Recursive nearest agglomeration (ReNA): fast clustering for approximation of structured signals

Andrés Hoyos-Idrobo, Gaël Varoquaux, Jonas Kahn, Bertrand Thirion

Recursive nearest agglomeration (ReNA): fast clustering for approximation of structured signals

Andrés HOYOS-IDROBO, Gaël VAROQUAUX, Jonas KAHN, and Bertrand THIRION

Abstract—In this work, we revisit fast dimension reduction approaches, as with random projections and random sampling. Our goal is to summarize the data to decrease computational costs and memory footprint of subsequent analysis. Such dimension reduction can be very efficient when the signals of interest have a strong structure, such as with images. We focus on this setting and investigate feature clustering schemes for data reductions that capture this structure. An impediment to fast dimension reduction is then that good clustering comes with large algorithmic costs. We address it by contributing a linear-time agglomerative clustering scheme, Recursive Nearest Agglomeration (ReNA). Unlike existing fast agglomerative schemes, it avoids the creation of giant clusters. We empirically validate that it approximates the data as well as traditional variance-minimizing clustering schemes that have a quadratic complexity. In addition, we analyze signal approximation with feature clustering and show that it can remove noise, improving subsequent analysis steps. As a consequence, data reduction by clustering features with ReNA yields very fast and accurate models, enabling to process large datasets on budget. Our theoretical analysis is backed by extensive experiments on publicly-available data that illustrate the computation efficiency and the denoising properties of the resulting dimension reduction scheme.

Index Terms—clustering, dimensionality reduction, matrix sketching, classification, neuroimaging, approximation

1 Introduction

CHEAP and ubiquitous sensors lead to a rapid increase of data sizes, not only in the sample direction –the number of measurements– but also in the feature direction. *Features* refer here to the dimensions of each observation: pixels/voxels of images, time points of signals, loci of genotypes etc. These "big data" put a lot of strain on data management and analysis. Indeed, they entail large memory and storage footprint, and the algorithmic cost of querying or processing them is often super-linear in the data size. Yet, such data often display a low-dimensional structure, for instance originating from the physical process probed by the sensor. Thus, the data can be well approximated by a lower-dimension representation, dropping drastically the cost of subsequent data management or analysis. The present paper focuses on these fast signal approximations.

The deluge of huge sensor-based data is ubiquitous: in imaging sciences –*e.g.* biological [1] or medical [2]–, genomics [3], [4], and seismology [5], to name a few applications. Taming the computational costs created by the rapid increase in signal resolution is an active research question. Many approaches integrate reduced signal representations in statistical analysis [1], [6], [7], [8], [9].

In signal processing and machine learning, fast signal approximation is central to speeding up algorithms: approximating kernels [10], fast approximate nearest neighbors [11], or randomized linear algebra [12]. Note that for all these, the only requirement on the reduced representation is that it preserves pairwise distances between signals. data reduction makes processing huge data sets much easier as it decreases both memory requirements and computation time. Indeed, even for linear complexity algorithms, the computation cost of growing data size is worse than linear once data no longer fit in cache or memory.

There are a wide variety of standard data-compression approaches. Typically, a data matrix is represented by a sketch matrix that is significantly smaller than the original, but approximates it well. Two sketching strategies are commonly employed: *i)* approximating the matrix by a small subset of its rows (or columns) (e.g. Nyström [13] and CUR [14]); *ii)* randomly combining matrix rows, relying on subspace embedding and strong concentration phenomena, e.g. random projections [15]. Random projections are appealing as they come with theoretical guaranties on the expected distortion. For information retrieval, state-of-the-art indexing of times series can be achieved with a symbolic representation [16] that finds a regular piecewise constant approximation of the signal.

Here, we are interested in representing *structured* signals, and using this structure to improve the data approximation and speed up its computation. Such signals are often modeled as generated from a random process acting on a topology. Individual features of the data then form vertices of a graph. Edges can be predefined by the specificity of the acquisition process, such as the physics of the sensors. Thus, the connectivity of the features is independent from the data themselves. Henceforth, we refer to this connectivity as *structure*[1]. Note that such a description is not limited to regular grids, such as time-series or images, and encompasses

- *A. Hoyos-Idrobo, G.Varoquaux and B.Thirion are with Parietal team, Inria, CEA, University Paris-Saclay, 91191, Gif sur Yvette, France. E-mails: firstname.lastname@inria.fr*
- *J. Kahn is with the Institut de Mathématiques de Toulouse, UMR5219, Université de Toulouse, CNRS, France. E-mail: jonas.kahn@math.univ-toulouse.fr*

1. We consider the structure as prior information, and assume it static across time.

for instance data on a folded surface [17].

Our contribution focuses on fast dimension reduction that adapts to common statistics across the data, as with the Nyström approximation, but unlike random projections. Our goal is to speed up statistical analysis, *e.g.* machine learning, benefiting from the structure of the data. For this, we use *feature grouping*, approximating a signal by partitioning its feature space into several subsets and replacing the value of each subset with a constant value. We use a clustering algorithm to adapt the partition to the data statistics. Agglomerative clustering algorithms are amongst the fastest approaches to extract many clusters with graph-connectivity constraints. However, they fail to create clusters of evenly-distributed size, favoring a few huge clusters[2].

Contributions: our contributions are two-fold.

i) We analyze dimensionality reduction of structured signals by feature grouping. We show that it has a denoising effect on these signals, hence improves subsequent statistical analysis. *ii)* We introduce a fast agglomerative clustering that is well suited to perform the feature grouping. This clustering algorithm finds clusters of roughly even size in linear time, maintaining meaningful information on the structure of the data. Our pipeline is very beneficial for analysis of large-scale structured datasets, as the dimension reduction is very fast, and it reduces the computational cost of various statistical estimators without losing accuracy.

The paper is organized as follows. In section 2, we give prior art on fast dimension reduction and analyze the theoretical performance of feature grouping. In section 3, we introduce ReNA, a new fast clustering algorithm. In section 4, we carry out extensive empirical studies comparing many fast dimension-reduction approaches and show on real-life data that feature grouping can have a denoising effect.

Notations: Column vectors are written using bold lower-case, e.g., $\mathbf{x}$. For a vector $\mathbf{x}$, the i-th component of $\mathbf{x}$ is denoted $\mathbf{x}_i$. Matrices are written using bold capital letters, e.g., $\mathbf{X}$. The jth column vector of $\mathbf{X}$ is denoted $\mathbf{X}_{*,j}$, the ith row vector of $\mathbf{X}$ is denoted $\mathbf{X}_{i,*}$, and $[n]$ denotes $\{1, \ldots, n\}$. Letters in calligraphic, e.g. $\mathcal{P}$ denotes sets or graphs, and it will be clarified by the context. Let $\{\mathcal{C}_i\}_{i=1}^k$ be short for the set $\{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$. $|\cdot|$ denotes the cardinality of a set. The ℓ_p norm of a vector $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k] \in \mathbb{R}^k$ is defined as $\|\mathbf{x}\|_p = \left(\sum_{i=1}^k |\mathbf{x}_i|^p\right)^{\frac{1}{p}}$, for $p = [1, \infty)$.

2 Dimension reduction of structured signals

In this section, we review useful prior art on random projections and random sampling. Then we analyze signal approximation with feature grouping.

2.1 Background and related prior art

Signal approximation with random projections or random sampling techniques is now central to many data analysis, machine learning, or signal processing algorithms.

Let $\mathbf{X} \in \mathbb{R}^{p \times n}$ be a data matrix composed of n samples and p features (i.e. pixels/voxels). We are interested in an operator $\mathbf{\Phi} \in \mathbb{R}^{k \times p}$ that reduces the dimension of the data in the feature direction, acting as a preprocessing step to make further analysis more tractable. This operator should maintain approximately the pairwise distance between pairs of samples $(\mathbf{X}_{*,i}, \mathbf{X}_{*,j}) \in \mathbf{X}^2$ for $(i, j) \in [n]^2$,

$$\|\mathbf{\Phi}\, \mathbf{X}_{*,i} - \mathbf{\Phi}\, \mathbf{X}_{*,j}\|_2^2 \approx \|\mathbf{X}_{*,i} - \mathbf{X}_{*,j}\|_2^2, \;\; \forall (i,j) \in [n]^2. \quad (1)$$

Note that this approximation needs to hold only on the data submanifold, and not the entire $\mathbb{R}^p$.

Random projections: A standard choice is to build $\mathbf{\Phi}$ with random projections, $\mathbf{\Phi}_{\mathsf{RP}}$ [19]. It is particularly attractive due its algorithmic simplicity and theoretical guaranties that make it ϵ-isometric (see Eq. 2).

Lemma 2.1. By the Johnson-Lindenstrauss lemma [15], the pairwise distances among a collection $\mathcal{X}$ of n-points in $\mathbb{R}^p$ are approximately maintained when the points are mapped randomly to an Euclidean space of dimension $k = O(\epsilon^{-1} \log n)$ up to a distortion at most ϵ. More precisely, given $\epsilon, \delta \in (0, 1)$ and $k \leq p$, there exists a random linear projection $\mathbf{\Phi}_{\mathsf{RP}} : \mathbb{R}^p \to \mathbb{R}^k$ such that for every $\mathbf{x}, \mathbf{x}'$ in $\mathcal{X}$, the following relations hold:

$$(1-\epsilon)\|\mathbf{x}-\mathbf{x}'\|_2^2 \leq \|\mathbf{\Phi}_{\mathsf{RP}}\, \mathbf{x} - \mathbf{\Phi}_{\mathsf{RP}}\, \mathbf{x}'\|_2^2 \leq (1+\epsilon)\|\mathbf{x}-\mathbf{x}'\|_2^2, \quad (2)$$

with probability at least $1 - \delta$.

Johnson-Lindenstrauss embeddings have been widely used in the last years. By providing a low-dimensional representation of the data, they can speed up algorithms dramatically, in particular when runtime depends super-linearly on the data dimensionality. In addition, as this representation of the data is accurate in the sense of the ℓ_2 norm, it can be used to approximate shift-invariant kernels [10], [20].

The $\mathbf{\Phi}_{\mathsf{RP}}$ matrix can be generated by sampling from a Gaussian distribution with rescaling. In practice, a simple and efficient generation scheme can yield a very sparse random matrix with good properties [21], [22].

This approach suffers from two important limitations: *i)* inverting the random mapping from $\mathbb{R}^p$ to $\mathbb{R}^k$ is difficult, requiring more constraints on the data (e.g. sparsity), which entails another estimation problem. As a result, it yields less meaningful or easily interpretable results, as the ensuing inference steps cannot be made explicit in the original space. *ii)* This approach is suboptimal for structured datasets, since it ignores the properties of the data, such as a possible spatial smoothness.

Random sampling: A related technique is random sampling, and in particular the Nyström approximation method. This method is mainly used to build a low-rank approximation of a matrix. It is particularly useful with kernel-based methods when the number n of samples is large, given that the complexity of building a kernel matrix is at least quadratic in n [23]. It has become a standard tool when dealing with large-scale datasets [13].

The idea is to preserve the spectral structure of a kernel matrix $\mathbf{K}$ using a subset of columns of this matrix, yielding a low-rank approximation. This can be cast as building a data-driven feature mapping $\mathbf{\Phi}_{\mathsf{Nys}} \in \mathbb{R}^{k \times p}$. In a linear setting, the kernel matrix is defined as $\mathbf{K} = \mathbf{X}^\mathsf{T}\, \mathbf{X}$, which leads to the following approximation:

$$\mathbf{K}_{i,j} = \langle \mathbf{X}_{*,i}, \mathbf{X}_{*,j} \rangle \approx \langle \mathbf{\Phi}_{\mathsf{Nys}}\, \mathbf{X}_{*,i}, \mathbf{\Phi}_{\mathsf{Nys}}\, \mathbf{X}_{*,j} \rangle . \quad (3)$$

2. This phenomenon is known as percolation in random graphs [18].

Here, building a base $\mathbf{\Phi}_{\text{Nys}}$ is achieved by randomly sampling $k \ll n$ points from $\mathbf{X}$, and then normalizing them –i.e. obtaining an orthogonal projector to the span of the subsampled data– see algorithm 2 in supplementary materials. The cost of the SVD dominates the complexity of this method $O(pk \min\{p, k\})$. This method is well suited for signals with a common structure, for instance images that share a common spatial organization captured by $\mathbf{\Phi}_{\text{Nys}}$. As the Nyström approximation captures the structure of the data, it can also act as a regularization [24].

2.2 Dimension reduction by feature grouping

Here we analyze feature grouping for signal approximation.

2.2.1 The feature-grouping matrix and approximation

Feature grouping defines a matrix $\mathbf{\Phi}$ that extracts piecewise constant approximations of the data [25]. Let $\mathbf{\Phi}_{\text{FG}}$ be a matrix composed with constant amplitude groups (clusters). Formally, the set of k clusters is given by $\mathcal{P} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k\}$, where each cluster $\mathcal{C}_q \subset [p]$ contains a set of indexes that does not overlap other clusters, $\mathcal{C}_q \cap \mathcal{C}_l = \emptyset$, for all $q \neq l$. Thus, $(\mathbf{\Phi}_{\text{FG}}\ \mathbf{x})_q = \alpha_q \sum_{j \in \mathcal{C}_q} \mathbf{x}_j$ yields a reduction of a data sample $\mathbf{x}$ on the q-th cluster, where α_q is a constant for each cluster. With an appropriate permutation of the indexes of the data $\mathbf{x}$, the matrix $\mathbf{\Phi}_{\text{FG}}$ can be written as

$$\mathbf{\Phi}_{\text{FG}} = \begin{bmatrix} \alpha_1 \text{—} \alpha_1 & 0 \text{—} 0 & \ldots & 0 \text{—} 0 \\ 0 \text{—} 0 & \alpha_2 \text{—} \alpha_2 & \ldots & 0 \text{—} 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 \text{—} 0 & 0 \text{—} 0 & \ldots & \alpha_k \text{—} \alpha_k \end{bmatrix} \in \mathbb{R}^{k \times p}.$$

We choose $\alpha_q = 1/\sqrt{|\mathcal{C}_q|}$ to set the non-zero singular values of $\mathbf{\Phi}_{\text{FG}}$ to 1, making it an orthogonal projection.

We call $\mathbf{\Phi}_{\text{FG}}\ \mathbf{x} \in \mathbb{R}^k$ the *reduced* version of $\mathbf{x}$ and $\mathbf{\Phi}_{\text{FG}}^{\mathsf{T}} \mathbf{\Phi}_{\text{FG}}\ \mathbf{x} \in \mathbb{R}^p$ the *approximation* of $\mathbf{x}$. Note that having an approximation of the data means that the ensuing inference steps can be made explicit in the original space. As the matrix $\mathbf{\Phi}_{\text{FG}}$ is sparse, this approximation follows the same principle as [26], speeding up computational time and reducing memory storage.

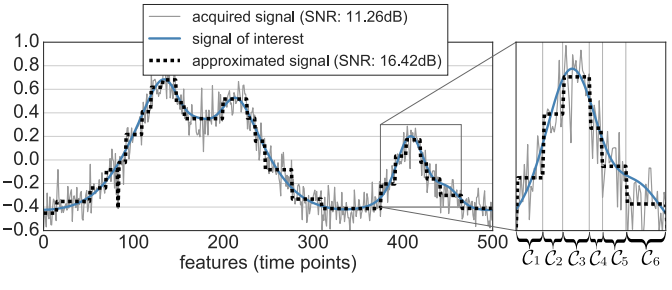


Figure 1. **Illustration of the approximation of a signal:** Piece-wise constant approximation of a 1D signal contaminated with additive Gaussian noise, $\mathbf{x} \in \mathbb{R}^{500}$. This signal is a sample of our statistical problem. The approximation is built by clustering features (here time-points) with a connectivity constraint (i.e. using a spatially-constrained Ward clustering). Only 25 clusters can preserve the structure of the signal and decrease the noise, as seen from the signal-to-noise-ratio (dB).

Let $M(\mathbf{x})$ be the approximation error for a data $\mathbf{x}$ given a feature grouping matrix $\mathbf{\Phi}_{\text{FG}}$,

$$M(\mathbf{x}) = \left\| \mathbf{x} - \mathbf{\Phi}_{\text{FG}}^{\mathsf{T}} \mathbf{\Phi}_{\text{FG}}\ \mathbf{x} \right\|_2^2, \tag{4}$$

this is often called inertia in the clustering literature. This corresponds to the sum of all the local errors (the approximation error for each cluster), $M(\mathbf{x}) = \sum_{q=1}^{k} m_q$, where $m_q(\mathbf{x})$ is the sum of squared differences between the values in the q-th cluster and its representative center, as follows

$$m_q(\mathbf{x}) = \left\| \mathbf{x}_{\mathcal{C}_q} - \frac{(\mathbf{\Phi}_{\text{FG}}\ \mathbf{x})_q}{\sqrt{|\mathcal{C}_q|}} \right\|_2^2, \tag{5}$$

where $\mathbf{x}_{\mathcal{C}_q} \in \mathbb{R}^{|\mathcal{C}_q|}$ are the values $\mathbf{x}_i$ such that $i \in \mathcal{C}_q$. The squared norm of the data $\mathbf{x}$ is then decomposed in two terms: fidelity and inertia, taking the form (see section 2 in supplementary materials):

$$\|\mathbf{x}\|_2^2 = \underbrace{\|\mathbf{\Phi}_{\text{FG}}\ \mathbf{x}\|_2^2}_{\text{Reduced norm}} + \underbrace{\sum_{q=1}^{k} \left\| \mathbf{x}_{\mathcal{C}_q} - \frac{(\mathbf{\Phi}_{\text{FG}}\ \mathbf{x})_q}{\sqrt{|\mathcal{C}_q|}} \right\|_2^2}_{M(\mathbf{x})\text{: Inertia}}. \tag{6}$$

Eq. 6 is key to understanding the desired properties of a matrix $\mathbf{\Phi}_{\text{FG}}$. In particular, it shows that it is beneficial to work in a large k regime to reduce the inertia.

2.2.2 Capturing signal structure

We consider data with a specific structure, e.g. spatial data. Well-suited dimensionality reduction can leverage this structure to bound the approximation error. We assume that the data $\mathbf{x} \in \mathbb{R}^p$ are generated from a process acting on a space with a neighborhood structure (topology). To encode this structure, the data matrix $\mathbf{X}$ is associated with an undirected graph $\mathcal{G}$ with p vertices $\mathcal{V} = \{v_1, v_2, \ldots, v_p\}$. Each vertex of the graph corresponds to column index in the data matrix $\mathbf{X}$ and the presence of an edge means that these features are connected. For instance, for 2D or 3D image data, the graph is a 2D or 3D lattice connecting neighboring pixels. The graph defines a graph distance between features $\text{dist}_{\mathcal{G}}$. In practice, we perform the calculations with the adjacency matrix $\mathbf{G}$ of the graph $\mathcal{G}$.

Definition 2.1. **L-Smoothness of the signal:** A signal $\mathbf{x} \in \mathbb{R}^p$ structured by a graph $\mathcal{G}$, is pairwise Lipschitz smooth with parameter L when it satisfies

$$|\mathbf{x}_i - \mathbf{x}_j| \leq L\, \text{dist}_{\mathcal{G}}(v_i, v_j), \quad \forall (i, j) \in [p]^2. \tag{7}$$

This definition means that the signal is smooth with respect to the graph that encodes the underlying structure. Note that $\text{dist}_{\mathcal{G}}$ has no unit since the scale is fixed by having each edge have length 1.

Lemma 2.2. Let $\mathbf{x} \in \mathbb{R}^p$ be a pairwise L-Lipschitz signal, and $\mathbf{\Phi}_{\text{FG}} \in \mathbb{R}^{k \times p}$ be a fixed feature grouping matrix, formed by $\{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$ clusters. Then the following holds:

$$\|\mathbf{x}\|_2^2 - L^2 \sum_{q=1}^{k} |\mathcal{C}_q|\, \text{diam}_{\mathcal{G}}(\mathcal{C}_q)^2 \leq \|\mathbf{\Phi}_{\text{FG}}\ \mathbf{x}\|_2^2 \leq \|\mathbf{x}\|_2^2, \tag{8}$$

where $\text{diam}_{\mathcal{G}}(\mathcal{C}_q) = \sup_{v_i, v_j \in \mathcal{C}_q} \text{dist}_{\mathcal{G}}(v_i, v_j)$.

See section 2 in supplementary materials for a proof.

We see that the approximation is better if:

i) The cluster sizes are about the same. Even if the clusters are balls of dimension d, the $\mathrm{diam}_{\mathcal{G}}(\mathcal{C}_q)$ is of order $|\mathcal{C}_q|^{1/d}$. We thus expect the clusters to be compact.

ii) The clusters have a small diameter. As the left-hand side of Eq.8 is upper bounded by $L^2\,\mathrm{diam}_{\mathcal{G}}(\mathcal{C}_q)^2$, and $\mathrm{diam}_{\mathcal{G}}(\mathcal{C}_q) \leq |\mathcal{C}_q|$. We see that the clusters have to be small.

These arguments are based only on the assumption of smoothness of the signal. Refining Eq. 8 gives an intuition on how a partition could be adapted to the data:

Corollary 2.1. Let L_q be the smoothness index inside cluster C_q, for all $q \in [k]$. This is the minimum L_q such that:

$$|\mathbf{x}_i - \mathbf{x}_j| \leq L_q\,\mathrm{dist}_{\mathcal{G}}(v_i,\, v_j), \quad \forall (i,j) \in C_q^2.$$

Then the following two inequalities hold:

$$\begin{aligned} \|\mathbf{x}\|_2^2 - \sum_{q=1}^{k} |\mathcal{C}_q| \sup_{\mathbf{x}_i,\mathbf{x}_j \in \mathbf{x}_{\mathcal{C}_q}} |\mathbf{x}_i - \mathbf{x}_j|_2^2 \leq \\ \|\mathbf{x}\|_2^2 - \sum_{q=1}^{k} L_q^2\,|\mathcal{C}_q|\,\mathrm{diam}_{\mathcal{G}}(\mathcal{C}_q)^2 \leq \|\mathbf{\Phi}_{\mathsf{FG}}\,\mathbf{x}\|_2^2. \end{aligned} \tag{9}$$

We can see that the approximation is better if: *i)* the signal in a cluster is homogeneous (low L_q); *ii)* clusters in irregular areas (high L_q) are smaller.

Clusters $\mathcal{P}$ of the graph $\mathcal{G}$ are defined as connected components of a subgraph. In this context, the size of the largest group, $\max_{q\in[k]} |\mathcal{C}_q|$, can be studied with percolation theory, that characterizes the appearance of a giant connect component as edges are added [18].

2.2.3 Approximating signals with unstructured noise

Noise hinders subsequent statistical estimation. Feature grouping can reduce the noise under certain conditions. To explore them, we consider an additive noise model: the acquired data $\mathbf{X}$ is a spatially-structured signal of interest $\mathbf{S}$ contaminated by unstructured noise $\mathbf{N}$,

$$\mathbf{X}_{*,i} = \mathbf{S}_{*,i} + \mathbf{N}_{*,i}, \quad \forall i \in [n]. \tag{10}$$

Applying the feature grouping matrix $\mathbf{\Phi}_{\mathsf{FG}}$ to the acquired signal reduces the noise via within-cluster averaging. In particular, i.i.d. noise with zero-mean and variance σ^2, leads to the following relation between the Mean Squared Error of the approximated data, $\mathrm{MSE}_{\mathrm{approx}}$, and the non-reduced one $\mathrm{MSE}_{\mathrm{orig}}$ (see section 3 in supplementary materials):

$$\mathrm{MSE}_{\mathrm{approx}} \leq L^2 \sum_{q=1}^{k} |\mathcal{C}_q|\,\mathrm{diam}_{\mathcal{G}}(\mathcal{C}_q)^2 + \frac{k}{p}\,\mathrm{MSE}_{\mathrm{orig}}, \tag{11}$$

where p denotes the number of features (i.e. pixels/voxels), and k denotes the number of clusters. This gives a denoising effect if the smoothness parameter satisfies

$$L^2 \leq \frac{(p-k)}{\sum_{q=1}^{k} |\mathcal{C}_q|\,\mathrm{diam}_{\mathcal{G}}(\mathcal{C}_q)^2}\,\sigma^2. \tag{12}$$

When the signal of interest is smooth enough and the cluster sizes are roughly even, the feature grouping will reduce the noise, preserving the information of the low-frequency signal $\mathbf{S}_i$. Fig. 1 presents a graphical illustration of the reduction and denoising capabilities of feature grouping.

The challenge is then to define a good $\mathbf{\Phi}_{\mathsf{FG}}$, given that data-unaware feature partitions are sub-optimal, as they do not respect the underlying structures and lead to signal loss.

3 ReNA: A fast structured clustering algorithm

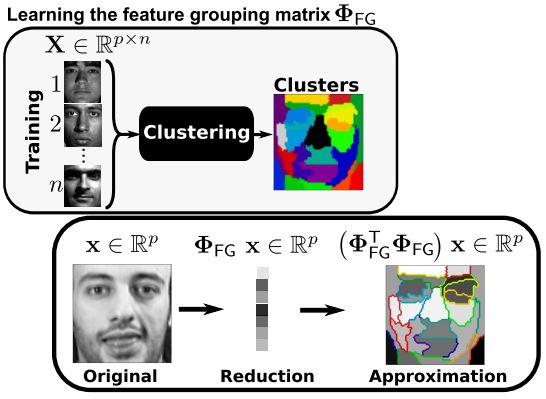


Figure 2. **Illustration of feature grouping:** In images, feature-grouping data approximation corresponds to a super-pixel approach. In a more general setting, this approach consists in finding a data-driven reduction $\mathbf{\Phi}_{\mathsf{FG}}$ using clustering of features. Then, the data $\mathbf{x}$ are reduced to $\mathbf{\Phi}_{\mathsf{FG}}\,\mathbf{x}$ and then used for further statistical analysis (e.g. classification).

In the feature-grouping setting above, we now consider a data-driven approach to build the matrix $\mathbf{\Phi}_{\mathsf{FG}}$. We rely on feature clustering: a clustering algorithm is used to define the groups of features from the data. $\mathbf{X} \in \mathbb{R}^{p\times n}$ is represented by a reduced version $\mathbf{\Phi}_{\mathsf{FG}}\,\mathbf{X}$, where p is potentially very large (greater than $100\,000$), whereas k is smaller but close enough to p (e.g. $k = \lfloor p/20 \rfloor$). As illustrated in Fig. 2, once the reduction operator $\mathbf{\Phi}_{\mathsf{FG}}$ has been learned, it can be applied to new data coming form the same generative process.

3.1 Existing fast clustering algorithms

K-means clustering is a natural choice as it minimizes the total inertia in Eq. 6. But it tends to be expensive in our setting: The conventional k-means algorithm has a complexity of $O(p\,k)$ per iterations [27]. However, the larger the number of clusters, the more iterations are needed to converge, and the worst case complexity is given[3] by $O(p^{k+2/n})$ [28]. This complexity becomes prohibitive with many clusters.

Super-pixel approaches: In computer vision, feature clustering can be related to the notion of *super-pixels* (super-voxels for 3D images). The most common fast algorithm for super-pixels is SLIC [27], which has a low computational cost and produces super-pixels/super-voxels of roughly even sizes. SLIC performs a local clustering of the image values with a spatial constraint, using as a distance measure the combination of two Euclidean distances: image values and spatial positions. The SLIC algorithm is related to K-means, but it performs a fixed small number of iterations, resulting in a complexity of $O(np)$. In the large-k regime, it can be difficult to control precisely the number of clusters, as some clusters often end up empty in the final assignment.

3. Note that here n and p are swapped compared to common clustering literature, as we are doing *feature* clustering.

Agglomerative clustering: Agglomerative clustering algorithms are fast in the setting of a large number k of clusters. Unlike most clustering algorithms, adding a graph structure constraint makes them even faster, as they can then discard associations between non-connected nodes.

Agglomerative clustering schemes start off by placing every data element in its own cluster, then they proceed by merging repeatedly the closest pair of connected clusters until finding the desired number of clusters [29]. Various methods share the same approach, differing only in the linkage criterion used to identify the clusters to be merged. The most common linkages are single, average, complete [29] and Ward [30]. Average-linkage, complete-linkage, and Ward are generally preferable over single-linkage, as they tend to yield more balanced clusters. Yet single-linkage clustering is often used as it is markedly faster; it can be obtained via a Minimum Spanning Tree and has a complexity of $O(np + p \log p)$ [31]. Single linkage is also related to finding connected components of a similarity graph [32], an approach often used to group pixels on images [33]. Average-linkage, complete-linkage and Ward are more costly, as they have a worst case complexity of $O(np^2)$ [31].

The approximation properties of feature grouping are given by the distribution of cluster sizes and the smoothness of the signal (Eq. 8). Balanced clusters are preferable for low errors. Nevertheless, agglomerative clustering on noisy data often leads to a "preferential attachment" behavior: large clusters grow faster than smaller ones. In this case, the largest cluster dominates the distortion, as in Lemma. 2.2. By considering the clusters as connected components on a similarity graph, this behavior can be linked to percolation theory [18], that characterizes the appearance of a giant connected component (i.e. a huge cluster). In this case, the clustering algorithm is said to *percolate*, and thus cannot yield balanced cluster sizes.

In brief, single-linkage clustering is fast but suffers form percolation issues [34] and Ward's algorithm performs often well in terms of goodness of fit for large k [35].

More sophisticated agglomerative strategies have been proposed in the framework of computer vision (e.g. [36]), but they have not been designed to avoid percolation and do not make it possible to control the number k of clusters.

3.2 Contributed clustering algorithm: ReNA

For feature clustering on structured signals, an algorithm should take advantage of the generative nature of the data, e.g. for images, work with local image statistics. Hence we rely on neighborhood graphs [37].

Neighborhood graphs form an important class of geometric graphs with many applications in signal processing, pattern recognition, or data clustering. They are used to model local relationships between data points, with ϵ-neighborhood graphs or k-nearest neighbor graphs. The ϵ-nearest neighbor graph is the core of clustering methods for which the number k of clusters is implicitly set by the ϵ neighborhood's radius [38]. K-nearest neighbor graphs are also used for clustering and theoretical results show that they can identify high-density modes of the samples [39]. However, measurement noise on between-sample similarities hinders their recovery of this structure as neighborhood graphs tend to percolate [4] for k greater than or equal to 2. In contrast, the 1-nearest neighbor graph (1-NN) is not likely to percolate [40]. For this reason, we use the 1-NN graph.

In a nutshell, our algorithm relies on extracting the connected components of a 1-NN graph. To reach the desired number k of clusters, we apply it recursively. The algorithm outline is as follows:

Initialization: We start by placing each of the p features of the data $\mathbf{X}$ in its own cluster $\mathcal{P} = \{\mathcal{C}_1, \ldots, \mathcal{C}_p\}$. We use the binary adjacency matrix $\mathbf{G} \in \{0,1\}^{p \times p}$ of the graph $\mathcal{G}$, that encodes the topological structure of the features (i.e. one values denote connected vertices, whereas zero represents non-connected vertices).

Nearest neighbor grouping: We build the similarity graph which encodes the affinity between features. We then find the nearest neighbor graph of this similarity graph, and extract the connected components of this subgraph to reduce the data matrix $\mathbf{X}$ and the topological structure $\mathbf{G}$. Hence, we use the 1-nearest neighbor of each vertex (i.e. feature) of the similarity graph as linkage criterion. These operations are summarized in the next steps:

1) **Graph representation:** We build the similarity graph $\mathcal{D}$ of the data $\mathbf{X}$, represented by the adjacency matrix $\mathbf{D} \in \mathbb{R}^{p \times p}$. The weights in $\mathbf{D}$ are only assigned for edges in $\mathbf{G}$ [5].
2) **Finding 1-NN:** Creating a 1-nearest neighbor graph $\mathcal{Q}$, represented by the matrix $\mathbf{Q} \in \{0,1\}^{p \times p}$, where each vertex of $\mathcal{D}$ is associated with its nearest neighbor in the sense of the dissimilarity measure (e.g. Euclidean distance, although a distance is not needed).
3) **Getting the clusters:** We use [41][6] to extract the set of connected components of $\mathbf{Q}$ and assign them to the new set of clusters $\mathcal{P}$.
4) **Reduction step:** The clusters are used to reduce the graph $\mathcal{G}$ and the data $\mathbf{X}$. This boils down to averaging features and grouping edges.

Stopping condition: Nearest neighbor grouping can be performed repeatedly on the reduced versions of the graph $\mathcal{G}$ and the data $\mathbf{X}$ until the desired number k of clusters is reached.

The algorithm is iterated until the desired number of clusters k is reached. At each iteration, a connected components routine extracts them from $\mathbf{Q}$ and returns them as a set of clusters $\mathcal{P}$. In the last iteration of the algorithm, if there are less than k connected components, $\mathbf{Q}$ is pruned of its edges with largest edge values to keep only the $q-k$ shortest edges, so that no less than k components are formed.

The number of iterations is at most $O\{\log(p/k)\}$ as the number of vertices is divided by 2 (at least) at each step; in practice, we never have to go beyond 5 iterations. The cost of computing similarities is linear in n and, as all the operations involved are also linear in the number of vertices p, the total procedure is $O(np)$.

4. [39] explicitly used structured random graphs in their analysis and exclude Erdös-Rényi graphs, *i.e.* random connections, as created by unstructured noise.

5. This corresponds to an element-wise condition, where a similarity weight is assigned only if the edges are connected according to $\mathbf{G}$.

6. This algorithm is implemented in the SciPy package. However, other variants of [42] can be used.

Algorithm 1 Recursive nearest neighbor (ReNA) clustering

Require: Data $\mathbf{X} \in \mathbb{R}^{p \times n}$, sparse matrix $\mathbf{G} \in \mathbb{R}^{p \times p}$ representing the associated connectivity graph structure, nearest-neighbor subgraph extraction function NN, connected components extraction function ConnectComp [41], desired number k of clusters.

Ensure: Clustering of the features $\mathcal{P} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k\}$

$q = p$ {Initializing the number of clusters to p}

$t = 0$

$\mathbf{X}^{(t)} = \mathbf{X}$

$\mathbf{G}^{(t)} = \mathbf{G}$

while $q > k$ **do**

$\mathbf{D}^{(t)}_{i,j} \leftarrow \begin{cases} \|\mathbf{X}^{(t)}_{i,*} - \mathbf{X}^{(t)}_{j,*}\|_2^2 & \text{if } \mathbf{G}^{(t)}_{i,j} \neq 0 \\ \infty & \text{otherwise} \end{cases}, (i,j) \in [q] \times [|\mathcal{P}|]$

{Create a similarity weighted graph.}

$\mathbf{Q} \leftarrow \mathrm{NN}(\mathbf{D}^{(t)})$ {1-nearest neighbor graph.}

$\mathcal{P} \leftarrow \mathrm{ConnectComp}(\mathbf{Q})$,

{Sets of connected components of 1-nearest neighbor graph.}

$\mathbf{U}_{i,j} \leftarrow \begin{cases} 1 & \text{if } i \in \mathcal{C}_j \\ 0 & \text{otherwise} \end{cases}, (i,j) \in [q] \times [|\mathcal{P}|]$

{Assignment matrix}

$\mathbf{X}^{(t+1)} \leftarrow (\mathbf{U}^\mathsf{T}\mathbf{U})^{-1}\mathbf{U}^\mathsf{T}\mathbf{X}^{(t)}$, $\mathbf{X}^{(t+1)} \in \mathbb{R}^{|\mathcal{P}| \times n}$

{Reduced data matrix. Note that the computation boils down to averaging grouped features.}

$\mathbf{G}^{(t+1)} \leftarrow \mathrm{support}(\mathbf{U}^\mathsf{T}\mathbf{G}^{(t)}\mathbf{U})$, $\mathbf{G}^{(t+1)} \in \mathbb{R}^{|\mathcal{P}| \times |\mathcal{P}|}$

{Reduced between-cluster topological model; the non-zero values are then replaced by ones.}

$q = |\mathcal{P}|$ {Update the number of clusters}

$t = t + 1$

end while

return $\mathcal{P}$

Even in presence of noise, the cluster diameter does not grow fast thanks to the small number of iterations, hence there is a denoising behavior from Eq. 11.

Note that the NN are calculated on the non-zero values encoded by $\mathbf{G}$ (structure). Additionally, the connected components are not symmetric (see Fig. 3). Thus, we simply take $\mathbf{Q} = (\mathbf{Q} + \mathbf{Q}^\mathsf{T})$[7] to symmetrize them. We also use sparse matrices to perform all calculations hence we avoid forming $p \times p$ matrices.

Fig. 3 presents one iteration of the nearest neighbor grouping on a regular square lattice. The pseudo-code of ReNA is given in algorithm 1 and an illustration on a 2D brain image in Fig. 4.

4 Experimental Study

In this section, we conduct a series of experiments to assess the quality of the dimensionality reduction scheme and its viability as a preprocessing step for several statistical analyses. Table. 1 gives a summary of the datasets used.

We investigate feature grouping with a variety of clustering algorithms: single-linkage, average-linkage, complete-linkage, Ward, SLIC, and ReNA. We use the Euclidean distance for all algorithms and for all hierarchical clustering methods we use the spatial structure as constraints on the agglomeration steps (as in [36]). We compare them to other fast dimensionality reductions: random projections, random sampling, as well as image downsampling. We measure

7. This corresponds to a logical or operation.

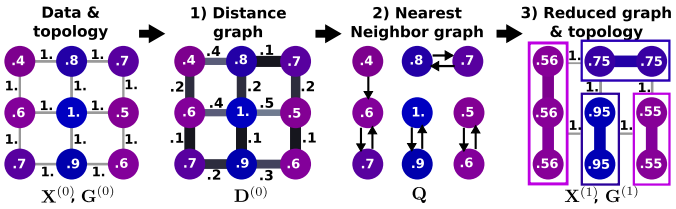


Figure 3. **The nearest neighbor grouping:** The algorithm receives a data matrix $\mathbf{X}$ represented on a regular square lattice $\mathbf{G}$. *left)* The nodes correspond to the feature values and the edges are the encoded topological structure. *1) Graph representation:* We calculate the similarity matrix $\mathbf{D}$. *2) Finding 1-NN:* We proceed by finding the 1-nearest neighbors subgraph $\mathbf{Q}$ according to the similarity measure. *3) Getting the clusters and reduction step:* We extract the connected components of $\mathbf{Q}$ and merge the connected nodes.

Table 1
Summary of the datasets and the tasks performed with them.

Dataset	Description	n	p	Task
Synthetic	Cube	10	$\{8, 16, 64, 128\}^3$	Time complexity (supp mat)
		1 000	240 000	Distortion
Faces [43]	Grayscale face images	2 414	32 256	Recognition of 38 subjects
OASIS [44]	Anatomical brain images	403	140 398	Gender discrimination Age prediction
HCP [2]	Functional brain images	8 294	254 000	Predict 17 cognitive tasks Spatial ICA

their ability to represent the data and characterize their percolation behavior when it is relevant. We use prediction to evaluate their denoising properties. To characterize beyond ℓ_2 approximation, we also consider methods relying on higher moments of the data distribution: ℓ_1 penalization and independent component analysis (ICA). Note that downsampling images with linear interpolation can be seen as using data-independent clusters, all of the same size.

We present results as a function of the fraction of the signal, the ratio between the number k of components and its largest possible value. We have two cases: *i)* for random projections and feature grouping the ratio is $k/p \times 100\%$; *ii)* for random sampling the ratio corresponds to $k/n \times 100\%$.

4.1 Datasets

4.1.1 Synthetic data

We generate a synthetic data set composed of 1 000 3D images with and without noise. Each one is a cube of $p = 50^3$ voxels containing a spatially smooth random signal (FWHM=8 voxels), our signal of interest $\mathbf{S}$. The acquired signal $\mathbf{X}$ is $\mathbf{S}$ contaminated by zero-mean additive Gaussian noise, with a Signal-to-Noise Ratio (SNR) of $2.06dB$.

4.1.2 The extended Yale B face recognition dataset

This dataset was designed to study illumination effects on face recognition [43] and consists of $n = 2\,414$ images of 38 identified individuals under 64 lighting conditions. Each image was converted to grayscale, cropped, and normalized to 192×168 pixels, leaving $p = 32\,256$ features. There are 38 classes for the face recognition task, one per subject.

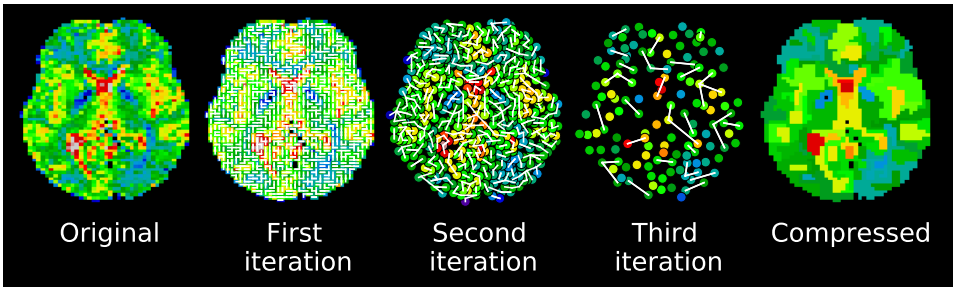
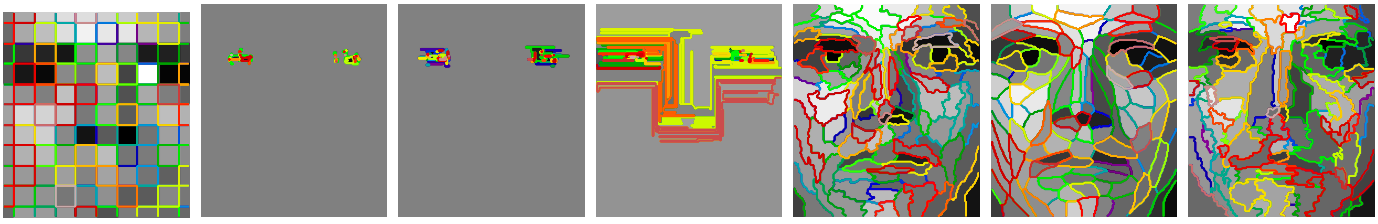


Figure 4. **Illustration of the working principle of the Recursive Nearest Neighbor, ReNA:** The white lines represent the edges of the connectivity graph. The algorithm receives the original sequence of images, considering each feature (i.e. pixel or voxel in the image) as a cluster. From now on, for each iteration, the nearest clusters are merged (i.e. removing edges from the connectivity graph), yielding a reduced graph, until the desired number of clusters is found.



(a) downsampling (b) single-linkage (c) average-linkage (d) complete-linkage (e) Ward (f) SLIC (g) ReNA

Figure 5. **Clusters obtained for the extended Yale B face dataset using various feature grouping schemes:** ($k = 120$). Single, average and complete linkage clustering fail to represent the spatial structure of the data, finding a huge cluster leaving only small islands apart. Downsampling fails to capture the global appearance. In contrast, methods yielding balanced clusters maintain this structure. Colors are random.

4.1.3 The Open Access Series of Imaging Studies (OASIS)

The OASIS dataset[8] [44] consists of anatomical brain images (Voxel Based Morphometry) of 403 subjects. These images were processed with the SPM8 software to obtain modulated grey matter density maps realigned to the Montreal Neurological Institute (MNI) template [45] with a 2mm resolution. Masking with a gray-matter mask yields $p = 140\,398$ voxels and 1 GB of dense data. We perform two prediction tasks: *i)* Gender classification and *ii)* age regression.

4.1.4 Human Connectome Project (HCP)

We use functional Magnetic Resonance Imaging (fMRI) data from the Human Connectome Project (HCP) [2]: 500 participants (13 removed for quality reasons), scanned at rest –typically analyzed via ICA– and during tasks [46] –typically analyzed with linear models. We use the minimally preprocessed data [47], resampled at 2mm resolution.

Task data: We use tasks relating to different cognitive labels on working memory and cognitive control.

Resting-state data: We use the two resting-state sessions from 93 subjects. Each session represents data with $p \approx 220\,000$ and $n = 1\,200$, totaling 200 GB of dense data for all subjects and sessions.

4.2 Technical Aspects

We use scikit-learn for logistic and Ridge regression, fastICA, clustering, and random projections [48]. We rely on scikit-image for SLIC [49], on Nilearn to handle neuroimaging data, and on Scipy [50] to extract graph connected components. Code for ReNA and experiments is available[9].

8. OASIS was supported by grants P50 AG05681, P01 AG03991, R01 AG021910, P50 MH071616, U24 RR021382, R01 MH56584.

9. https://github.com/ahoyosid/ReNA

4.3 Quality assessment experiments

We vary the number of clusters k and evaluate the performance of $\mathbf{\Phi}$ with three measures: *i)* the computation time; *ii)* the signal distortion; *iii)* the size of the largest cluster. As a dimension reduction learned from data may capture noise in addition to signal, we test the learned $\mathbf{\Phi}$ on left-out data, in a cross-validation scheme splitting the data randomly 50 times. Each time, we learn $\mathbf{\Phi}$ on half of the noisy data and apply it to the other half to measure distortion with regards to the non-noisy signal. We vary the number of clusters $k \in [0.01p, p]$. For Nyström approximation, we vary the dimensionality $k \in [0.01n, n]$.

Fig. 5 shows the clusters found by the various algorithms on the faces dataset. Single, average, and complete linkage have percolated, failing to retain the spatial structure of the faces. Downsampling also fails to capture this structure, while Ward, SLIC and ReNA perform well in this task.

Distortion: We want to test whether the reduction $\mathbf{\Phi}\,\mathbf{X}$ of the noisy data is true to the uncorrupted signal $\mathbf{S}$,

$$\|\mathbf{\Phi}\,\mathbf{X}_{*,i} - \mathbf{\Phi}\,\mathbf{X}_{*,j}\|_2 \approx \|\mathbf{S}_{*,i} - \mathbf{S}_{*,j}\|_2,\ \forall (i,j) \in [n]^2. \quad (13)$$

To do so, we split the uncorrupted signal matrix $\mathbf{S}$ and the measured noisy data matrix $\mathbf{X}$ into train and test sets, $(\mathbf{S}^{\text{train}}, \mathbf{S}^{\text{test}})$ and $(\mathbf{X}^{\text{train}}, \mathbf{X}^{\text{test}})$. We learn a matrix $\mathbf{\Phi}$ on the noisy training data, $\mathbf{X}^{\text{train}}$. Then, we compare the pairwise distance between test samples of uncorrupted signal, $\mathbf{S}^{\text{test}}$ to the pairwise distance between corresponding samples of noisy data after dimensionality reduction, $\mathbf{\Phi}\,\mathbf{X}^{\text{test}}$. Finally, we report the relative distortion between these two distances (for a full description see section 4, supplementary materials). We carry out this experiment on two datasets: *i)* synthetic data, and *ii)* brain activation images (motor tasks) from the HCP dataset.

Fig. 6 (*top*) presents the results on the distortion. Note that SLIC stops early in the range of the number of clusters. In synthetic data, the clustering methods based on first-

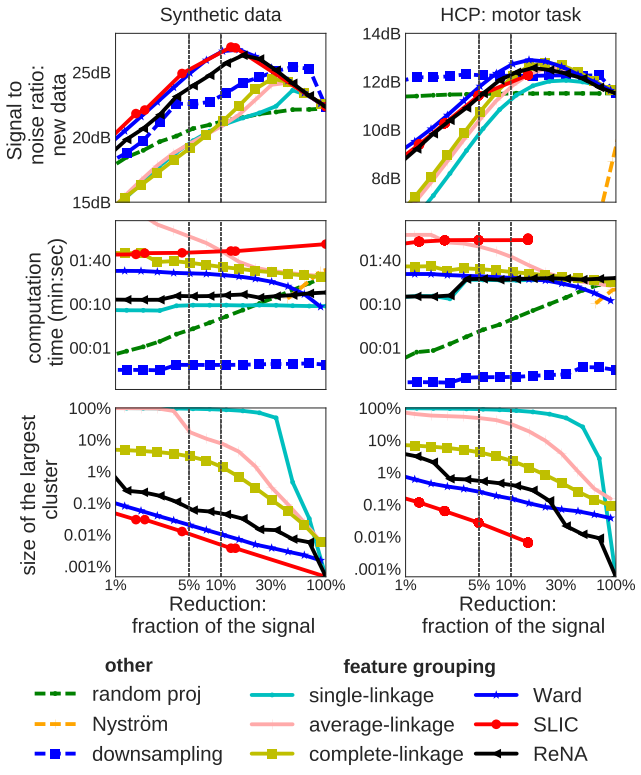


Figure 6. **Assessment of various approximation techniques on synthetic and brain imaging data:** Evaluation of the performance varying the number k of clusters. *(top)* Empirical measure of the distortion of the approximated distance. For a fraction of the signal between 5% and 30% Ward, SLIC and ReNA present a denoising effect, improving the approximation of the distances. In contrast, traditional agglomerative clustering fails to preserve distances in the reduced space. Downsampling displays an intermediate performance. *(center)* Regarding computation time, downsampling and random sampling outperform all the alternatives, followed by random projections and single-linkage. The proposed method is almost as fast as single-linkage. *(bottom)* Percolation behavior, measured via the size of the largest cluster. Ward, SLIC and ReNA are the best avoiding huge clusters. The vertical dashed lines indicate the useful value range for practical applications ($k \in [\lfloor p/20 \rfloor, \lfloor p/10 \rfloor]$).

order linkage criteria (single, average, complete linkage) fail to represent the data accurately. By contrast, SLIC, Ward and ReNA achieve the best representation performance. These methods also show an expected denoising effect for $\lfloor p/20 \rfloor < k < \lfloor p/10 \rfloor$). In such range, the learned approximation matches approximately the smoothing kernel that characterizes the input signal. Downsampling also exhibits a denoising effect, needing more components than the non-percolating methods. For the HCP dataset, the denoising effect is subtle, given that we do not have access to noiseless signals. Downsampling and random projections find a plateau in the relative distortion curve, meaning that the signal has a low entropy that is captured with only few components. In both datasets, dimensionality reduction by random projections and Nyström fail to decrease the noise. This is because they guarantee good approximate distances, and hence represent also the noise.

Computation time: Fig. 6 (*center*) gives computation time for the different methods. Dimension reduction by downsampling is the fastest, as it does not require any training and the computational time lies in the linear interpolation. It is followed by the Nyström approximation. While computation time of Nyström approximation and random projections increases with the reduction fraction of signal, agglomerative clustering approaches become faster, as they require less merges. Random projections are faster than clustering approaches to reduce signals to a size smaller than 30% of their original size. Among the clustering approaches, single-linkage and ReNA are the fastest, as expected. Note that the cost of the clustering methods scales linearly with the number of samples, hence can be reduced by subsampling: using less data to build the feature grouping.

Percolation behavior: As percolation is characterized by the occurrence of a huge cluster when k decreases, we report the size of the largest cluster varying the number k of clusters on Fig. 6 (*bottom*). Among the traditional agglomerative methods, single and average linkage display the worst behavior and quickly percolate. Complete-linkage exhibits a more progressive behavior, with large clusters that grow slowly in the small k regime. On the other hand, Ward and SLIC are most resilient to percolation. Indeed, they are both known to create clusters of balanced size. Finally, ReNA achieves a slightly worse performance, but mostly avoids huge clusters.

4.4 Use in prediction tasks

To evaluate the denoising properties of dimension reduction, we now consider their use in prediction tasks. We use linear estimators as they are standard in high dimensional problems. We consider ℓ_2 and ℓ_1 penalties. For the ℓ_2 case, the operator $\mathbf{\Phi}_{\mathsf{FG}}^{\mathsf{T}}\mathbf{\Phi}_{\mathsf{FG}}$ acts like a kernel. For such estimators, dimension reductions that preserve pairwise distance are thus well theoretically motivated [10].

For each estimation problem, we use the relevant metric (explained variance[10] for regression and accuracy[11] for classification). We measure the performance of the pipeline: dimension reduction + estimator. Results are compared to those obtained without dimension reduction (raw data).

4.4.1 *Spatial approximation on a faces recognition task*

A classic pipeline to tackle face recognition consists of first dimensionality reduction of the data followed by classifier training. Pipelines may include random projections, PCA, downsampling [43], or dictionary learning [51]. Dimension reduction is motivated because varying illumination on a subject with a fixed pose creates a low-dimensional subspace [52].

We follow a study on reduced face representations [43], computing prediction accuracy for various feature-space dimensions $k \in \{30, 56, 120, 504\}$, corresponding to downsampling ratios of $\{1/32, 1/24, 1/16, 1/8\}$. For the classifier, we use an ℓ_2 or ℓ_1 logistic regression with a multi class *one-vs-rest* strategy and set the regularization parameter λ by 10-fold nested cross-validation. We measure prediction error with 50 iterations of cross-validation, randomly splitting half of each subject's images into train and test set.

Fig. 7 reports the prediction accuracies. For high reduction factors, Ward, Nyström, and ReNA perform up to 10%

10. The explained variance is defined as $R^2 = 1 - \frac{\mathrm{Var(model - signal)}}{\mathrm{Var(signal)}}$
11. Accuracy is defined by: $1 - \frac{\text{number of miss-classifications}}{\text{total number of samples}}$

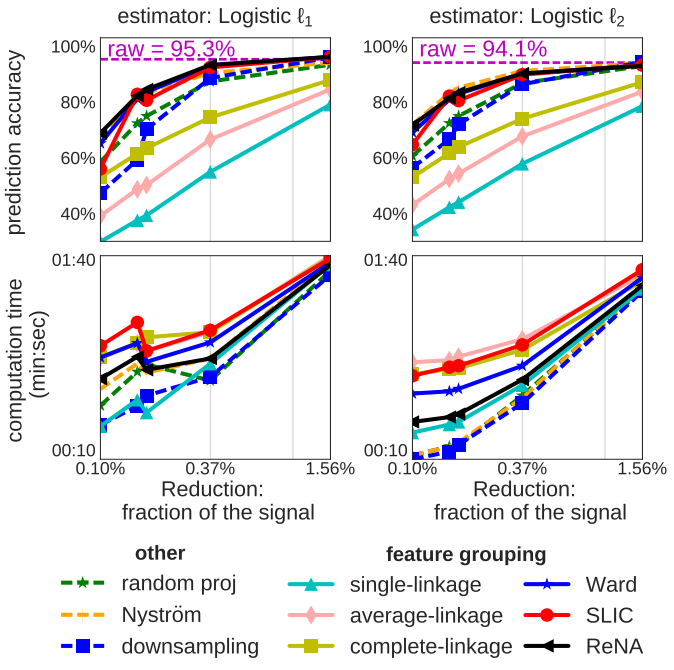


Figure 7. **Face prediction accuracy for various approximation schemes:** Prediction accuracy as function of the feature space dimension obtained for various approximation schemes and classifiers for face recognition on the extended Yale B dataset. The clustering methods finding balanced clusters need less features to have a fair performance, and they obtain significantly higher scores than the percolating methods.

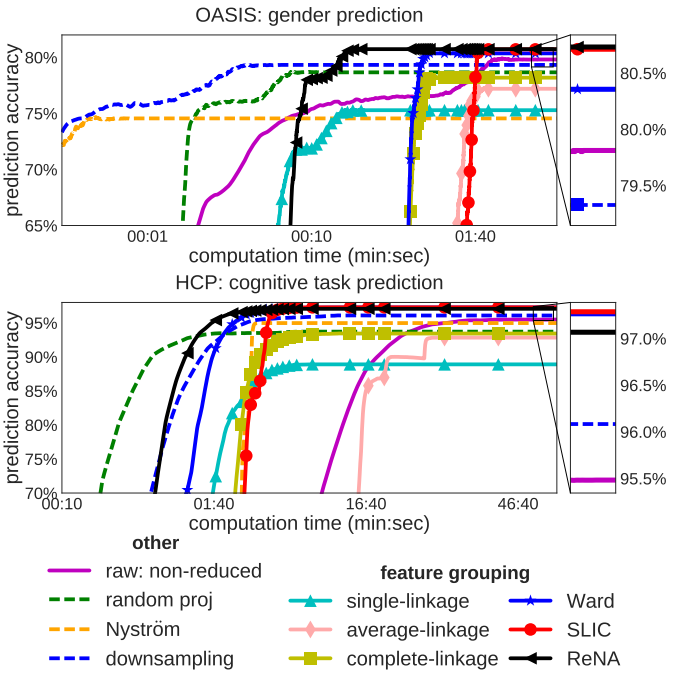


Figure 8. **Computation time taken to reach a solution:** Quality of the fit of a ℓ_2 penalized logistic regression as function of the computation time for a fixed number of clusters. In both datasets, Ward, SLIC and ReNA obtain significantly higher scores than estimation on non-reduced data with less computation time to reach a stable solution. Note that the time displayed does include cluster computation.

better than random projections or downsampling: representations adjusted on the data outperform data-independent reduction operators. For raw data, without reduction, prediction accuracy is around 95.3% and 94.1% for the ℓ_1 and ℓ_2 penalization respectively. Similar performance is obtained after reducing the signal by a factor of 64 with random projections, Nyström, downsampling, Ward, SLIC or ReNA. In contrast, single, average and complete linkage clustering fail to achieve the same performance. This shows the importance of finding balanced clusters.

Regarding computation time, data reduction speeds up the convergence of the logistic regression. Nyström and downsampling are the fastest methods. Random projections, single-linkage, and ReNA follow, all with similar performances. Average, complete linkage, Ward, and SLIC are slightly slower on this dataset.

4.4.2 *Trade-offs: prediction accuracy on a time budget*

Here, we examine the impact of the signal approximation on prediction accuracy and prediction time. We use several datasets: in addition to faces, anatomical and functional brain images. Supervised learning on brain images [53], [54] is a typical setting that faces a rapid increase in dimensionality. Indeed, with progresses in MRI, brain images are becoming bigger, leading to computational bottlenecks. The Human Connectome Project (HCP) is prototypical of these challenges, scanning 1 200 subjects with high-resolution protocols. We consider both anatomical brain images (OASIS dataset, $n = 403$ and $p = 140\,000$) and functional brain images (HCP dataset, $n = 8\,294$ and $p = 250\,000$), with 3 different prediction problems: age and gender prediction from anatomy, and discriminating 17 cognitive tasks from functional imaging.

We use the dimension reduction approaches to speed up predictor training. We are interested in the total computation time needed to learn a model: the cost of computing the compressed representation and of training the classifier. Based on prior experiments, we set $k = \lfloor p/20 \rfloor$ for random projections, downsampling and clustering methods, and $k = \lfloor n/10 \rfloor$ for Nyström, except for the faces dataset, where we use the $k = 504$ for all the methods. For classification, we use a multinomial logistic regression with an ℓ_2 penalty and an l-BFGS solver and for regression we use a ridge.

First, for a better understanding, we show on Fig. 8 convergence of the logistic-regression solver as a function of time on the two brain classification tasks. Time is spent in learning the data reduction and iterations of the solver. Interestingly, for some dimension reduction approaches, the prediction reaches quickly a good accuracy regime, in particular for Ward clustering and ReNA. As with previous experiments, feature clustering with single, average, and complete linkage lead to poor prediction. On the opposite, SLIC, Ward, and ReNA give better prediction that non reduced data, due to the denoising effect of feature clustering.

We then review systematically across datasets the impact of the various data reductions on prediction accuracy, the time taken to compute the reductions, and the total time to convergence (Fig. 9). We find that dimension reduction with clustering algorithms that yield balanced clusters (Ward, SLIC, and ReNA) achieves similar or better accuracy than raw data while bringing drastic time savings. Random projections and the percolating methods give consistently

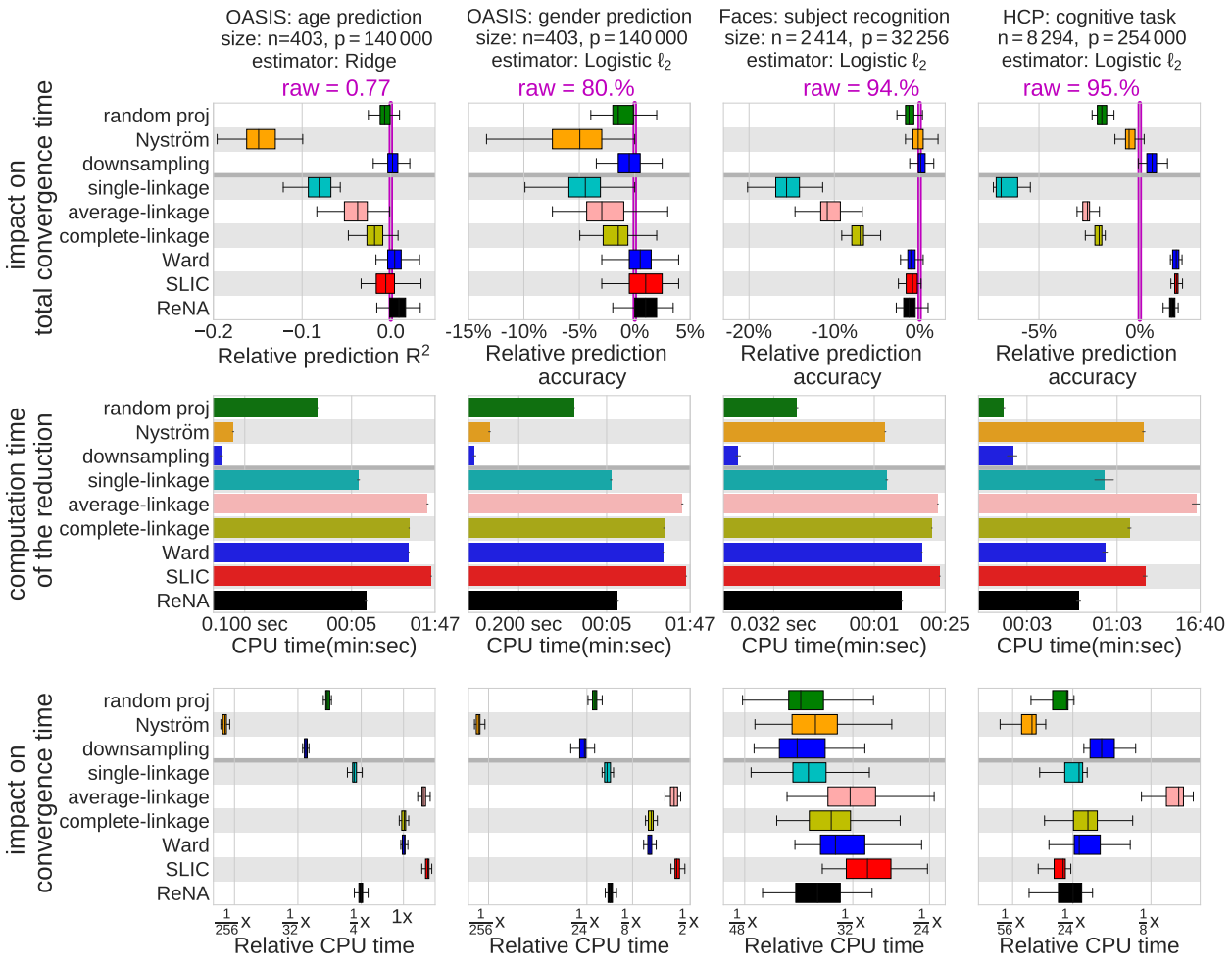


Figure 9. **Impact of reduction methods on prediction for various datasets:** *(Top)* Each bar represents the impact of the corresponding option on the prediction accuracy, relatively to the mean prediction with non-reduced data. Downsampling has the same performance as raw data. On the other hand, random projections, Nyström, single, average and complete linkage algorithms are consistently the worst ones across datasets. Ward, SLIC and ReNA perform at least as good as non-reduced data. *(middle)* Regarding the computation time to find a reduction, single-linkage and ReNA are consistently the best among the clustering algorithms. Random projections perform better than Nyström when the number of samples is large. Downsampling is the fastest across datasets. *(Bottom)* The time to converge for single-linkage and ReNA is almost the same. Average, complete-linkage and Ward are consistently the slowest. SLIC performs well on large datasets. Nyström and random projections are the fastest across datasets. Single-linkage and ReNA are the fastest clustering methods. ReNA strikes a good trade off between time and prediction accuracy.

worse prediction accuracy than raw data. On the OASIS dataset, downsampling, SLIC, and Ward achieve the same prediction accuracy as raw, and perform better than raw on other datasets. Nyström only performs as good as raw data on the faces dataset with an ℓ_2 penalized logistic regression. ReNA has a slightly worse performance than raw only in this dataset, and displays a better performance than raw on the remaining datasets (p-value $< 10^{-4}$). This illustrates the reduction of the spatial noise afforded by non-percolating clustering methods.

4.5 Use in a spatial ICA task

Aside from the ℓ_1-penalized estimator, the data processing steps studied above depend only on pairwise distances between samples. We now investigate dimension reduction before an Independent Component Analysis (ICA), which probes higher moments of the data distribution. ICA is used routinely on resting-state fMRI to separate signal from noise or obtain functional networks [55]. We use 93 subjects of the HCP data, with two rest fMRI sessions, each containing 1200 brain images.

We compare ICA on the raw data and after dimension reduction to 5% of the number of voxels ($k = \lfloor \frac{p}{20} \rfloor$). For Nyström, the dimension is set to 10% of the number n of samples ($k = \lfloor \frac{n}{10} \rfloor$). In each subject, we extract 40 independent components, a standard choice in the literature. We investigate *i)* how similar the components obtained are before and after reduction; *ii)* how similar the components of session 1 and session 2 are with different reduction approaches. This second experiment gives a measure of the variability due to noise. In both cases, we measure components similarity with the absolute value of their correlation, and match them across sessions with the Hungarian algorithm.

Fig. 10 summarizes the use of dimension reduction in ICA of rest fMRI. We find that the 40 components are highly similar before and after data reduction with downsampling and Ward: the average absolute correlation greater than 0.8. SLIC and ReNA have a slightly worse performance, with an average correlation greater than 0.74. On the other hand,

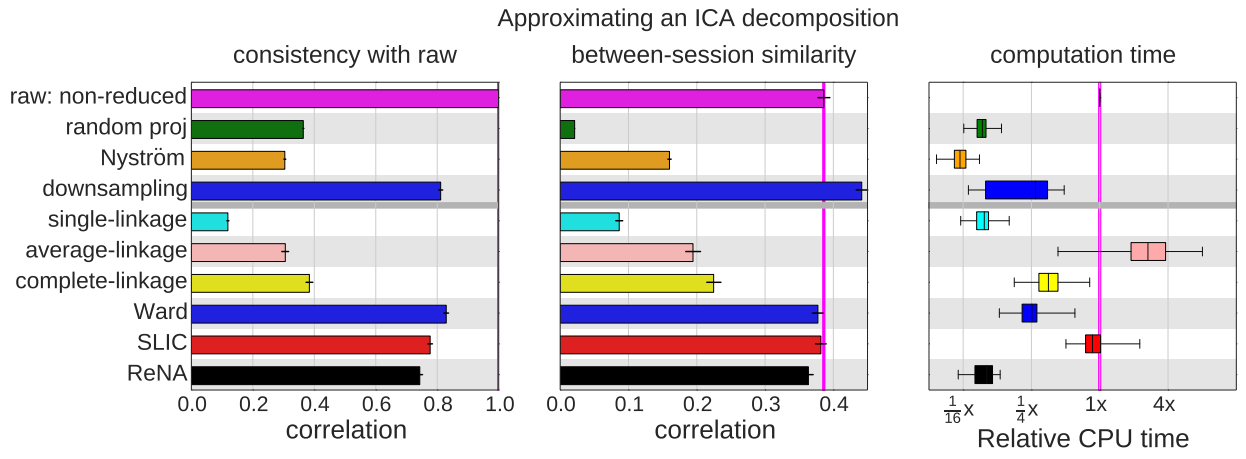


Figure 10. **Spatial ICA reproducibility after dimension reduction:** Reproducibility of 40 spatial independent components on fMRI for 93 subjects, with a fixed reduced dimension (see section 4.5). *(Left)* the similarity of downsampling, Ward, SLIC and ReNA with respect to the non-compressed components is high. *(Middle)* across two sessions, donwsampling yields components more consistent than raw data. Ward, SLIC and ReNA perform as well as raw data, while other approaches fail to do so. *(Right)* regarding computational time, ReNA outperforms downsampling, Ward and SLIC, and performs as well as single-linkage and random projections. It is 16 time faster than working on raw data.

single-linkage, average-linkage, complete-linkage, Nystöm, and random projections do not recover the components (average correlation < 0.4). As expected, the components between sessions obtained by non-percolating clustering (Ward, SLIC, and ReNA) are similar to the original ones. Donwsampling improves the similarity with respect to raw: the estimation problem is simpler and less noisy. On the opposite, single, average, and complete linkage degrade the similarity: they lose signal due to the large cluster created by percolation. Random projections and Nyström perform poorly. Indeed, they average data across the images, destroying the high-order moments of the data by creating signals more Gaussian than the originals. As a consequence, ICA cannot recover the sources derived from the original data. By contrast, the non-percolating clustering algorithms extract local averages of the data, that preserve its non-Gaussianity, as it has a spatial structure. Hence the spatial ICA is successful even though it has access to less samples. Finally, dimensionality reduction using ReNA speeds up the total analysis by a factor of 15.

5 Summary and Discussion

Fast dimension reduction is a crucial tool to tackle the rapid growth in datasets size, sample-wise and feature-wise. In particular, grouping features is natural when there is an underlying regularity in the observed signal, such as spatial structure in images or a more general neighborhood structure connecting features. We studied here a data-driven approach to perform feature grouping, where groups are first learned from a fraction of the data using a clustering algorithm, then used to build a compressed representation for further analysis.

We showed that feature grouping can preserve well the pairwise Euclidean distances between images. This property makes it well suited for ℓ_2-based algorithms, like shift-invariant kernel-based methods, or to approximate queries in information-retrieval settings. We also clarified under which hypotheses this scheme leads to a beneficial bias/variance compromise: as clustering adapts to the data, it reaches more optimal regimes than simple downsampling-based compression.

Additionally, we proposed a linear-time graph-structured clustering algorithm, ReNA, that is efficient with many clusters. This algorithm iteratively performs 1-nearest neighbor grouping, reduces the graph at each iteration, then averages the input features and repeats the process until it reaches the desired number of clusters. We have shown empirically that it is fast, does not percolate, and provides excellent performance in feature grouping for dimension reduction of structured data.

Our experiments have shown that on moderate-to-large datasets, non-percolating feature-grouping schemes (i.e. Ward, SLIC, and ReNA) most often outperform state-of-the-art fast data-approximation approaches for machine learning, namely random projection and random sampling. Using these methods in a predictive pipeline increases the quality of statistical estimations: they yield more accurate predictions than with all features. This indicates that feature grouping leads to a good approximation of the data, capturing structure and reducing noise. This denoising is due to the smoothness of the signal of interest: unlike the noise, the signal displays structure captured by feature grouping.

A key benefit of the ReNA clustering algorithm is that it is very fast while avoiding percolation. As a result, it gives impressive speed-ups for real-world multivariate statistical problems: often more than one order of magnitude. Note that the computational cost of ReNA is linear in the number of samples, hence additional computation gains can be obtained by sub-sampling its training data, as in Nyström approaches. In this work, we did not investigate the optimal choice of the number k of clusters, because we do not view compressed representations as a meaningful model per se, but as an approximation to reduce data dimension without discarding too much information. The range $k \in \left[\left\lfloor \frac{p}{20} \right\rfloor, \left\lfloor \frac{p}{10} \right\rfloor\right]$ is a useful regime as it gives a good trade-off between computational efficiency and data fidelity. In our experiments, $k = \left\lfloor \frac{p}{20} \right\rfloor$ gave enough data fidelity

for statistical analysis to perform at least as well as on raw data. In this regime, Ward clustering gives slightly better approximations of the original data, however it is slower, often by several orders of magnitude, hence it is impractical.

We have shown that feature grouping is useful beyond ℓ_2-distance-based methods: it also gives good performance on estimators relying on higher order moments (e.g. ICA) or sparsity (ℓ_1-based regression or classification)[12]. As future work, it would be interesting to investigate the use of ReNA-based feature grouping in expensive sparse algorithms, for instance with sparse dictionary learning, where feature sub-sampling can give large speed ups [56]. Similarly, the combination of clustering, randomization, and sparsity has also been shown to be an effective regularization for some ill-posed inverse problems [25], [57]. This is all the more important that computation cost is a major roadblock to the adoption of such estimators.

An important aspect of feature grouping compared to other fast dimension reductions, such as random projections, is that the features of the reduced representation make sense for the application. Consequently, the dimension reduction step can be inverted, and any statistical analysis performed after reduction can be reported with regard to the original signal.

Given that ReNA clustering is very fast, the proposed featuring-grouping is an extremely promising avenue to speed up any statistical analysis of large datasets where the information is in the large-scale structure of the signal. Such approach is crucial for domains where the resolution of the sensors is rapidly increasing, in medical or biological imaging, genomics, spectroscopy, or geospatial data.

Acknowledgments

Data were provided in part by the Human Connectome Project, WU-Minn Consortium (Principal Investigators: D. Van Essen and K. Ugurbil; 1U54MH091657) funded by the 16 NIH Institutes and Centers that support the NIH Blueprint for Neuroscience Research; and by the McDonnell Center for Systems Neuroscience at Washington University. Research leading to these results has received funding from the European Union's Horizon 2020 Framework Programme for Research and Innovation under Grant Agreement No 720270 (Human Brain Project SGA1).

References

[1] F. Amat, E. W. Myers, and P. J. Keller, "Fast and robust optical flow for time-lapse microscopy using super-voxels," *Bioinformatics*, vol. 29, pp. 373–80, 2013.

[2] D. V. Essen, K. Ugurbil, E. Auerbach, D. Barch, T. Behrens, R. Bucholz, A. Chang, L. Chen, M. Corbetta, S. Curtiss, S. D. Penna, D. Feinberg, M. Glasser, N. Harel, A. Heath, L. Larson-Prior, D. Marcus, G. Michalareas, S. Moeller, R. Oostenveld, S. Petersen, F. Prior, B. Schlaggar, S. Smith, A. Snyder, J. Xu, and E. Yacoub, "The human connectome project: A data acquisition perspective," *NeuroImage*, vol. 62, pp. 2222–2231, 2012.

[3] R. Overbeek, N. Larsen, G. D. Pusch, M. D'Souza, E. S. Jr, N. Kyrpides, M. Fonstein, N. Maltsev, and E. Selkov, "WIT: integrated system for high-throughput genome sequence analysis and metabolic reconstruction," *Nucleic Acids Research*, vol. 28, pp. 123–125, 2000.

12. On the faces data for the ℓ_1 estimator, ReNA performs as well as raw data.

[4] J. R. Cole, Q. Wang, J. A. Fish, B. Chai, D. M. McGarrell, Y. Sun, C. T. Brown, A. Porras-Alfaro, C. R. Kuske, and J. M. Tiedje, "Ribosomal database project: data and tools for high throughput rrna analysis," *Nucleic Acids Research*, 2013.

[5] T. Addair, D. Dodge, W. Walter, and S. Ruppert, "Large-scale seismic signal analysis with hadoop," *Computers & Geosciences*, vol. 66, p. 145, 2014.

[6] H. Jégou, M. Douze, and C. Schmid, "Improving bag-of-features for large scale image search," *Int. J. Comput. Vision*, vol. 87, pp. 316–336, 2010.

[7] A. Hoyos-Idrobo, Y. Schwartz, G. Varoquaux, and B. Thirion, "Improving sparse recovery on structured images with bagged clustering," *IEEE PRNI*, 2015.

[8] K. Chakrabarti, E. Keogh, S. Mehrotra, and M. Pazzani, "Locally adaptive dimensionality reduction for indexing large time series databases," *ACM Trans. Database Syst.*, vol. 27, pp. 188–228, 2002.

[9] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Dimensionality reduction for fast similarity search in large time series databases," *Knowledge and information Systems*, vol. 3, p. 263, 2001.

[10] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *NIPS*, 2007.

[11] N. Ailon and B. Chazelle, "The fast Johnson-Lindenstrauss transform and approximate nearest neighbors," *SIAM J. Comput.*, vol. 39, pp. 302–322, 2009.

[12] N. Halko, P. G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM Rev.*, vol. 53, pp. 217–288, 2011.

[13] A. Gittens and M. W. Mahoney, "Revisiting the Nyström method for improved large-scale machine learning." in *ICML*, 2013, pp. 567–575.

[14] M. W. Mahoney and P. Drineas, "CUR matrix decompositions for improved data analysis," *PNAS*, vol. 106, pp. 697–702, 2009.

[15] W. Johnson and J. Lindenstrauss, "Extensions of Lipschitz mappings into a Hilbert space," in *Conference in modern analysis and probability*, ser. Contemporary Mathematics. American Mathematical Society, 1984, vol. 26, pp. 189–206.

[16] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing SAX: a novel symbolic representation of time series," *Data Mining and knowledge discovery*, pp. 107–144, 2007.

[17] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *Signal Processing Magazine, IEEE*, vol. 30, pp. 83–98, 2013.

[18] D. Stauffer and A. Aharony, *Introduction to Percolation Theory*. Oxford University Press, New York, 1971.

[19] C. Hedge, A. C. Sankaranarayanan, W. Yin, and R. G. Baraniuk, "NuMax: a convex approach for learning near-isometric linear embeddings," *IEEE Trans. Signal Process.*, vol. 63, p. 6109, 2015.

[20] Y. Lu, P. S. Dhillon, D. P. Foster, and L. H. Ungar, "Faster ridge regression via the subsampled randomized hadamard transform." in *NIPS*, 2013, pp. 369–377.

[21] D. Achlioptas, "Database-friendly random projections: Johnson-Lindenstrauss with binary coins," *Journal of Computer and System Sciences*, vol. 66, pp. 671–687, 2003.

[22] J. A. Tropp, "Improved analysis of the subsampled randomized hadamard transform," *Advances in Adaptive Data Analysis*, vol. 3, pp. 115–126, 2011.

[23] C. Williams and M. Seeger, "Using the Nyström method to speed up kernel machines," in *NIPS*, 2001, pp. 682–688.

[24] A. Rudi, R. Camoriano, and L. Rosasco, "Less is more: Nyström computational regularization," in *NIPS*, 2015, pp. 1648–1656.

[25] P. Bühlmann, P. Rütimann, S. van de Geer, and C.-H. Zhang, "Correlated variables in regression: clustering and sparse estimation," *Journal of Statistical Planning and Inference*, vol. 143, p. 1835, 2013.

[26] L. Le Magoarou and R. Gribonval, "Flexible multilayer sparse approximations of matrices and applications," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, pp. 688–700, 2016.

[27] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, p. 2274, 2012.

[28] D. Arthur and S. Vassilvitskii, "How slow is the k-means method?" in *Annual Symposium on Computational Geometry*, 2006, p. 144.

[29] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, 2nd ed., ser. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2009.
[30] J. H. Ward, "Hierarchical grouping to optimize an objective function," *J. Am. Stat. Ass.*, vol. 58, p. 236, 1963.
[31] D. Müllner, "Modern hierarchical, agglomerative clustering algorithms," *ArXiv e-prints*, 2011.
[32] B. Mirkin, "Mathematical classification and clustering: From how to what and why," in *Classification, data analysis, and data highways*. Springer, 1998, pp. 172–181.
[33] A. Rosenfeld and J. L. Pfaltz, "Sequential operations in digital picture processing," *Journal of the ACM (JACM)*, vol. 13, p. 471, 1966.
[34] M. Penrose, "Single linkage clustering and continuum percolation," *Journal of Multivariate Analysis*, vol. 53, pp. 94 – 109, 1995.
[35] B. Thirion, G. Varoquaux, E. Dohmatob, and J.-B. Poline, "Which fMRI clustering gives good brain parcellations?" *Frontiers in Neuroscience*, vol. 8, 2014.
[36] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *Int. J. Comput. Vision*, vol. 59, p. 167, 2004.
[37] D. Eppstein, M. S. Paterson, and F. Yao, "On nearest-neighbor graphs". discrete and computational geometry," vol. 17, pp. 263–282, 1997.
[38] M. Ester, H. peter Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise." AAAI Press, 1996, pp. 226–231.
[39] M. Maier, M. Hein, and U. von Luxburg, "Optimal construction of k-nearest-neighbor graphs for identifying noisy clusters," *Theoretical Computer Science*, vol. 410, no. 19, pp. 1749 – 1764, 2009, algorithmic Learning Theory.
[40] S.-H. Teng and F. F. Yao, "k-nearest-neighbor clustering and percolation theory," *Algorithmica*, vol. 49, pp. 192–211, 2007.
[41] D. J. Pearce, "An improved algorithm for finding the strongly connected components of a directed graph," Tech. Rep., 2005.
[42] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM journal on computing*, vol. 1, pp. 146–160, 1972.
[43] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, pp. 210–227, 2009.
[44] D. S. Marcus, T. H. Wang, J. Parker, J. G. Csernansky, J. C. Morris, and R. L. Buckner, "Open access series of imaging studies (OASIS): cross-sectional MRI data in young, middle aged, nondemented, and demented older adults." *J Cogn Neurosci*, vol. 19, pp. 1498–1507, 2007.
[45] W. Chau and A. R. McIntosh, "The talairach coordinate of a point in the MNI space: how to interpret it," *NeuroImage*, vol. 25, pp. 408–416, 2005.
[46] D. M. Barch, G. C. Burgess, M. P. Harms, S. E. Petersen, B. L. Schlaggar, M. Corbetta, M. F. Glasser, S. Curtiss, S. Dixit, C. Feldt, D. Nolan, E. Bryant, T. Hartley, O. Footer, J. M. Bjork, R. Poldrack, S. Smith, H. Johansen-Berg, A. Z. Snyder, D. C. V. Essen, and W. U.-M. H. Consortium, "Function in the human connectome: task-fMRI and individual differences in behavior." *Neuroimage*, vol. 80, pp. 169–189, 2013.
[47] M. F. Glasser, S. N. Sotiropoulos, J. A. Wilson, T. S. Coalson, B. Fischl, J. L. Andersson, J. Xu, S. Jbabdi, M. Webster, and J. R. Polimeni, "The minimal preprocessing pipelines for the human connectome project," *Neuroimage*, vol. 80, pp. 105–124, 2013.
[48] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, p. 2825, 2011.
[49] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, and the scikit-image contributors, "Scikit-image: image processing in Python," *PeerJ*, vol. 2, p. e453, 2014.
[50] E. Jones, T. Oliphant, and P. Peterson, "SciPy: open source scientific tools for Python," 2014.
[51] Z. J. Xiang, H. Xu, and P. J. Ramadge, "Learning sparse representations of high dimensional data on large scale dictionaries," in *NIPS*, 2011, pp. 900–908.
[52] R. Basri and D. Jacobs, "Lambertian reflection and linear subspaces," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, p. 218, 2003.
[53] T. M. Mitchell, R. Hutchinson, R. S. Niculescu, F. Pereira, X. Wang, M. Just, and S. Newman, "Learning to decode cognitive states from brain images," *Machine Learning*, vol. 57, p. 145, 2004.
[54] Y. Schwartz, B. Thirion, and G. Varoquaux, "Mapping cognitive ontologies to and from the brain," in *NIPS*, 2013.
[55] S. M. Smith, C. F. Beckmann, J. Andersson, E. J. Auerbach, J. Bijsterbosch, G. Douaud, E. Duff, D. A. Feinberg, L. Griffanti, M. P. Harms, M. Kelly, T. Laumann, K. L. Miller, S. Moeller, S. Petersen, J. Power, G. Salimi-Khorshidi, A. Z. Snyder, A. T. Vu, M. W. Woolrich, J. Xu, E. Yacoub, K. Uğurbil, D. C. V. Essen, and M. F. Glasser, "Resting-state fMRI in the human connectome project," *NeuroImage*, vol. 80, pp. 144–168, 2013.
[56] A. Mensch, J. Mairal, B. Thirion, and G. Varoquaux, "Dictionary learning for massive matrix factorization," *ICML*, 2016.
[57] G. Varoquaux, A. Gramfort, and B. Thirion, "Small-sample brain mapping: sparse recovery on spatially correlated designs with randomization and clustering," in *ICML*, 2012, p. 1375.

Andrés Hoyos Idrobo received his is B.S degree in Electronic and Biomedical engineering from the University Autónoma de occidente, Colombia, and his M.S.c degree in applied mathematics from École Normale Supérieure de Cachan, France. He has a Ph.D computer science from University Paris-Saclay. His research focus on faster methods for brain activity decoding, in particular, randomized algorithms.

Gaël Varoquaux is a computer-science researcher at INRIA. He develops statistical learning for functional neuroimaging, applied to cognitive brain mapping and brain pathologies. In addition, he is invested in data-science software, as project lead for scikit-learn, joblib, and nilearn. He has a PhD in quantum physics and is a graduate from École Normale Supérieure, Paris.

Jonas KAHN is a mathematician with interests ranging from fundamental subjects such as operator algebras or random geometric spaces to statistics and applications. On the latter side, he has worked in particular on quantum statistics, and compressed sensing with physical constraints. He has two PhD in mathematics, from Orsay and Leiden, and has graduated from École Normale Supérieure.

Bertrand Thirion is the principal investigator of the Parietal team (Inria-CEA) within the main French Neuroimaging center, Neurospin. He is deputy scientific leader of the Inria Saclay research center. His main research interests address the design of machine learning and statistical analysis techniques for neuroimaging, with applications related to the study of human vision, brain disease diagnosis, population imaging and cognitive brain mapping.