Feature Grouping as a Stochastic Regularizer for High-Dimensional Structured Data

Sergül Aydöre [1] **Bertrand Thirion** [2] **Gaël Varoquaux** [2]

Abstract

In many applications where collecting data is expensive, for example neuroscience or medical imaging, the sample size is typically small compared to the feature dimension. These datasets call for intelligent regularization that exploits known structure, such as correlations between the features arising from the measurement device. However, existing structured regularizers need specially crafted solvers, which are difficult to apply to complex models. We propose a new regularizer specifically designed to leverage structure in the data in a way that can be applied efficiently to complex models. Our approach relies on feature grouping, using a fast clustering algorithm inside a stochastic gradient descent loop: given a family of feature groupings that capture feature covariations, we randomly select these groups at each iteration. Experiments on two real-world datasets demonstrate that the proposed approach produces models that generalize better than those trained with conventional regularizers, and also improves convergence speed, and has a linear computational cost.

1. Introduction

Fitting complex machine learning (ML) models has lead to impressive gains in accuracy in various fields, such as computer vision, speech processing, and natural language processing (LeCun et al., 2015a; Mnih et al., 2013). Yet, the success of complex models has not carried over to high-dimensional small-sample data such as full-brain images, despite clear potential (Plis et al., 2014; Suk et al., 2016). Indeed, complex models are prone to *overfitting* in settings such as those encountered in neuroimaging:

1. Large feature dimension: Neuroimaging data are very high-dimensional, due to the progress in image resolution. For example, functional Magnetic Resonance Images (fMRIs) [1] are represented by 4D-arrays of 3D images over time. The total dimensionality is in the order of 10^7. This leads to the phenomenon known as the *curse of dimensionality*, and is often an obstacle so the success of ML.
2. Noise in the data: Neuroimaging data contain a significant amount of physiological, respiratory, and mechanical artifacts unrelated to the effect of interest. Removal of this noise is a difficult task. Ideally, we need an ML model that is robust against noise.
3. Small sample size: Neuroimaging data typically have small sample sizes due to the logistics and cost of data acquisition, as well as the effort required to recruit subjects. It takes several hours to collect data from a single individual. Therefore, the number of examples in neuroimaging data is usually in the order of hundreds, as opposed to other ML applications, such as computer vision, in which modern data sets comprise at least hundreds of thousands samples.

These challenges are not limited to neuroimaging applications. They are common in medical imaging, genomics, chemistry, and financial applications (Fan & Li, 2006; Consortium et al., 2015). *Regularization* is crucial for the success of ML in such settings. The optimal regularization strategy for a given dataset should leverage the known structure of the data. Yet, classic approaches to structured regularization (Bach et al., 2012) entail high computational cost and are not-well suited for fitting complex models with stochastic gradient descent. Here, we introduce a *structured regularization* strategy integrated in a *stochastic gradient descent* (SGD) loop to tackle the challenges described above.

1.1. Related works: strategies to tackle overfit

A long-standing body of work tackle overfit in ML models. Here, we briefly mention approaches that are most closely related to the present effort. Conventional approaches to

[1]Stevens Institute of Technology, New Jersey, USA [2]Inria Saclay, Palaiseau, France. Correspondence to: Sergul Aydore <sergulaydore@gmail.com>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

[1]fMRI is a noninvasive neuroimaging modality that measures brain activity during cognitive tasks in humans.

mitigate overfitting include penalizing model weights, seeking a reduced-dimensionality parametrization, or sharing weights between related inputs or outputs.

Regularization with ℓ_1 or ℓ_2 penalties (Tibshirani, 1996) reduces overfitting by biasing weights to avoid large values due to chance. In high-dimensional data, features often display groups of highly correlated or irrelevant features (Bühlmann et al., 2013). Structured penalties (Bach et al., 2012) leverage a priori hypotheses on these groups, fostering sparsity accordingly. These approaches are based on the group lasso (Yuan & Lin, 2006) which generalizes ℓ_1 regularization to groups of features. Zhao et al. (2009) similarly generalize ℓ_2 regularization to capture feature groups. A drawback of these formulation is that they require groups of features to be manually identified. Using the overlapping group lasso (Jacob et al., 2009) enables more systematic definitions of groups (Bach et al., 2012). However, as the dimension grows, the number of overlapping groups gives rise to prohibitive computational costs. In addition, these approaches are limited to convex models.

Feature grouping by actually merging the features into a single variable gives faster algorithms, though these are not formulated as a single optimization and rely on heuristics (García-Torres et al., 2016). Using a clustering algorithm to group features is a long-standing dimensionality reduction technique used for ML on high-dimensional data (McCallum et al., 2000; Thalamuthu et al., 2006; Xu & Wunsch, 2005). Combining it with model ensembling gives more robustness to the feature grouping (Varoquaux et al., 2012).

In general, a good dimensionality reduction can limit overfit and improve prediction of a model by reducing its input dimensionality, and thus the number of model parameters. Using random matrices to project data onto a lower dimensional space can capture the important properties of the data (Bingham & Mannila, 2001; Achlioptas, 2003) and thus give very computationally efficient regularizations (Durrant & Kabán, 2013; Alaoui & Mahoney, 2015; Cannings & Samworth, 2017).

Stochastic regularizations also exploit randomness for efficient approaches to prevent overfit. The prototypical example in neural networks is Dropout (Srivastava et al., 2014). Dropout modifies the network structure at each update within an SGD loop: it removes units randomly from the network during training and uses an approximate averaging procedure across these "thinned" networks during testing. Integrating random perturbations within SGD gives a computationally cheap form of ensembling (Bachman et al., 2014). Dropout at the input layer can be viewed as data augmentation with random projections (Bouthillier et al., 2015; Vinh et al., 2016).

Another approach to tackle overfitting is by crafting models with suitable inductive bias, for instance by imposing shared weights to capture invariances of the data. Indeed, in a linear model or a fully-connected layer of a neural network, the number of model weights increases with the number of inputs and the number of outputs. Convolutional neural networks (CNNs) circumvent this difficulty with weight sharing. Rather than fitting one parameter per input pixel, CNNs re-use the same parameters by sliding a filter across the input image. They typically use pooling layers that introduce translation invariance and improve generalization (Hinton et al., 2012). Indeed, CNNs are very successful on natural images because a cat should be modeled as the same object whether it is shifted to the left or to the right. Such invariances also hold for text processing (Kalchbrenner et al., 2014), but not for brain activation images. They display meaningful structure that is specific to given features, *i.e.* brain locations. Non-translation-invariant problems require a departure from CNNs even in computer vision, *eg* with pixel-specific filters (Ren et al., 2015).

1.2. Proposed approach

In this study, we use feature grouping to develop a computationally efficient stochastic regularization approach for data with a general dependency structure across the features. Our algorithm relies on a bank of *feature grouping* matrices to group the features for training. These feature grouping matrices are adapted to the data, but they can be pre-computed outside of the optimization loop for computational efficiency. Optimization is performed by a stochastic gradient descent (SGD), sampling a matrix from the bank during forward propagation to project the features into a reduced representation. The gradient is computed in this low-dimensional space. In order to update the weights during back propagation, we project the gradient back to the original feature dimension. This procedure results in weights expressed on the original feature dimension (brain voxels for neuroimaging; pixels for image processing), that can be used at test time. As the projection matrices are sparse by design, projection is fast and adds only a small computational cost. On the other hand, gradients can be computed more efficiently in the lower-dimensional space. When applied to neural networks, our algorithm is suitable for the input layer only, because it relies on pre-computed projection matrices. These matrices depend on the values of the features, which do not change during training, unlike inputs to intermediate layers. Standard regularization techniques should be applied to the subsequent layers. We target high-dimensional problems: the input layer typically has more parameters than intermediate layers, and therefore calls for dedicated regularization. Figure 1 illustrates our approach for a neural network with a single hidden layer.

The feature grouping approach we employ is a linear-time agglomerative clustering scheme, Recursive Nearest

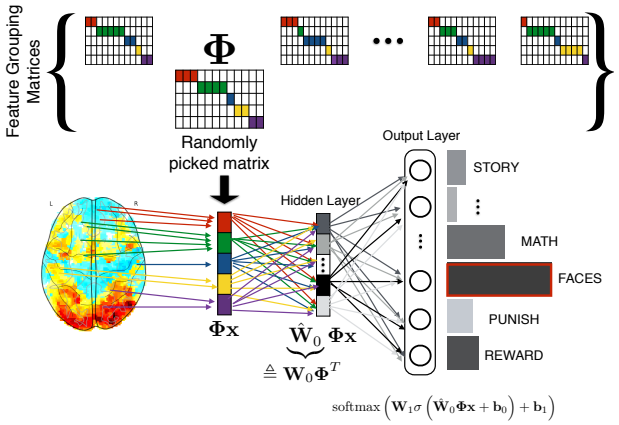


Figure 1. **Illustration of the proposed approach**: Forward propagation of a neural network with a single hidden layer using *feature grouping* during training. The parameters of the neural network to be estimated are $\mathbf{W}_0, \mathbf{b}_0, \mathbf{W}_1, \mathbf{b}_1$. A bank of feature grouping matrices are pre-generated where each matrix is calculated from a sub-sample of the training test. At each SGD iteration, a feature grouping matrix is sampled from the bank of pre-generated matrices. The gradient is then computed with respect to $\hat{\mathbf{W}}_0$ to update $\mathbf{W}_0$ in backpropagation.

Agglomeration (ReNA) proposed by Hoyos-Idrobo et al. (2019). ReNA is similar to the simple linear iterative clustering (SLIC) algorithm (Achanta et al., 2012) used to produce super-pixels in computer vision applications. The advantages of using a fast averaging procedure are two-fold: (i) it has a denoising effect on structured signals; and (ii) it reduces the dimension of signals with computation time linear in the feature dimension.

In the following sections, we provide details of the computational complexity of our approach. We also provide theoretical implications for generalized linear models. We demonstrate the success of our approach in noisy and small-sample settings by applying it to fully-connected multi-layer perceptrons (MLPs) and logistic regressions on the Olivetti faces dataset (HOPPER, 1992), and a publicly available task fMRI data set from the Human Connectome Project (Van Essen et al., 2013). In both cases, our approach outperforms ℓ_2 regularization and dropout applied to the same models, as well as CNNs with dropout. Note that it cannot be combined with CNNs as the structured projection removes the redundant topography which convolutions exploit. Experimental results demonstrate that feature grouping outperforms other methods by the greatest margin when the data size is limited and when the data are contaminated with noise.

2. Model

We consider supervised-learning settings. Let $\mathbf{x} \in \mathbb{R}^p$ a feature vector with $y \in \mathbb{R}$ the corresponding target. The model is a function $f : \mathbb{R}^p \to \mathbb{R}$ with parameters $\boldsymbol{\Theta}$. These parameters are estimated by minimizing the empirical risk over training samples $(\mathbf{x}_i, y_i)$ for $i \in \{1, \cdots, n\}$ such that:

$$\hat{\boldsymbol{\Theta}} = \arg\min_{\boldsymbol{\Theta}} \frac{1}{n} \sum_{i=1}^{n} L\left(f(\mathbf{x}_i; \boldsymbol{\Theta}), y_i\right) \tag{1}$$

where L is the cost per sample. For neural networks with an MLP architecture, the parameter set is $\boldsymbol{\Theta} = \{\mathbf{W}_0, \mathbf{b}_0, \mathbf{W}_1, \mathbf{b}_1, \cdots, \mathbf{W}_H, \mathbf{b}_H\}$ where H denotes the number of hidden layers, $\mathbf{W}_i$ represents the weights and $\mathbf{b}_i$ represents the bias at the i-th layer.

2.1. Dimensionality reduction by feature grouping

We assume that $\mathbf{x}$ represents high-dimensional data with a strong spatial structure as with fMRI data (where $p \sim 10^5 - 10^6$). Reducing the dimensionality of these signals reduces memory requirements and speeds up training steps. This reduced representation helps training if the signal present in $\mathbf{x}$ is preserved. This can be achieved by capturing the signal structure in the dimensionality reduction. Structure-aware dimensionality reduction is indeed known to be useful for neuroimaging data (Mwangi et al., 2014).

We use a data-driven feature averaging approach, ReNA. The features are clustered, and their values are replaced with a single value for each cluster. Let $\boldsymbol{\Phi} \in \mathbb{R}^{k \times p}$ be the dimensionality reduction matrix that projects the data to a lower-dimensional space, with $k \ll p$. The clusters are a partition of the features $\mathcal{P} = \{\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_k\}$, where $\mathcal{C}_q$ is the set of indices that belong to cluster q and $\mathcal{C}_q \cap \mathcal{C}_l = \emptyset$ for $q \neq l$. Approximation on the q-th cluster can be written as: $(\boldsymbol{\Phi}\mathbf{x})_q = \alpha_q \sum_{j \in \mathcal{C}_q} \mathbf{x}_j$, where $\alpha_q = 1/\sqrt{\text{card}(\mathcal{C}_q)}$ is a constant for cluster q chosen to make $\boldsymbol{\Phi}$'s rows orthogonal. $\boldsymbol{\Phi}\mathbf{x} \in \mathbb{R}^k$ is the projected, or reduced, version of $\mathbf{x}$ and $\boldsymbol{\Phi}^T\boldsymbol{\Phi}\mathbf{x}$ is a piecewise constant approximation of $\mathbf{x}$.

We use the ReNA clustering algorithm to obtain the projection matrices $\boldsymbol{\Phi}$. ReNA is a graph-constrained clustering: when the graph represents the dependencies between the features of the signal, feature grouping with ReNA has been shown to have a denoising effect which improves subsequent analysis (Hoyos-Idrobo et al., 2019). The algorithm starts with p clusters, one per feature. Clusters are then recursively merged until the desired number of clusters remain. Merging is achieved by a greedy graph cutting algorithm. For data on a grid, as with image data, the initial graph connects pixels or voxels to their neighbors, with edge weights determined by the data. A new graph is constructed to express the connectivity after merging features, and the process is repeated. Though we use ReNA, our framework can employ any clustering algorithm. The benefits of ReNA are that it is a fast structured clustering algorithm that leads to good signal approximations.

2.2. Stochastic Regularizer with Feature Grouping

We now describe our algorithm. First, we generate a bank of feature grouping matrices $\Phi = \left\{\mathbf{\Phi}^{(1)}, \mathbf{\Phi}^{(2)}, \cdots, \mathbf{\Phi}^{(b)}\right\}$ using ReNA. Each $\mathbf{\Phi}^{(i)}$ is generated using r samples from the training data set selected randomly with replacement. Then we begin the SGD loop for model training. At each iteration, which consists of a gradient calculation and a weight update, we sample a random $\mathbf{\Phi}^{(i)}$ from the bank Φ. We use $\mathbf{\Phi}^{(i)}$ to project the training samples onto a lower dimensional space, and compute gradients in this lower dimensional space. This operation affects only the weight matrix in the input layer of the neural network, while subsequent weights and all biases are treated in a standard way. Instead of computing the gradient with respect to the $h \times p$ dimensional matrix $\mathbf{W}_0$, where h is the number of units in the first hidden layer, we compute the gradient with respect to the $h \times k$ dimensional weight matrix, called $\hat{\mathbf{W}}_0 \stackrel{def}{=} \mathbf{W}_0\mathbf{\Phi}^T$. Computational operations with $\hat{\mathbf{W}}_0$ are much cheaper than those with $\mathbf{W}_0$ because $k \ll p$.

During the update of $\mathbf{W}_0$, we project the gradient back to the original space. This operation can be interpreted as using $\mathbf{W}_0\mathbf{\Phi}^T\mathbf{\Phi}$ as a weight matrix instead of $\mathbf{W}_0$. Since $\mathbf{\Phi}^T\mathbf{\Phi}\mathbf{x}$ is an approximation of $\mathbf{x}$, it is equivalent to deriving the weight matrix $\mathbf{W}_0$ from the approximation of the input. Feature grouping acts as a stochastic regularizer by forcing the model to learn from these approximated inputs.

We describe the resulting estimator for training a neural network with H layers in Algorithm 1. Since the weights $\mathbf{W}_0$ we learn match the original feature dimension, the grouping matrices can be discarded after training completes, and no special procedure is needed at test time.

2.3. Interpretation of the proposed approach

With randomized feature grouping matrices in the SGD, we are effectively computing the parameters such that:

$$\hat{\mathbf{\Theta}} = \arg\min_{\mathbf{\Theta}} \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{\mathbf{\Phi}}\left[L\left(f(\mathbf{\Phi}^T\mathbf{\Phi}\mathbf{x}_i;\mathbf{\Theta}), y_i\right)\right] \quad (2)$$

instead of Equation 1. We investigate the effect of this approach for generalized linear models (GLM) as used by Wager et al. (2013) to uncover dropout's properties. Here $\mathbf{\Theta} = \{\boldsymbol{\beta}\}$, where $\boldsymbol{\beta}$ is a vector of parameters. The generalized linear model framework models the response y given a feature vector $\mathbf{x}$ and the model parameter $\boldsymbol{\beta}$ as:

$$p\left(y \mid \mathbf{x}; \boldsymbol{\beta}\right) \triangleq h(y)\exp\left(y\mathbf{x}^T\boldsymbol{\beta} - A\left(\mathbf{x}^T\boldsymbol{\beta}\right)\right) \quad (3)$$

where $h(y)$ is a quantity independent of $\mathbf{x}$ and $\boldsymbol{\beta}$; and $A(.)$ is the log-partition function which is equivalent to $\|\mathbf{x}^T\boldsymbol{\beta}\|^2$ for a least squares regression or Gaussian model.

Algorithm 1 Training of a Neural Network with Feature Grouping as a Stochastic Regularizer

Require: Learning Rate η
Require: Initial Parameters for H layers
$\mathbf{\Theta} \triangleq \{\mathbf{W}_0, \mathbf{b}_0, \mathbf{W}_1, \mathbf{b}_1, \cdots, \mathbf{W}_H, \mathbf{b}_H\}$
Ensure: Generate a bank of feature grouping matrices where each is generated by randomly sampling r samples from the training data set with replacement
$\Phi = \left\{\mathbf{\Phi}^{(1)}, \mathbf{\Phi}^{(2)}, \cdots, \mathbf{\Phi}^{(b)}\right\}$
1: **while** stopping criteria not met **do**
2: Sample a minibatch of m samples from the training set $\{\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(m)}\}$ with corresponding labels $y^{(i)}$
3: Sample $\mathbf{\Phi}$ from the bank Φ.
4: Define $\mathbf{\Xi} \triangleq \left\{\hat{\mathbf{W}}_0, \mathbf{b}_0, \mathbf{W}_1, \mathbf{b}_1, \cdots, \mathbf{W}_H, \mathbf{b}_H\right\}$ where $\hat{\mathbf{W}}_0 \triangleq \mathbf{W}_0\mathbf{\Phi}^T$.
5: Compute gradient estimate:
$\mathbf{g} \leftarrow \frac{1}{m}\nabla_{\mathbf{\Xi}}\sum_i \mathcal{L}\left(f(\mathbf{\Phi}\mathbf{x}^{(i)};\mathbf{\Xi}), y^{(i)}\right)$
6: Apply updates:
- $\mathbf{W}_0 \leftarrow \mathbf{W}_0 - \eta\mathbf{g}_{\mathbf{W}_0}\mathbf{\Phi}$
where $\mathbf{g}_{\mathbf{W}_0} \triangleq \frac{1}{m}\nabla_{\hat{\mathbf{W}}_0}\sum_i \mathcal{L}\left(f(\mathbf{\Phi}\mathbf{x}^{(i)};\mathbf{\Xi}), y^{(i)}\right)$
- $\mathbf{b}_j \leftarrow \mathbf{b}_j - \eta\mathbf{g}_{b_j}$
where $\mathbf{g}_{b_j} \triangleq \frac{1}{m}\nabla_{\mathbf{b}_j}\sum_i \mathcal{L}\left(f(\mathbf{\Phi}\mathbf{x}^{(i)};\mathbf{\Xi}), y^{(i)}\right)$
for $j \in \{0, \cdots, H\}$
- $\mathbf{W}_j \leftarrow \mathbf{W}_j - \eta\mathbf{g}_{\mathbf{W}_j}$
where $\mathbf{g}_{\mathbf{W}_j} \triangleq \frac{1}{m}\nabla_{\mathbf{W}_j}\sum_i \mathcal{L}\left(f(\mathbf{\Phi}\mathbf{x}^{(i)};\mathbf{\Xi}), y^{(i)}\right)$
for $j \in \{1, \cdots, H\}$
7: **end while**

We now separate $\mathbf{\Phi}$ in two terms: $\mathbf{\Phi}^T\mathbf{\Phi} = \mathbf{\Omega} + \mathbf{\Delta}$ where $\mathbf{\Omega} = \mathbb{E}[\mathbf{\Phi}^T\mathbf{\Phi}]$ is the deterministic term and $\mathbf{\Delta}$ is zero-mean noise term such that $\mathbb{E}\left[\mathbf{\Delta}\right] = \mathbf{0}$. $\mathbf{\Omega}$ captures the commonalities across multiple realizations of ReNA. As these are shaped by the feature graph used to impose structure, $\mathbf{\Omega}$ typically resembles a graph smoothing operator. The sum in Equation 2 can then be written as:

$$\sum_{i=1}^{n} \mathbb{E}_{\mathbf{\Phi}}\left[L\left(f\left(\mathbf{\Phi}^T\mathbf{\Phi}\mathbf{x}_i;\mathbf{\Theta}\right), y_i\right)\right] \quad (4)$$

$$= \sum_{i=1}^{n} -y_i\mathbf{x}_i^T\mathbf{\Omega}\boldsymbol{\beta} + \mathbb{E}_{\mathbf{\Phi}}\left[A\left(\mathbf{x}_i^T\left(\mathbf{\Omega}+\mathbf{\Delta}\right)\boldsymbol{\beta}\right)\right] \quad (5)$$

We apply second-order Taylor approximation to the term $A\left(\mathbf{x}^T\left(\mathbf{\Omega}+\mathbf{\Delta}\right)\boldsymbol{\beta}\right)$ around $\mathbf{x}^T\mathbf{\Omega}\boldsymbol{\beta}$ as a standard quadratic approximation also used by Bishop (1995); Rifai et al. (2011); Wager et al. (2013) and take the expectation:

$$\mathbb{E}_{\mathbf{\Phi}}\left[A\left(\mathbf{x}^T(\mathbf{\Omega}+\mathbf{\Delta})\boldsymbol{\beta}\right)\right] \approx A\left(\mathbf{x}^T\mathbf{\Omega}\boldsymbol{\beta}\right) + \frac{1}{2}A''\left(\mathbf{x}^T\mathbf{\Omega}\boldsymbol{\beta}\right)\mathbb{E}_{\mathbf{\Phi}}\left[\|\mathbf{x}^T\mathbf{\Delta}\boldsymbol{\beta}\|^2\right] \quad (6)$$

The first-order term $\mathbb{E}_{\mathbf{\Phi}}\left[A'\left(\mathbf{x}^T\mathbf{\Omega}\boldsymbol{\beta}\right)\mathbf{x}^T\mathbf{\Delta}\boldsymbol{\beta}\right]$ vanishes be-

cause $\mathbb{E}[\boldsymbol{\Delta}] = \mathbf{0}$. Substituting this into Equation 5 gives:

$$\sum_{i=1}^{n} \mathbb{E}_{\boldsymbol{\Phi}}\left[L\left(f\left(\boldsymbol{\Phi}^T\boldsymbol{\Phi}\mathbf{x}_i;\boldsymbol{\Theta}\right),y_i\right)\right] \approx \sum_{i=1}^{n} \underbrace{-y_i\mathbf{x}_i^T\boldsymbol{\Omega}\boldsymbol{\beta} + A\left(\mathbf{x}_i^T\boldsymbol{\Omega}\boldsymbol{\beta}\right)}_{L(\boldsymbol{\Omega}\mathbf{x}_i,y_i;\boldsymbol{\beta})} + \underbrace{\frac{1}{2}\sum_{i=1}^{n} A''\left(\mathbf{x}_i^T\boldsymbol{\Omega}\boldsymbol{\beta}\right)\mathrm{Var}_{\boldsymbol{\Phi}}\left[\mathbf{x}_i^T\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{\beta}\right]}_{\triangleq R(\boldsymbol{\beta})} \quad (7)$$

The above equation shows that our cost function consists of two terms: (i) the loss on the smoothed input $\boldsymbol{\Omega}\,\mathbf{X}$ (ii) a regularization cost $R(\boldsymbol{\beta})$. It is known that the term $A''\left(\mathbf{x}_i^T\boldsymbol{\beta}\right)$ corresponds to the variance of y_i given $\mathbf{x}_i$ under GLM settings (McCulloch & Neuhaus, 2001). Hence $A''\left(\mathbf{x}_i^T\boldsymbol{\Omega}\boldsymbol{\beta}\right)$ is the variance of the model given the smoothed input features $\boldsymbol{\Omega}\mathbf{x}_i$. Note that this term is constant for linear regression and equivalent to $p_i(1-p_i)$ where $p_i = 1/(1-\exp\left(-\mathbf{x}_i^T\boldsymbol{\Omega}\boldsymbol{\beta}\right))$ with feature grouping for logistic regression.

The term $\mathrm{Var}_{\boldsymbol{\Phi}}\left[\mathbf{x}_i^T\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{\beta}\right]$ corresponds to the variance of the estimated target due to the randomization introduced by stochastic regularizer. Using the definition $\boldsymbol{\Phi}^T\boldsymbol{\Phi} = \boldsymbol{\Omega} + \boldsymbol{\Delta}$ and symmetricity of $\boldsymbol{\Delta}$, it reduces to:

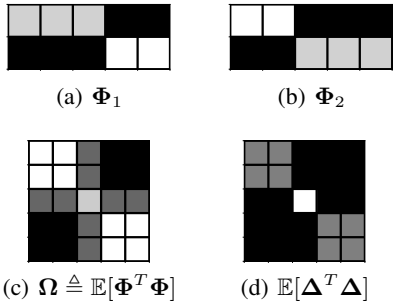
$$\begin{aligned}\mathrm{Var}_{\boldsymbol{\Phi}}\left[\mathbf{x}_i^T\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{\beta}\right] &= \mathrm{Var}_{\boldsymbol{\Phi}}\left[\mathbf{x}_i^T\boldsymbol{\Delta}\boldsymbol{\beta}\right] \\ &= \boldsymbol{\beta}^T\mathbb{E}\left[\boldsymbol{\Delta}\mathbf{x}_i\mathbf{x}_i^T\boldsymbol{\Delta}\right]\boldsymbol{\beta} \quad (8)\end{aligned}$$

If we used a $\boldsymbol{\Phi}$ matrix corresponding to a dropout on the input layer, randomly masking features, we would have $\boldsymbol{\Omega} = \mathbf{I}$ and $\boldsymbol{\Delta}$ a diagonal matrix where each i-th diagonal term has $\mathbb{E}[\Delta_i] = 0$ and $\mathbb{E}\left[\Delta_i^2\right] = \delta/(1-\delta)$ where δ is the dropout probability. Assuming $\mathbb{E}[\Delta_i\Delta_j] = 0$ for $i \neq j$, it can be written as:

$$\mathrm{Var}_{\boldsymbol{\Phi}}\left[\mathbf{x}_i^T\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{\beta}\right] = \frac{\delta}{1-\delta}\sum_{j=1}^{p} x_{ij}^2\beta_j^2 \quad (9)$$

where x_{ij} is the j-th entry of $\mathbf{x}_i$. For linear regression, this is equivalent to ridge regression after orthogonalizing the features.

However, for feature grouping, the matrix $\mathbb{E}\left[\boldsymbol{\Delta}\mathbf{x}_i\mathbf{x}_i^T\boldsymbol{\Delta}\right]$ rescales the feature vector $\mathbf{x}_i$ by the variance of the cluster membership for each feature. For instance, if a feature consistently appears in a certain cluster, then the membership variance for this feature will be low. If, on the other hand, a feature appears in a certain cluster only in half of the samples, then the variance will be high and will have a large weight in penalty term $\mathrm{Var}_{\boldsymbol{\Phi}}\left[\mathbf{x}_i^T\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{\beta}\right]$. As the clusters in $\boldsymbol{\Phi}$ are obtained from bootstrap replicates of the data, this penalty term captures the local spatial stability of the data.



(a) $\boldsymbol{\Phi}_1$ (b) $\boldsymbol{\Phi}_2$ (c) $\boldsymbol{\Omega} \triangleq \mathbb{E}[\boldsymbol{\Phi}^T\boldsymbol{\Phi}]$ (d) $\mathbb{E}[\boldsymbol{\Delta}^T\boldsymbol{\Delta}]$

Figure 2. Visualization for a toy example where number of feature is $p = 5$ and number of clusters if $k = 2$. (a) and (b) The two feature-grouping matrices $\boldsymbol{\Phi}_1$ and $\boldsymbol{\Phi}_2$ (c) Average of $\boldsymbol{\Phi}_1$ and $\boldsymbol{\Phi}_2$; i.e. $\boldsymbol{\Omega}$ (d) Variance of $\boldsymbol{\Phi}$, i.e. estimated variance of $\boldsymbol{\Delta}$. $\boldsymbol{\Omega}$ indeed does appear as a smoothing matrix, and $\mathbb{E}[\boldsymbol{\Delta}^T\boldsymbol{\Delta}]$ captures the spatial homogeneity: it is large for the central feature while the sides are more smoothed by $\boldsymbol{\Omega}$ and stabilized by the edges.

Figure 2 illustrates these terms with a toy example. Our bank of feature-grouping matrices is made of 2 matrices, $\boldsymbol{\Phi}_1$ and $\boldsymbol{\Phi}_2$. Note that the third feature appears with the first two features in matrix $\boldsymbol{\Phi}_1$ whereas it appears with the last two features in $\boldsymbol{\Phi}_2$. Figure 2(c) shows the average of $\boldsymbol{\Phi}_i^T\boldsymbol{\Phi}_i$ which captures the general topography of the groups. Figure 2(d) shows the variance of these two matrices which captures the high variance of feature 3. This way, our algorithm penalizes the features that are more noisy via the term $\mathrm{Var}_{\boldsymbol{\Phi}}\left[\mathbf{x}_i^T\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{\beta}\right]$ in $R(\boldsymbol{\beta})$ while used smoothed features $\boldsymbol{\Omega}$ in the optimized loss function $L\left(\boldsymbol{\Omega}\mathbf{x}_i, y_i; \boldsymbol{\beta}\right)$ and the regularized term $A''\left(\mathbf{x}^T\boldsymbol{\Omega}\boldsymbol{\beta}\right)$ in $R(\boldsymbol{\beta})$.

2.4. Computational Complexity

The computational complexity of optimizing a given neural network with the feature-grouping stochastic regularizer differs from the standard approach only for the parameter $\mathbf{W}_0$. Learning the rest of the parameters $\{\mathbf{b}_0, \mathbf{W}_1, \mathbf{b}_1, \cdots, \mathbf{b}_H, \mathbf{W}_H\}$ is unchanged. Therefore, it is sufficient to compare performance for logistic regression where the size of $\mathbf{W}_0$ is $l \times p$ instead of $h \times p$ where l is the total number of classes. The computational complexity of logistic regression, solved with the stochastic regularizer using feature grouping breaks down in four parts: *(i)* computation of the bank of $\boldsymbol{\Phi}$ matrices (Step 1 in Algorithm 1;) *ii)* multiplication by $\boldsymbol{\Phi}$ in summation in Step 6; *(iii)* computation of gradient in Step 6; and *iv)* update in Step 7.

The computational complexity of computing each $\boldsymbol{\Phi}$ using ReNA is $\mathcal{O}\left(rp\log\left(p/k\right)\right)$ (Hoyos-Idrobo et al., 2019) where r is the number of samples used, p is the number of features and k is the number of clusters. Since the bank has b such matrices, the total computational complexity of computing the bank is $\mathcal{O}\left(brp\log\left(p/k\right)\right)$. This is a constant factor independent of the number of iterations.

Computing $\mathbf{\Phi x}$, where $\mathbf{\Phi}$ is of dimension $p \times k$ would be $\mathcal{O}(kp)$. However, as $\mathbf{\Phi}$ is sparse, this reduces to $\mathcal{O}(p)$. Since there are m samples in a minibatch, the total computational complexity is $\mathcal{O}(mp)$. Computational complexity of gradient computation for a row of $\hat{\mathbf{W}}_0$ for a given sample $\mathbf{x}^{(i)}$ is $\mathcal{O}(k)$. Computing the gradient across all rows and samples in a minibatch has complexity $\mathcal{O}(lmk)$, l being the number of tasks. Updating one row of $\mathbf{W}_0$ requires right multiplying the gradient with respect to $\hat{\mathbf{W}}_0$ by $\mathbf{\Phi}^T$ which would be $\mathcal{O}(kp)$, but due to the sparse structure of $\mathbf{\Phi}$, it reduces to $\mathcal{O}(p)$. As there are l rows, the total computational complexity for update is $\mathcal{O}(lp)$.

Projection, gradients computation, and update are done for each epoch, so the computational complexity for the full iteration would be $\mathcal{O}(T\,m\,p + T\,l\,m\,k + T\,l\,p)$ where T is the total number of iterations. Hence the total computational complexity of logistic regression with feature grouping using ReNA can be written as:

$$\mathcal{O}(b\,r\,p\log(p/k) + T\,m\,p + T\,l\,m\,k + T\,l\,p) \qquad (10)$$

which is linear in the dimension of the input size p and number of classes l. Computational complexity of standard logistic regression, on the other hand, is $\mathcal{O}(T\,l\,m\,p)$.

3. Experiments

We presented a regularization algorithm that relies on feature grouping. Our approach can be easily integrated into fully-connected feedforward neural networks. In order to validate the effectiveness of our algorithm and compare it with conventional approaches we experiment in noisy and low sample size settings on face (Olivetti) and neuroimaging (HCP) datasets:

Olivetti Faces: The Olivetti dataset consists of grayscale 64×64 face images from 40 subjects (HOPPER, 1992). For each subject, there are 10 different images with varying lighting and facial expressions. The target class for this data set is the identity of the individual whose picture was taken. We randomly split the data into test and train such that the test dataset has 132 samples and the training dataset, 268 samples. As the faces are well centered, the data has a strong non-translation-invariant structure.

HCP: The Human Connectome Project (HCP) has released a large openly-accessible fMRI dataset. Here we use task fMRI that includes seven tasks: 1. Working Memory, 2. Gambling, 3. Motor, 4. Language 5. Social Cognition, 6. Relational Processing, and 7. Emotion Processing. These tasks have been chosen to map different brain systems. The dataset includes 500 different subjects with images registered to the standard MNI atlas. For a given subject and task, a GLM was fitted to each fMRI dataset (Barch et al., 2013). Then volumetric contrasts of parameter estimate (COPE) were computed to assess differences between different task components, resulting into brain maps. We use 20 different contrasts as described in Table A.1. fMRI data are sampled in a common space of $91 \times 109 \times 91$ with 2mm isotropic voxels. We transformed 3D data into 1D arrays of size $p = 270\,806$ for our supervised classification algorithms. Our goal is to classify 20 cognitive contrasts given $p = 270\,806$ features. The test dataset includes $1\,964$ samples with at least 95 samples from each target class whereas the training set has $7\,785$ samples.

HCP - small: In order to perform fast experimentation, we use a smaller number of classes and voxels from the HCP data set. We select 8 different contrasts from tasks: 1. Working Memory, 2. Gambling, 3. Relational, 4. Emotion, and 5. Social as described in Table A.2 that are harder to classify. fMRI data are resampled to a common space of $46 \times 55 \times 46$ with 4mm isotropic voxels. We transformed 3D data into 1D arrays of size $33,854$. Our goal is to classify 8 cognitive contrasts given $33,854$ features. The test data includes 791 samples with at least 97 samples from each target class whereas the training set has 3052 samples.

3.1. Architectures

We used three typical machine learning architectures in our experiments: (i) logistic regression (ii) multilayer perceptron (MLP) with a single hidden layer of size 256, and (iii) convolutional neural network (CNN) (LeCun et al., 2015b) that consists of two convolutional layers, two sub-sampling (pooling) layers and two fully connected layers. Convolutional layers for the Olivetti data set used 5×5 convolutions with stride 1 and the sub-sampling layers are 2×2 max pooling layers. Convolutional layers for the HCP and HCP-small data sets used 7×7 and 5×5 convolutions with stride 2 in the first and second layers, respectively and the sub-sampling layers are 2×2 max pooling layers. ReLU activation functions are used both in MLP and CNN.

3.2. Training

We use the standard SGD algorithm with learning rate 0.01 for the Olivetti dataset and 0.05 for the HCP dataset. We use a cross entropy loss. We ran experiments for logistic regression long enough (200 epochs for Olivetti and 500 epochs for HCP and HCP-small) to guarantee convergence. We applied *early stopping* on MLP and CNN architectures when the validation loss stopped improving in 10 (also known as patience parameter) subsequent epochs. We repeated each experiment with 10 different random initializations.

3.3. Parameter tuning for regularizers

We vary the regularization parameter for ℓ_2 from 10^{-7} to 10 with a grid of factors of 10, and the dropout probability parameter for dropout from 0.1 to 0.7 with a grid of 0.2 for

Figure 3. Sampled feature grouping matrices from the bank. Each matrix is computed using randomly selected 50 samples from the Olivetti faces training data set. Note that each row of $\mathbf{\Phi}$ matrix is reshaped to the size of the image and then all rows are overlayed for visualization purposes.

logistic regression and MLP architectures. We used only dropout with dropout probability 0.5 for CNNs.

For our approach, each feature grouping matrix $\mathbf{\Phi}$ is computed over $r = 50$ randomly picked samples from the training data set for k clusters, where k is set to 10 % of the total number of features. For each epoch during training, a feature grouping matrix was randomly picked from the bank of $b = 100$ matrices. Figure 3 shows samples of feature grouping matrices from the bank. Empirically, feature grouping regularization is not very sensitive to the choice of b and r (Table A.3). Moreover, we see that the performance degrades only with extremely small number of k (Table A.4). For MLP, we also combine feature grouping with dropout at intermediate layers to regularize them.

3.4. Computational details

We use Python 3.6 for implementation (Oliphant, 2007) using open-source libraries PyTorch (Paszke et al., 2017), scikit-learn (Pedregosa et al., 2011), NiBabel (Brett et al., 2016), nilearn (Abraham et al., 2014), joblib (Varoquaux & Grisel, 2009) and NumPy (Walt et al., 2011). Experiments are run using Nvidia GeForce GTX 1060 and 16GB RAM. Our implementation is openly available at `https://github.com/sergulaydore/Feature-Grouping-Regularizer`.

3.5. Results in noisy settings

In order to explore robustness of classification approaches with different regularizers, we add zero-mean Gaussian noise with varying standard deviations. Here, we use Olivetti faces and HCP-small data sets. The SNR (the ratio of power of signal to noise) of Olivetti and HCP-small becomes 3 dB and -2 dB respectively with moderate additional noise. These values reduce to 0.6 dB and -5 dB with severe additional noise. We trained three architectures using different regularizers for three noise levels (none, medium and high). We report the test accuracy results with average and standard error computed over 10 experiments in Table 1. We report the best results across ℓ_2 and dropout parameters.

For the Olivetti faces dataset, MLP with dropout outperforms other architectures and regularizers when there is no additive Gaussian noise. However, it does not retain its performance as Gaussian noise is added. Architectures trained with feature grouping, on the other hand, are robust to increasing noise level. Although CNNs do not have an impressive performance, their performance degrades less quickly with noise compared to the other architectures with ℓ_2 and dropout.

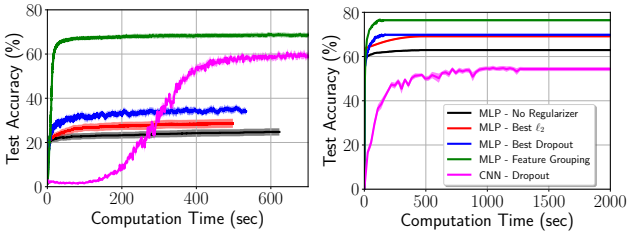
	MODEL	REGULARIZER	TEST ACCURACY (%)	
			OLIVETTI	HCP-small
No noise	LR	None	85.23±0.70	86.80±0.18
		Best ℓ_2	86.52±1.01	87.22±0.12
		Best dropout	85.76±0.88	87.35±0.15
		feature grouping	**86.52±0.68**	**87.37±0.29**
	MLP	None	85.38 ± 1.10	88.02 ± 0.18
		Best ℓ_2	87.73±0.81	**88.31±0.14**
		Best dropout	**89.55±0.88**	87.72±0.13
		feature grouping	85.45±1.08	87.36±0.57
	CNN	dropout, $p = .5$	83.56±1.43	74.96±0.61
Medium noise level	LR	None	50.83±1.79	79.77±0.35
		Best ℓ_2	51.06±1.16	79.97±0.36
		Best dropout	52.20±1.21	79.90±0.30
		feature grouping	**80.00±0.83**	**84.16±0.24**
	MLP	None	54.55±1.63	76.94±0.20
		Best ℓ_2	56.59±1.53	80.66±0.22
		Best dropout	61.82±1.14	80.01±0.42
		feature grouping	**80.91±1.02**	**83.75±0.35**
	CNN	dropout, $p = .5$	77.65±1.11	63.94±1.27
High noise level	LR	None	22.27±0.54	71.42±0.61
		Best ℓ_2	24.62±1.50	71.76±0.43
		Best dropout	24.09±1.54	72.10±0.48
		feature grouping	**64.55±1.77**	**77.93±0.38**
	MLP	None	25.00±1.89	62.92±0.40
		Best ℓ_2	28.56±2.09	69.13±0.32
		Best dropout	34.02±1.48	69.81±0.56
		feature grouping	**68.79±1.04**	**76.45±0.52**
	CNN	dropout, $p = .5$	56.89±1.62	54.35±0.93

Table 1. Average and standard error of test accuracy for different regularizers at various noise levels for the Olivetti and HCP-small data sets for logistic regression (LR), MLP and CNN models.

Similarly, for HCP-small data set, architectures with feature grouping are more robust against additive Gaussian noise. Unlike the Olivetti data set, CNNs perform poorly for all noise settings. This could be because we use 2D convolutions instead of 3D convolutions for a 3D data set. However, 3D convolutions demand much more memory than our available computational resources. Furthermore, the *translation invariance* property of CNNs does not help for brain images, and is in fact detrimental.

We compare the computational performances of CNN with dropout and MLP with different regularizers for Olivetti faces and HCP-small under high noise settings. Figure 4 clearly shows that MLP with feature grouping achieves higher accuracy in shorter time despite the high noise for both data sets.

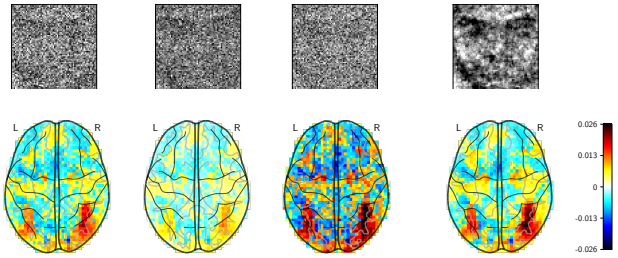
We also show in Figure 5 the learned weights averaged over 10 different initializations for each approach. The weights



(a) Olivetti Faces (b) HCP-small

Figure 4. **Noisy settings:** Performance in terms of test accuracy as a function of computation time for neural networks using feature grouping and best parameters for other regularizers with high noise (a) Olivetti Faces (b) HCP-small.

from the feature grouping approach visually look less noisy, which explains the superior performance of this approach in noisy settings.



(a) (b) (c) (d)

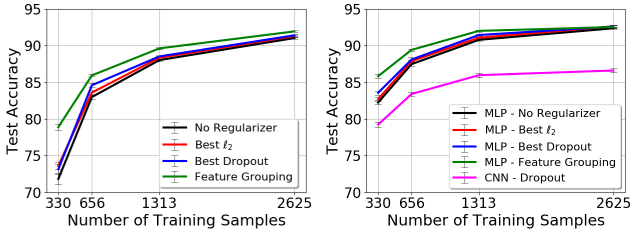
Figure 5. Visualization of the learned weights for logistic regression. Top: Olivetti faces data set with high noise level for an arbitrarily selected subject. Bottom: a single subject while performing FACES task. (a) No regularizer (b) Best ℓ_2 (c) Best Dropout (d) Feature Grouping

3.6. Results for small-sample settings

We explore the robustness of different approaches in small-sample settings for the HCP data set. Figure 6 shows logistic regression and MLP learning curves with each regularizer: the test accuracy when different numbers of samples are used. It shows that feature grouping clearly outperforms the other approaches both for logistic regression and MLP when fewer samples are used. MLPs perform better than logistic regression even for small sample sizes. Similar to the results from HCP-small, CNNs do not perform well on this data set. The difference between feature grouping, ℓ_2, and dropout disappears as the number of samples increases.

4. Conclusion

We propose a new stochastic regularizer, based on feature clustering and averaging, randomized inside an SGD loop.



(a) Logistic Regression (b) Neural Networks

Figure 6. **Small-sample settings:** Performance in terms of test accuracy as a function of number of samples using feature grouping and best parameters for other regularizers, for HCP data set.

Our regularizer directly exploits structure in the data by constructing clusters of correlated features. This makes it particularly well-suited to data with very high dimensionality. Unlike classic structured regularizers, our approach can be plugged into any model, including non-convex ones, solved by gradient descent. In deep architectures, it operates on the input layer, which is likely to contain more parameters than subsequent layers in high-dimensional problems.

We demonstrate the effectiveness of our regularizer on two problems with structure in the features: frame-aligned faces and fMRI. In both cases, our method outperforms dropout, ℓ_2 regularization, and convolutional neural networks when the noise level increases. On the fMRI data, we also show that our method performs best as the sample size decreases.

Our approach comes with little computational cost: it only adds to the SGD update loop a cost linear in the feature dimension, but reduces the memory usage of subsequent steps. Experimental results confirm that neural networks trained with our regularizer converge in the same amount of time as with other regularizers, but with higher accuracy.

Regularizations can be seen as modifying the objective function optimized during training. Our stochastic regularizer forces the model to learn from smoothed inputs. Its effect decomposes into interpretable components: the loss on the smoothed inputs, and a regularization term which shrinks model weights for the noisiest feature clusters. We also draw connections to dropout.

The approach proposed here introduces new ideas for developing structured regularizations, departing from the classic framework of engineering penalties. Our findings suggest that the combination of structured random matrices and stochastic optimization for regularization should be explored further as it is versatile and computationally efficient. The approach should be tested on other data that present a strong stationary structure, such as spectra in chemistry. Lastly, beyond ReNA, other clustering approaches could be used, including other agglomerative approaches (Müllner, 2011), or density-based methods.

Acknowledgements

This project has received funding from the European Unions Horizon 2020 Research and Innovation Programme under Grant Agreement No. 785907 (HBP SGA2). We would like to thank Olivier Grisel, Syed Ashrafulla and Philip Gautier for valuable discussions.

References

Abraham, A., Pedregosa, F., Eickenberg, M., Gervais, P., Mueller, A., Kossaifi, J., Gramfort, A., Thirion, B., and Varoquaux, G. Machine learning for neuroimaging with scikit-learn. *Frontiers in neuroinformatics*, 8:14, 2014.

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Süsstrunk, S. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012.

Achlioptas, D. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of computer and System Sciences*, 66(4):671–687, 2003.

Alaoui, A. and Mahoney, M. W. Fast randomized kernel ridge regression with statistical guarantees. In *Advances in Neural Information Processing Systems*, pp. 775–783, 2015.

Bach, F., Jenatton, R., Mairal, J., and Obozinski, G. Structured sparsity through convex optimization. *Statistical Science*, pp. 450–468, 2012.

Bachman, P., Alsharif, O., and Precup, D. Learning with pseudo-ensembles. In *Advances in Neural Information Processing Systems*, pp. 3365–3373, 2014.

Barch, D. M., Burgess, G. C., Harms, M. P., Petersen, S. E., Schlaggar, B. L., Corbetta, M., Glasser, M. F., Curtiss, S., Dixit, S., Feldt, C., et al. Function in the human connectome: task-fmri and individual differences in behavior. *Neuroimage*, 80:169–189, 2013.

Bingham, E. and Mannila, H. Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 245–250. ACM, 2001.

Bishop, C. M. Training with noise is equivalent to tikhonov regularization. *Neural computation*, 7(1):108–116, 1995.

Bouthillier, X., Konda, K., Vincent, P., and Memisevic, R. Dropout as data augmentation. *arXiv preprint arXiv:1506.08700*, 2015.

Brett, M., Hanke, M., Cipollini, B., Côté, M.-A., Markiewicz, C., Gerhard, S., Larson, E., Lee, G. R., Halchenko, Y., Kastman, E., et al. nibabel: 2.1. 0. *Zenodo*, 2016.

Bühlmann, P., Rütimann, P., van de Geer, S., and Zhang, C.-H. Correlated variables in regression: clustering and sparse estimation. *Journal of Statistical Planning and Inference*, 143(11):1835–1858, 2013.

Cannings, T. I. and Samworth, R. J. Random-projection ensemble classification. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 79(4):959–1035, 2017.

Consortium, . G. P. et al. A global reference for human genetic variation. *Nature*, 526(7571):68, 2015.

Durrant, R. J. and Kabán, A. Random projections as regularizers: Learning a linear discriminant ensemble from fewer observations than dimensions. 2013.

Fan, J. and Li, R. Statistical challenges with high dimensionality: Feature selection in knowledge discovery. *arXiv preprint math/0602133*, 2006.

García-Torres, M., Gómez-Vela, F., Melián-Batista, B., and Moreno-Vega, J. M. High-dimensional feature selection via feature grouping: A variable neighborhood search approach. *Information Sciences*, 326:102–118, 2016.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

HOPPER, A. The orl face database. at&t (olivetti) research laboratory cambridge, 1992.

Hoyos-Idrobo, A., Varoquaux, G., Kahn, J., and Thirion, B. Recursive nearest agglomeration (rena): fast clustering for approximation of structured signals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41 (3):669–681, 2019.

Jacob, L., Obozinski, G., and Vert, J.-P. Group lasso with overlap and graph lasso. In *Proceedings of the 26th annual international conference on machine learning*, pp. 433–440. ACM, 2009.

Kalchbrenner, N., Grefenstette, E., and Blunsom, P. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pp. 655–665, 2014.

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *Nature*, 521(7553):436–444, 2015a.

LeCun, Y. et al. Lenet-5, convolutional neural networks. *URL: http://yann. lecun. com/exdb/lenet*, pp. 20, 2015b.

McCallum, A., Nigam, K., and Ungar, L. H. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 169–178. ACM, 2000.

McCulloch, C. E. and Neuhaus, J. M. *Generalized linear mixed models*. Wiley Online Library, 2001.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Müllner, D. Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378*, 2011.

Mwangi, B., Tian, T. S., and Soares, J. C. A review of feature reduction techniques in neuroimaging. *Neuroinformatics*, 12(2):229–244, 2014.

Oliphant, T. E. Python for scientific computing. *Computing in Science & Engineering*, 9(3), 2007.

Paszke, A., Gross, S., Chintala, S., and Chanan, G. Pytorch, 2017.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

Plis, S. M., Hjelm, D. R., Salakhutdinov, R., Allen, E. A., Bockholt, H. J., Long, J. D., Johnson, H. J., Paulsen, J. S., Turner, J. A., and Calhoun, V. D. Deep learning for neuroimaging: a validation study. *Frontiers in neuroscience*, 8, 2014.

Ren, J. S., Xu, L., Yan, Q., and Sun, W. Shepard convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 901–909, 2015.

Rifai, S., Glorot, X., Bengio, Y., and Vincent, P. Adding noise to the input of a model trained with a regularized objective. *arXiv preprint arXiv:1104.3250*, 2011.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Suk, H.-I., Wee, C.-Y., Lee, S.-W., and Shen, D. State-space model with deep learning for functional dynamics estimation in resting-state fmri. *NeuroImage*, 129:292–307, 2016.

Thalamuthu, A., Mukhopadhyay, I., Zheng, X., and Tseng, G. C. Evaluation and comparison of gene clustering methods in microarray analysis. *Bioinformatics*, 22(19): 2405–2412, 2006.

Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

Van Essen, D. C., Smith, S. M., Barch, D. M., Behrens, T. E., Yacoub, E., Ugurbil, K., Consortium, W.-M. H., et al. The wu-minn human connectome project: an overview. *Neuroimage*, 80:62–79, 2013.

Varoquaux, G. and Grisel, O. Joblib: running python function as pipeline jobs. *packages. python. org/joblib*, 2009.

Varoquaux, G., Gramfort, A., and Thirion, B. Small-sample brain mapping: sparse recovery on spatially correlated designs with randomization and clustering. In *International Conference on Machine Learning*, 2012.

Vinh, N. X., Erfani, S., Paisitkriangkrai, S., Bailey, J., Leckie, C., and Ramamohanarao, K. Training robust models using random projection. In *Pattern Recognition (ICPR), 2016 23rd International Conference on*, pp. 531–536. IEEE, 2016.

Wager, S., Wang, S., and Liang, P. S. Dropout training as adaptive regularization. In *Advances in neural information processing systems*, pp. 351–359, 2013.

Walt, S. v. d., Colbert, S. C., and Varoquaux, G. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.

Xu, R. and Wunsch, D. Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–678, 2005.

Yuan, M. and Lin, Y. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68 (1):49–67, 2006.

Zhao, P., Rocha, G., and Yu, B. The composite absolute penalties family for grouped and hierarchical variable selection. *The Annals of Statistics*, pp. 3468–3497, 2009.