	0.09
	DeepGraph-39+Perm	***0.37***	***0.695***	*0.09*
Laplace Random Graphs (n=35)	GLasso	**0.46**	0.698	0.07
	Bdgraph	0.27	0.666	0.28
	DeepGraph-39	0.42	0.708	0.08
	DeepGraph-39+Perm	*0.44*	***0.729***	*0.08*
Gaussian Hub Graphs (n=35)	GLasso	0.31	0.733	0.06
	Bdgraph	0.28	0.716	0.2
	DeepGraph-39	0.33	***0.801***	0.06
	DeepGraph-39+Perm	***0.34***	0.781	*0.06*
Gaussian Small-World Graphs (n=35)	Glasso	0.34	0.582	0.11
	Bdgraph	0.45	0.695	0.17
	DeepGraph-39	0.43	0.687	0.11
	DeepGraph-39+Perm	*0.45*	***0.709***	*0.11*
	DeepGraph-39+Update	0.54	0.761	0.13
	DeepGraph-39+Update+Perm	***0.59***	***0.779***	0.14

Table 1: For each case we generate 20 graphs and data matrices sampled (with n samples) from distributions with those underlying graphs. DeepGraph outperforms other methods in terms of AUC. In terms of precision at expected sparsity levels DeepGraph has better performance for graphs with high-degree nodes such as small world and hub graphs (shown in Figure 2) and when model selection is difficult for glasso (n=15).

	Avg. time (s)
`sklearn` GraphLassoCV	4.81
`BDgraph`	42.13
DeepGraph-50	***0.21***

Table 2: Average execution time over 10 trials for 50 node problem on a CPU for Graph Lasso, BDMCMC, and DeepGraph-50

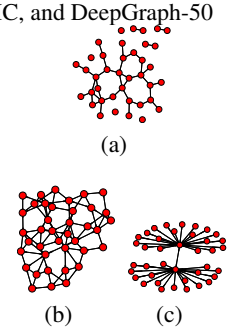

Figure 2: Example of (a) random (b) small world (c) hub networks used in experiments

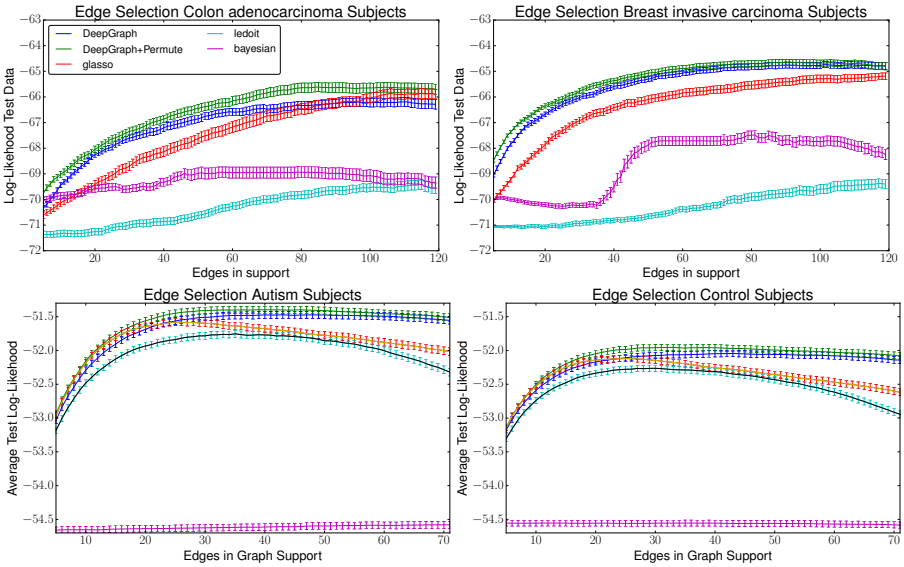

Figure 3: Average test likelihood for COAD and BRCA subject groups in gene data and neuroimaging data using different number of selected edges. Each experiment is repeated 50 times for genetics data. It is repeated 1000 times in the fMRI to obtain significant results due high variance in the data. DeepGraph with averaged permutation dominates in all cases for genetics data, while DeepGraph+Permuation is superior or equal to competing methods in the fMRI data.

the MCMC based approach described in previous section. For graphical lasso and Ledoit-Wolf, edge selection is based on thresholding partial correlation [3].

Additionally we evaluate the stability of the solutions provided by the various methods. This is important in several applications where one wants a low variance estimate of the edge set. We use Spearman correlations between pairs of solutions, as it is a measure of a monotone link between two variables. Results are shown in Table 3. DeepGraph has far better stability in all cases in this dataset.

	Gene BRCA	Gene COAD	ABIDE Control	ABIDE Autistic
Graph Lasso	$0.24 \pm .003$	0.32 ± 0.004	$0.22 \pm .003$	$0.25 \pm .003$
Ledoit-Wolfe	0.12 ± 0.002	0.15 ± 0.003	$0.13 \pm .003$	$0.14 \pm .003$
Bdgraph	0.06 ± 0.002	0.08 ± 0.003	N/A	N/A
DeepGraph	$\mathbf{0.74 \pm 0.004}$	$\mathbf{0.71 \pm 0.005}$	$\mathbf{0.29 \pm .004}$	$\mathbf{0.31 \pm .004}$
DeepGraph +Permute	0.60 ± 0.003	0.58 ± 0.0043	$0.22 \pm .004$	$0.22 \pm .004$

Table 3: Average Spearman correlation results for real data showing stability of solution amongst 50 trials

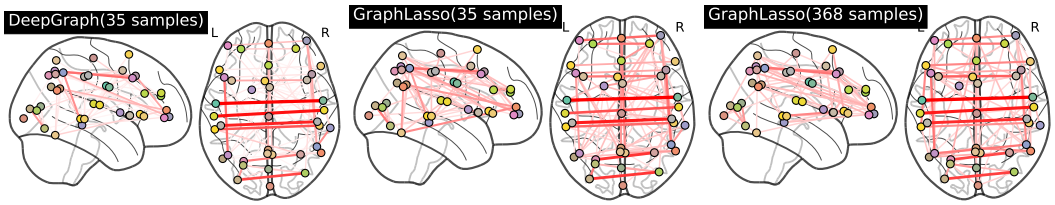

Figure 4: Example solution from DeepGraph and Graph Lasso in the small sample regime on the same 35 samples, along with a larger sample solution of Graph Lasso for reference. DeepGraph is able to extract similar key edges as graphical lasso

Resting State Functional Connectivity We evaluate our graph discovery method for the purpose of identifying brain functional connectivity in resting state fMRI data. Correlations in brain activity measured via fMRI reveal functional interactions between remote brain regions. These are an important measure to study psychiatric diseases that have no known anatomical support. Typical connectome analysis describes each subject or group by a GGM measuring functional connectivity between a set of regions [31]. We use the ABIDE dataset [8], a large scale resting state fMRI dataset. It gathers brain scans from 539 individuals suffering from autism spectrum disorder and 573 controls over 16 sites.[1] For our experiments we use an atlas with 39 regions of interest derived in [32].

We use the network DeepGraph-39, the same network from synthetic experiments, using the same evaluation protocol as used in the genom data. For both control and autism patients we use time series from 35 random subjects to estimate edges and corresponding precision matrices. We find that for both the Autism and Control group we can obtain edge selection comparable to graph lasso for very few selected edges. When the number of selected edges is in the range above 25 we begin to perform significantly better in edge selection as seen in Fig. 3. We evaluated stability of the results as shown in Tab. 3. DeepGraph outperformed the other methods across the board.

ABIDE has high variability across sites and subjects. To obtain low variance evaluation metrics we needed to perform 1000 folds to obtain well separated error bars, many more trials than were needed to see distinctions between approaches in the genetics experiment. We found that the birth-death MCMC method took very long to converge on this data, moreover the need for many folds to obtain significant results amongst the methods made this approach prohibitively slow to evaluate.

We show the edges returned by Graph Lasso and DeepGraph for a sample from 35 subjects (Fig. 4) in the control group. We also show the result of a large sample result based on 368 subjects from graphical lasso. In visual evaluation of the edges returned by DeepGraph we find that they closely align with expected results from a large sample estimation procedure. Furthermore we can see several edges in the subsample which were particularly strongly activated in both methods.

4 Discussion and Conclusions

Our method was competitive with strong baselines, particularly in the case of high-degree nodes and higher order cliques we found that our learned estimator performed better. When re-trained on the target distribution performance further improves. Most importantly the estimator generalizes well to real data finding relevant stable edges. We also observed that the learned estimators generalize to variations not seen at training time (e.g. different n or sparsity), learning generic computations. This points to potential to more easily scale the method to different graph sizes. One could consider transfer learning, where a network for one size of data is used as a starting point to learn a network working on larger dimension data. For very high dimensional settings, with $p \gg n$, it may make sense to input the data matrix directly and utilize a variable length architecture similar to [33].

Penalized maximum likelihood can provide performance guarantee under restrictive assumptions on the form of the distribution and not considering the regularization path. In the proposed method one could obtain empirical bounds under the prescribed data distribution. Additionally, at execution time

[1] `http://preprocessed-connectomes-project.github.io/abide/`

the speed of the approach can allow for re-sampling based uncertainty estimates and efficient model selection (e.g cross-validation) amongst several trained estimators.

We have introduced the concept of learning an estimator for determining the structure of an undirected graphical model. We proposed a network architecture and sampling procedure for learning such an estimator for the case of sparse GGMs. We obtained competitive results on synthetic data with various underlying distributions, as well as on challenging real-world data. We found that our method works particularly well compared to other approaches for small-world networks, an important class of graphs occurring frequently in real-world domains. We have shown that neural networks can obtain improved results over various statistical methods on real datasets, despite being trained with samples from parametric distributions. Our approach can allow for straightforward specifications of new priors and open new directions in efficient graphical structure discovery from few examples.

References

[1] A. Andoni, R. Panigrahy, G. Valiant, and L. Zhang. Learning polynomials with neural networks. In *ICML*, 2014.

[2] A. K. Balan, V. Rathod, K. Murphy, and M. Welling. Bayesian dark knowledge. In *NIPS*, 2015.

[3] S. B. S. Balmand and A. S. Dalalyan. On estimation of the diagonal elements of a sparse precision matrix. *arXiv:1504.04696*, 2015.

[4] T. Cai, W. Liu, and X. Luo. A constrained ℓ_1 minimization approach to sparse precision matrix estimation. *Journal of the American Statistical Association*, 106(494):594–607, 2011.

[5] N. Cohen, O. Sharir, and A. Shashua. On the expressive power of deep learning: a tensor analysis. *arXiv:1509.05009*, 2015.

[6] P. Danaher, P. Wang, and D. M. Witten. The joint graphical lasso for inverse covariance estimation across multiple classes. *Journal of the Royal Stat. Society(B)*, 76(2):373–397, 2014.

[7] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *NIPS*, 2014.

[8] A. Di Martino et al. The autism brain imaging data exchange: Towards a large-scale evaluation of the intrinsic brain architecture in autism. *Molecular psychiatry*, 19:659, 2014.

[9] D. K. Duvenaud et al. Convolutional networks on graphs for learning molecular fingerprints. In *NIPS*, 2015.

[10] J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

[11] E. Gómez, M. Gomez-Viilegas, and J. Marin. A multivariate generalization of the power exponential family of distributions. *Commun Stat Theory Methods*, 27(3):589–600, 1998.

[12] A. Graves. *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer, 2012.

[13] K. Gregor and Y. LeCun. Learning fast approximations of sparse coding. In *ICML*, 2010.

[14] H. Hara and A. Takemura. A localization approach to improve iterative proportional scaling in Gaussian graphical models. *Commun Stat Theory Methods*, 39(8-9):1643–1654, 2010.

[15] M. Henaff, J. Bruna, and Y. LeCun. Deep convolutional networks on graph-structured data. *arXiv:1506.05163*, 2015.

[16] J. Honorio, T. Jaakkola, and D. Samaras. On the statistical efficiency of $\ell_{1,p}$ multi-task learning of Gaussian graphical models. *arXiv:1207.4255*, 2012.

[17] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.

[18] S. L. Lauritzen. *Graphical models*. Oxford University Press, 1996.

[19] O. Ledoit and M. Wolf. A well-conditioned estimator for large-dimensional covariance matrices. *Journal of multivariate analysis*, 88(2):365–411, 2004.

[20] A. Lenkoski. A direct sampler for G-Wishart variates. *Stat*, 2(1):119–128, 2013.

[21] Y. Li et algi. Gated graph sequence neural networks. *ICLR*, 2016.

[22] D. Lopez-Paz et al. Towards a learning theory of cause-effect inference. In *ICML*, 2015.

[23] N. Meinshausen and P. Bühlmann. High-dimensional graphs and variable selection with the lasso. *The Annals of Statistics*, pages 1436–1462, 2006.

[24] B. Moghaddam, E. Khan, K. P. Murphy, and B. M. Marlin. Accelerating Bayesian structural inference for non-decomposable Gaussian graphical models. In *NIPS*, 2009.

[25] A. Mohammadi and E. C. Wit. Bayesian structure learning in sparse Gaussian graphical models. *Bayesian Analysis*, 10(1):109–138, 2015.

[26] K. Mohan, M. Chung, S. Han, D. Witten, S.-I. Lee, and M. Fazel. Structured learning of Gaussian graphical models. In *NIPS*, pages 620–628, 2012.

[27] F. Pedregosa et al. Scikit-learn: Machine learning in python. *JMLR*, 12:2825–2830, 2011.

[28] C. Peeters, A. Bilgrau, and W. van Wieringen. rags2ridges: Ridge estimation of precision matrices from high-dimensional data. *R package*, 2015.

[29] P. Ravikumar, M. J. Wainwright, G. Raskutti, and B. Yu. High-dimensional covariance estimation by minimizing ℓ_1-penalized log-determinant divergence. *EJS*, 5:935–980, 2011.

[30] S. Ryali et al. Estimation of functional connectivity in fMRI data using stability selection-based sparse partial correlation with elastic net penalty. *NeuroImage*, 59(4):3852–3861, 2012.
[31] G. Varoquaux and R. C. Craddock. Learning and comparing functional connectomes across subjects. *NeuroImage*, 80:405–415, 2013.
[32] G. Varoquaux, A. Gramfort, F. Pedregosa, V. Michel, and B. Thirion. Multi-subject dictionary learning to segment an atlas of brain spontaneous activity. In *IPMI*, 2011.
[33] O. Vinyals, M. Fortunato, and N. Jaitly. Pointer networks. In *NIPS*, 2015.
[34] W. Wang, M. J. Wainwright, and K. Ramchandran. Information-theoretic bounds on model selection for gaussian markov random fields. In *ISIT*, pages 1373–1377. Citeseer, 2010.

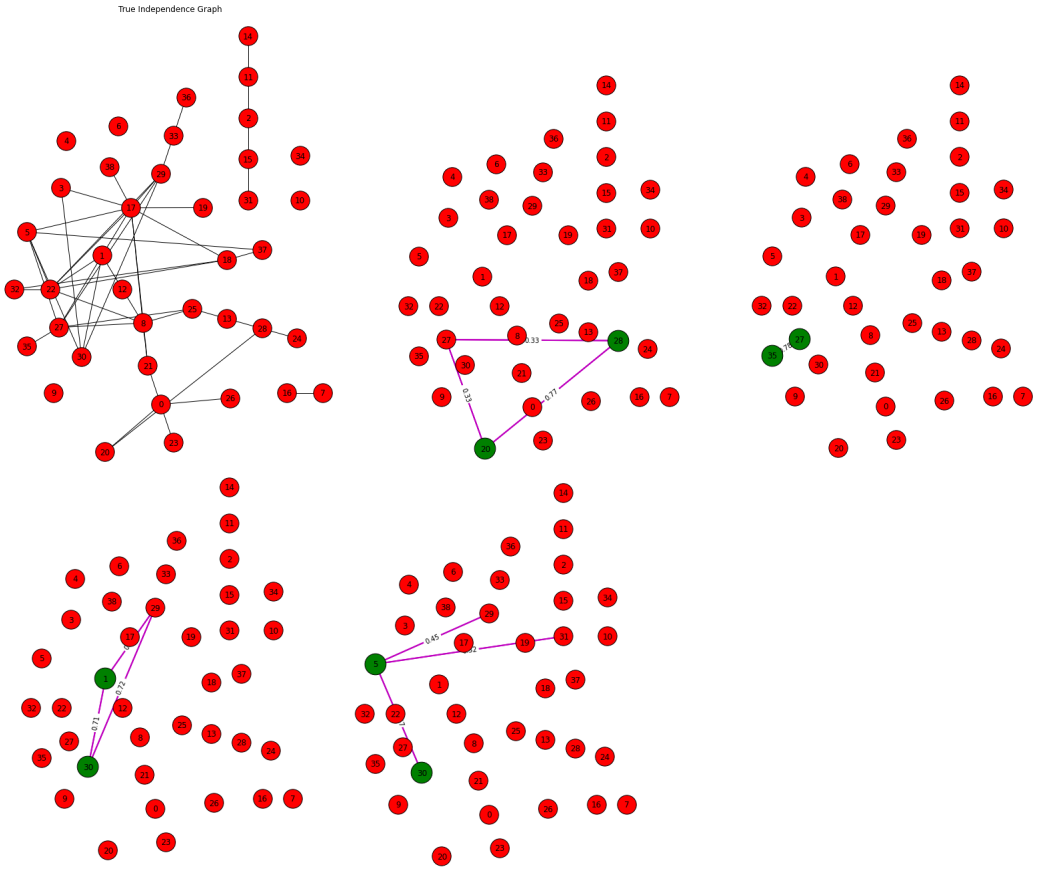

Figure 5: True graph structure and top activated partials corresponding to the covariance input for top activated outputs of the network. The two green nodes indicate the connection being evaluated, the magenta edges show the top partials corresponding to the input. We see the network is learning to associate outputs and inputs (not specified in any way) and potentially explore correlated nodes amongst the considered ones

A Supplementary Experiments

A.1 Analyzing the Network Jacobian

In [12] a method for loosely obtaining the attention of a trained neural network is used. The jacobian matrix of partial derivatives can be used for a given input to get an idea of which entries in the input matrix most affect the given outputs. We use this approach to begin to analyze the behavior of our DeepGraph-39 network. We gave one empirical covariance matrix to the network which was generated from the underlying graph specified in 5. We first note the jacobian is concentrated in just a few values, the top activated edges for the input closely correspond to the higher value partials in the overall jacobian matrix.

For the most activated output edges we visualize the entries or edges in the input (which can also be thought of as entries in a weighted adjacency matrix) that are most activated. We do this by taking a softmax of all partials for a given output edge and only keeping outliers (2 standard deviations). In 5 we show the top partials for 4 of the top output edges.

The relevant covariance entry corresponding to the edge at the output is almost always amongst the top partial values. We note that there is no explicit association of input relations outputs specified before or during training of the network so it has learned to associate the input and outputs. When there are other covariance entries pointed to they are usually connected to the relevant nodes either directly or indirectly. This may indicate the network is learning an algorithm to more succintly represent the connections of the two nodes.

A.2 Predicting Covariance Matrices

Using our framework it is possible to attempt to directly predict an accurate covariance matrix given a noisy one constructed from few observations. This is a more challenging task than predicting the edges. In this section we show preliminay experiments which given an empirical covariance matrix from few observations attempts to predict a more accurate covariance matrix that takes into account underlying sparse data dependency structure.

	mean $\|\hat{\Sigma} - \Sigma\|_2^2$	mean $\|\hat{\Sigma} - \Sigma\|_\infty$
Empirical	0.0267	0.543
Graph Lasso	0.0223	0.680
DeepGraph	0.0232	0.673

Table 4: Covariance prediction of ABIDE data. Averaged over 50 trials of 35 samples from the ABIDE Control data

Experimental Setup	Method	Prec@2.5%	AUC	CE
Gaussian Random Graph (n=35, sparsity=2%)	GLasso	0.41	0.701	0.02
	Bdgraph	0.39	0.727	0.15
	DeepGraph-39-G	0.42	0.744	0.03
	DeepGraph-39-G+Perm	0.42	0.752	0.03
Gaussian Random Graph (n=35, sparsity=15%)	GLasso	0.91	0.598	0.34
	Bdgraph	0.76	0.601	0.36
	DeepGraph-39-G	0.76	0.617	0.35
	DeepGraph-39-G+Perm	0.85	0.626	0.35

Table 5: For each scenario we generate 20 graphs with 39 nodes, and corresponding data matrix sampled from distributions with those underlying graphs. The number of samples is indicated by n.

One challenge is that outputs of our covariance predictor must be on the positive semidefinite cone, thus we choose to instead predict on the cholesky decompositions, which allows us to always produce positive definite covariances. We train the same DeepGraph-39 structure modifying the last layer to remove the non-linear sigmoid so that we can output covariance entry values, we train to predict on the cholesky decomposition of the true covariance matrices generated by our model with a squared loss.

We evaluate this network using the ABIDE dataset described in Section 3. The ABIDE data has a large number of samples allowing us to obtain a large sample estimate of the covariance and compare it to our estimator as well as graphical lasso and empirical covariance estimators. Using the large sample ABIDE empirical covariance matrix. We find that we can obtain competitive ℓ_2 and ℓ_∞ norm using few samples. We use 403 subjects from the ABIDE Control group each with a recording of $150 - 200$ samples to construct covariance matrix using a total of 77330 (some correlated) this acts as our very approximate estimate of the population Σ. We then evaluate covariance estimation on 35 samples using the empirical covariance estimator, graphical lasso, and DeepGraph trained to output covariance matrices. We repeat the experiment for 50 different subsamples of the data. We see in 4 that the prediction approach can obtain competitive results. In terms of ℓ_2 graphical lasso performs better, however our estimate is better than empirical covariance estimation and much faster then graphical lasso. In some applications such as robust estimation a fast estimate of the covariance matrix (automatically embedding sparsity assumptions) can be of great use. For ℓ_∞ error we see the empirical covariance estimation outperforms graphical lasso and DeepGraph for this dataset, while DeepGraph performs better in terms of this metric.

We note these results are preliminary, as the covariance predicting networks were not heavily optimized, moreover the ABIDE dataset is very noisy even when pre-processed and thus even the large sample covariance estimate may not be accurate. We believe this is an interesting alternate application of our paper.

A.3 Additional Synthetic Results on Sparsity

We investigate the affect of sparsity on DeepGraph-39 which has been trained with input that has sparsity $96\% - 92\%$ sparse. We find that DeepGraph performs well at the 2% sparsity level despite not seeing this at training time. At the same time performance begins to degrade for 15% but is still competitive in several categories. The results are shown in Table 5. Future investigation can consider how alternate variation of sparsity at training time will affect these results.