Using Feature Grouping as a Stochastic Regularizer for High-Dimensional Noisy Data

Sergul Aydore,
Electrical and Computer Engineering,
Stevens Institute of Technology, NJ, USA,
sergulaydore@gmail.com

Bertrand Thirion, Olivier Grisel, Gaël Varoquaux,
INRIA, Parietal team, Saclay, France
CEA, Neurospin, Gif-sur-Yvette, France
firstname.lastname@inria.fr

Abstract

The use of complex models—with many parameters—is challenging with high-dimensional small-sample problems: indeed, they face rapid overfitting. Such situations are common when data collection is expensive, as in neuroscience, biology, or geology. Dedicated regularization can be crafted to tame overfit, typically via structured penalties. But rich penalties require mathematical expertise and entail large computational costs. Stochastic regularizers such as dropout are easier to implement: they prevent overfitting by random perturbations. Used inside a stochastic optimizer, they come with little additional cost. We propose a structured stochastic regularization that relies on feature grouping. Using a fast clustering algorithm, we define a family of groups of features that capture feature covariations. We then randomly select these groups inside a stochastic gradient descent loop. This procedure acts as a structured regularizer for high-dimensional correlated data without additional computational cost and it has a denoising effect. We demonstrate the performance of our approach for logistic regression both on a sample-limited face image dataset with varying additive noise and on a typical high-dimensional learning problem, brain image classification.

1 Introduction

Recent developments in machine learning have lead to impressive gains in accuracy in various fields such as computer vision, speech processing, and natural language processing [18, 22]. Yet, these complex models have not been applied much to high-dimensional data-scarce situations such as brain imaging, despite a clear potential [28, 32]. Two main challenges of neuroimaging data—and more generally, life science and medicine data—are i) the difficulty and cost of data collection and labeling ii) the high dimensionality of the data due to rich temporal or spatial resolution.

Various approaches can mitigate overfitting. These include early stopping, penalizing weights ($\ell_1$ and $\ell_2$ regularization) [34] and weight sharing [25]. For a limited amount of data and unlimited computation resources, the best way is to “regularize” the weights. However, this requires training many different models to find optimum hyperparameters which is computationally a tedious task. Recently, Srivastava et al. [31] proposed a dropout technique that modifies the network structure at each epoch. The key idea is to randomly remove units in the network during training and use an approximate averaging procedure across these “thinned” networks during testing.

Preprint. Work in progress.
Regularization with weight penalties ($\ell_1$ and $\ell_2$) are designed for individual feature selection rather than the strength of the groups of features. However, it is known that for high-dimensional data some features might be highly correlated or irrelevant [10]. Clustering or grouping features into meaningful regions is an effective way to reduce the redundancy and complexity of machine learning models [14]. Therefore, it is a typical procedure to use feature grouping for machine learning tasks in high-dimensional settings [20, 33, 40]. The use of random matrices to project data into lower dimensional space is also a computationally efficient and effective dimensionality reduction method [6, 3, 12, 19]. Although random projections are not conventionally used for regularization, dropout at the visible layer can be viewed as a data augmentation with random projections [8, 37, 29, 13].

Yuan and Lin [41] introduced the group lasso, a generalization of the lasso ($\ell_1$ regularizer), to enforce structured sparsity —namely group-wise selection— for linear models and Zhao et al. [42] extended this approach to more general $\ell_2$ norms. However, it is computationally expensive and generalization beyond linear models is not clear.

In this work, we use feature grouping to develop an alternative, computationally efficient, stochastic regularization approach. Our algorithm relies on a bank of projection matrices to group the features for training. We sample from this bank at each stochastic gradient iteration and project the training data set to this reduced dimension. The gradient is computed in this reduced dimensional space. In order to update the weights, we project the gradient step back to the original feature space. This procedure results in weights expressed on the original features (brain voxels for neuroimaging; pixels for image processing), that can be used at test time. This way, we can avoid overfitting without increasing computational complexity, as the gradient is estimated in a smaller space. Note that our algorithm is designed to be implemented in the feature space, hence if it is applied in neural networks, it can be applied only at the input layer. Therefore, we restrict our focus to logistic regression in this study.

The feature grouping approach we used is a linear-time agglomerative clustering scheme, Recursive Nearest Agglomeration (ReNA) proposed by Hoyos-Idrobo et al. [16]. The core idea of ReNA is analogous to super-pixels in computer vision, typically obtained through simple linear iterative clustering (SLIC) algorithm [2]. The advantages of using a fast averaging procedure are two-fold: (i) it has a denoising effect on structured signals (ii) it reduces the dimension of signals in linear time.

We first validate our approach on a face image dataset, the Olivetti data [15] with varying additive noise. Then, we apply our approach to the biggest publicly available task fMRI data set, obtained from the Human Connectome Project [35]. Functional magnetic resonance imaging (fMRI) is a noninvasive neuroimaging modality that measures brain activity during cognitive tasks in humans. For both data sets, we restrict our focus to logistic regression for classification problems. Experimental results demonstrate that the proposed approach provides faster and better accuracy while acting as a data denoiser under stochastic gradient descent (SGD) settings.

### 2 Model

Let $x \in \mathbb{R}^p$ represent a random vector with a corresponding target $y \in \mathbb{R}$; the parameters that define the relationship between these two variables are represented by a function $f(.) : \mathbb{R}^p \rightarrow \mathbb{R}$ with parameters $\Theta$. Typically, parameters of the model are estimated by minimizing the empirical risk over training samples $(x_i, y_i)$ for $i \in \{1, \cdots, n\}$ such that:

$$\hat{\Theta} = \arg \min_{\Theta} \frac{1}{n} \sum_{i=1}^{n} L(f(x_i; \Theta), y_i)$$  \hspace{1cm} (1)

where $L(.)$ is the cost per sample. For logistic regression, the parameter set can be defined as $\Theta = \{W, b\}$ where $W$ represents weight variables and $b$ represents bias variables.

#### 2.1 Dimensionality Reduction Algorithm by Feature Grouping

We assume that $x$ represents high-dimensional data with a strong spatial structure as with fMRI data where $p \sim 10^3 - 10^6$. Reducing the dimensionality of these signals reduces memory requirements and speeds up algorithmic steps. Such a reduction can only be beneficial if it captures the statistical structure of $x$. Structure-aware dimension reduction is thus common for high-dimensional neuroimaging data [24]. We use a data-driven feature averaging approach, ReNA. The signals are approximated
We describe the resulting estimator for training logistic regression in Algorithm 1. At test time, we use the ReNA clustering approach [16] as a feature grouping for logistic regression. We will call “feature dropout”, can be described exactly as Algorithm 1, replacing the forward propagate using estimated values for $W$ and $b$. Note that applying dropout for logistic regression, which we will call “feature dropout”, can be described exactly as Algorithm 1 replacing the feature-grouping matrix $\Phi$ by a masking matrix, i.e., an identity matrix where diagonal elements are masked to zero randomly.

To describe the feature-averaging algorithm, let $\Phi \in \mathbb{R}^{k \times p}$ be the feature-grouping matrix that projects the data to a lower-dimensional space, with $k \ll p$. The clusters are a partition of the features $P = \{C_1, C_2, \ldots, C_q\}$, where $C_q$ is the set of indices that belong to cluster $q$ and $C_q \cap C_l = \emptyset$ for $q \neq l$. Approximation on the $q$-th cluster can be written as: $(\Phi x)_q = \alpha_q \sum_{j \in C_q} x_j$, where $\alpha_q$ is constant for cluster $q$ and is chosen as $\alpha_q = 1/\sqrt{|C_q|}$ to make $\Phi$ an orthogonal projection matrix. To ease data analysis, the matrix $\Phi$ is designed to group the features in a way that captures the underlying structure of the data. Once the matrix $\Phi$ is created, $\Phi x \in \mathbb{R}^k$ can be used as a reduced version of $x$ and $\Phi^T \Phi x$ is a piecewise constant approximation of $x$. Here, we build $\Phi$ from the data, using clustering. We use the ReNA clustering approach [16] as it is very fast, ii) it approximates the data well. In theoretical settings where the data are modeled as random, $\Phi$ is then itself random. Our framework is more general than the use of ReNA to define $\Phi$, it only requires a procedure to sample $\Phi$ matrices giving good representations of the signal. Here, we set the number of clusters to 20% of the number of features to achieve good computation time while not compromising data fidelity.

### 2.2 Stochastic Regularizer with Feature Grouping

In this section, we describe our novel algorithm that prevents overfitting using feature grouping for logistic regression. We first generate a bank of projection matrices: $\Phi = \{\Phi^{(1)}, \Phi^{(2)}, \ldots, \Phi^{(b)}\}$, where each $\Phi^{(i)}$ is generated using $r \leq n$ randomly selected samples from the training dataset $\{x^1, x^2, \ldots, x^n\}$. For each iteration during training, we sample $\Phi^{(i)}$ from the bank $\Phi$. We project the training samples onto a lower dimensional space using this matrix. Gradients are computed for this reduced representation. This way, for $l$ classes, instead of computing the gradient with respect to the $l \times k$ dimensional matrix $W$, we compute the gradient with respect to the $l \times k$ dimensional weight matrix where $k \ll p$, say $W'$, which is equal to $W\Phi^T$. To update $W'$, we project the gradient back to the original space. This operation can be interpreted as using $W\Phi^T\Phi$ as a weight matrix instead of $W$. Since $\Phi^T\Phi x$ is an approximation of $x$, it is equivalent to deriving the weight matrix $W$ from the approximation of the input. Recall that we are adding randomization by sampling from the bank $\Phi$, and additionally, these matrices are sparse, leading to very fast multiplications.

We describe the resulting estimator for training logistic regression in Algorithm 1. At test time, we forward propagate using estimated values for $W$ and $b$. Note that applying dropout for logistic regression, which we will call “feature dropout”, can be described exactly as Algorithm 1 replacing the feature-grouping matrix $\Phi$ by a masking matrix, i.e., an identity matrix where diagonal elements are masked to zero randomly.

### Algorithm 1 Training of Logistic Regression with Feature Grouping as a stochastic Regularizer

**Require:** Learning Rate $\epsilon$

**Require:** Initial Parameters $\Theta \triangleq \{W, b\}$

1. Generate a bank of projection matrices $\Phi = \{\Phi^{(1)}, \Phi^{(2)}, \ldots, \Phi^{(b)}\}$
2. while stopping criteria not met do
3. Sample a minibatch of $m$ samples from the training set $\{x^{(1)}, \ldots, x^{(m)}\}$ with corresponding labels $y^{(i)}$.
4. Sample $\Phi$ from the bank $\Phi$.
5. Define $\Xi \triangleq \{W, b\}$
6. Compute gradient estimate:
   $g \leftarrow \frac{1}{m} \nabla L \sum_{i} L \left(f(\Phi x^{(i)}; \Xi), y^{(i)}\right)$
7. Apply updates:
   $\bullet W \leftarrow W - \epsilon g_W \Phi$
   where $g_W = \frac{1}{m} \nabla W \sum_{i} L \left(f(\Phi x^{(i)}; \Xi), y^{(i)}\right)$
   $\bullet b \leftarrow b - \epsilon g_b$
   where $g_b = \frac{1}{m} \nabla b \sum_{i} L \left(f(\Phi x^{(i)}; \Xi), y^{(i)}\right)$
8. end while
2.3 Some insights: random reductions while fitting

With a randomized feature grouping in the stochastic gradient descent, we are effectively computing the parameters such that:

$$\hat{\Theta} = \arg \min_{\Theta} \frac{1}{n} \sum_{i=1}^{n} E_{\Phi} \left[ L \left( f(\Phi^{T} \Phi x_{i} ; \Theta), y_{i} \right) \right]$$

(2)

instead of equation 1. We investigate the effect of this approach for generalized linear models as used by Wager et al. [38] to uncover dropout’s properties. Let us assume that $\Theta = \{ \beta \}$, where $\beta$ is a vector of parameters. The generalized linear model framework models the distribution of the response $y$ given a feature vector $x$ and the model parameter $\beta$ as:

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

(3)

where $h(y)$ is a quantity independent of $x$ and $\beta$; and $A(.)$ is the log-partition function which is equivalent to $||x^{T} \beta||^2$ for least-squares regression or Gaussian models. The loss function (i.e. negative log-likelihood) is defined as $L \left( f(x; \Theta), y \right) = -\log p \left( y \mid x; \beta \right)$. Hence, the summation part of equation 2 can be written as:

$$\sum_{i=1}^{n} E_{\Phi} \left[ L \left( f(\hat{x}_{i} ; \Theta), y_{i} \right) \right] = \sum_{i=1}^{n} E_{\Phi} \left[ -(y_{i} \hat{x}_{i}^{T} \beta - A(\hat{x}_{i}^{T} \beta)) \right]$$

(4)

where $\hat{x}_{i} \triangleq \Phi^{T} \Phi x_{i}$. Now, let us define $\Phi^{T} \Phi = \Omega + \Delta$ where $\Omega = E[\Phi^{T} \Phi]$ is the deterministic term that acts as a smoothing term and $\Delta$ is zero-mean noise term such that $E[\Delta] = 0$. The equation above hence becomes:

$$\sum_{i=1}^{n} E_{\Phi} \left[ L \left( f(\hat{x}_{i} ; \Theta), y_{i} \right) \right] = \sum_{i=1}^{n} -(y_{i} \hat{x}_{i}^{T} \Omega \beta + E_{\Phi} \left[ A \left( \hat{x}_{i}^{T} \left( \Omega + \Delta \right) \beta \right) \right]$$

(5)

We apply second order Taylor approximation to the term $A \left( \hat{x}_{i}^{T} \left( \Omega + \Delta \right) \beta \right)$ around $x^{T} \Omega \beta$ as a standard quadratic approximation type also used by [7, 30, 38] and take the expectation that yields:

$$E_{\Phi} \left[ A \left( \hat{x}_{i}^{T} \left( \Omega + \Delta \right) \beta \right) \right] \approx A \left( x^{T} \Omega \beta \right) + \frac{1}{2} A'' \left( x^{T} \Omega \beta \right) E_{\Phi} \left[ ||x^{T} \Delta \beta||^2 \right].$$

(6)

The first order term in $E_{\Phi} \left[ A' \left( x^{T} \Omega \beta \right) x^{T} \Delta \beta \right]$ vanishes because $E[\Delta] = 0$. Substituting this into equation 5 gives:

$$\sum_{i=1}^{n} E_{\Phi} \left[ L \left( \hat{x}_{i}, y_{i} ; \Theta \right) \right] \approx \sum_{i=1}^{n} -y_{i} \hat{x}_{i}^{T} \Omega \beta + A \left( \hat{x}_{i}^{T} \Omega \beta \right) + \frac{1}{2} \sum_{i=1}^{n} A'' \left( \hat{x}_{i}^{T} \Omega \beta \right) E_{\Phi} \left[ ||x^{T} \Delta \beta||^2 \right]$$

$$= \sum_{i=1}^{n} L \left( \Omega x_{i}, y_{i} ; \beta \right) + \frac{1}{2} \sum_{i=1}^{n} A'' \left( \hat{x}_{i}^{T} \Omega \beta \right) \text{Var}_{\beta} \left[ x_{i}^{T} \Phi^{T} \Phi \beta \right]$$

(7)

The above equation shows that our cost function consists of two terms: (i) loss on the smoothed input (ii) regularization cost $R(\beta)$. It is known that the term $A'' \left( \hat{x}_{i}^{T} \beta \right)$ corresponds to the variance of $y_{i}$ given $x_{i}$ under GLM settings [21]. Hence, $A'' \left( \hat{x}_{i}^{T} \Omega \beta \right)$ is the variance of the model given the smooth input features $\Omega x_{i}$. Note that this term is constant for linear regression and equivalent to $p_{i}(1 - p_{i})$ where $p_{i} = 1/(1 - \exp(-\hat{x}_{i}^{T} \Omega \beta))$ with feature grouping for logistic regression.

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

$$\text{Var}_{\beta} \left[ x_{i}^{T} \Phi^{T} \Phi \beta \right] = \text{Var}_{\beta} \left[ x_{i}^{T} \Delta \beta \right] = \beta^{T} \text{E} \left[ \Delta x_{i} x_{i}^{T} \Delta \right] \beta$$

(8)
We illustrate this phenomenon in Figure 1 with a toy example. Let us assume \( \Phi \) which is linear in the dimension of the input size \( T \) where \( \Phi = \Phi_1 \) with the first two features in matrix \( \Phi_1 \). Another sampled feature grouping matrix \( \Phi_2 \) (c) Average of \( \Phi_1^T \Phi_1 \) and \( \Phi_2^T \Phi_2 \); i.e. estimated \( \Omega \) (d) Variance of \( \Phi_k \); i.e. estimated variance of \( \Delta \).

For feature dropout regularization, \( \Omega = I \) and \( \Delta \) is a diagonal matrix where each \( i \)-th diagonal term has \( \mathbb{E}[\Delta_i] = 0 \) and \( \mathbb{E}[\Delta_i^2] = \delta / (1 - \delta) \) where \( \delta \) is the dropout probability. Assuming \( \mathbb{E}[\Delta_i \Delta_j] = 0 \) for \( i \neq j \), it can be written as:

\[
\text{Var}_{\Phi_k} \left[ x_i^T \Phi^T \Phi_k \beta \right] = \frac{\delta}{1 - \delta} \sum_{j=1}^{p} x_{ij}^2 \beta_j^2
\]

where \( x_{ij} \) is the \( j \)-th entry of \( x_i \). For linear regression, this is equivalent to ridge regression after orthogonalizing the features. However, for feature grouping, the matrix \( \mathbb{E}[\Delta x_i x_i^T \Delta] \) rescales the feature vector \( x_i \) by the variance of each feature belonging to a cluster. For instance, if a feature consistently appears in a certain cluster, then the variance for that 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 \( \text{Var}_{\Phi_k} \left[ x_i^T \Phi^T \Phi_k \beta \right] \).

We illustrate this phenomenon in Figure 1 with a toy example. Let us assume \( \Phi_1 \) and \( \Phi_2 \) are two feature grouping matrices in our bank of projection matrices. Notice that the third feature appears with the first two features in matrix \( \Phi_1 \) whereas it appears with the last two features in \( \Phi_2 \). Figure 1(c) shows the average of \( \Phi_1^T \Phi_1 \) (where \( i \in \{1, 2\} \) which captures the general topography of the groups. Figure 1(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 \( \text{Var}_{\Phi_k} \left[ x_i^T \Phi^T \Phi_k \beta \right] \) in \( R(\beta) \) while still optimizing the objective using global features represented by \( \Omega \) via the loss function \( L(\Omega x_i, y_i; \beta) \) and the term \( \lambda'' \langle x_i^T \Omega \beta \rangle \) in \( R(\beta) \).

### 2.4 Computational Complexity

The computational complexity of logistic regression, solved with stochastic regularizer using feature grouping breaks down in four parts: i) computation of the bank of \( \Phi \) matrices (Step 1 in Algorithm 1) ii) multiplication by \( \Phi \) in summation in Step 6 iii) computation of gradient in Step 6 and iv) update in Step 7. The computational complexity of computing each \( \Phi \) using ReNA is \( O(rp \log (p/k)) \) \[16\]. Since the bank has \( b \) such matrices, the total computational complexity of computing the bank is \( O(brp \log (p/k)) \). This is a constant factor independent of the number of epochs. Computing \( \Phi x_i \) where \( \Phi \) is of dimension \( p \times k \) would be \( O(kp) \). However, as \( \Phi \) is sparse, this reduces to \( O(p) \). Since there are \( m \) samples in a minibatch, the total computational complexity is \( O(mp) \). Computational complexity of gradient computation for a row of \( \tilde{W} \) for a given sample \( x_i^{(i)} \) is \( O(k) \). Computing the gradient across all rows and samples in a minibatch has complexity \( O(lmk) \).

Updating one row of \( \tilde{W} \) requires right multiplying the gradient by \( \Phi^T \) which would be \( O(kp) \), but due to the sparse structure of \( \Phi \), it reduces to \( O(p) \). As there are \( l \) rows, the total computational complexity for update is \( O(lp) \). Projection, gradients computation and update are done for each epoch, so the computational complexity for the full iteration would be \( O(TM lp + Tlm k + Tl p) \) where \( T \) is the total number of epochs. Hence the total computational complexity of logistic regression with ReNA can be written as:

\[
O(brp \log (p/k) + Tm lp + Tlm k + Tlp)
\]

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 \( O(Tlmp) \).
3 Empirical results

3.1 Olivetti faces dataset

The Olivetti data set consists of grayscale $64 \times 64$ face images from 40 subjects [15]. 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 use 33% of 400 images as test and the rest as the training data set. This dataset is an instance of a high-dimensional small-sample problem. In order to explore robustness of classification approaches against noise, we added zero-mean Gaussian noise with varying standard deviations: 0.1, 0.2, 0.3, 0.4 and 0.5. We repeated each experiment with 10 different seeds. We report the results with average and standard error computed from 10 seeds.

Experiments with feature dropout: We also benchmark the standard dropout approach. We applied feature dropout with probabilities of 0.05, 0.1, 0.2 and 0.3. The best performance was achieved when the dropout probability is 0 at all noise levels as shown in Figure 2(a) (detailed results in Figure 6) indicating that feature dropout does not improve the performance. For the rest of the analysis, we use feature dropout with $p = 0.05$ for comparison with feature grouping and $\ell_2$ regularizer.

Experiments with an $\ell_2$ regularizer: We applied an $\ell_2$ regularizer with penalty parameters 0.0001, 0.001, 0.01, 0.1, 1, 10. The best performance was achieved when the regularization parameter $\lambda$ was 0.0001 as shown in Figure 2(b) (detailed results at different noise levels in Figure 7).

Varying $\ell_2$ regularization in addition to feature grouping: We applied feature grouping to reduce the dimension where the clusters were learned from the training data and then applied $\ell_2$ regularizer with the penalty parameters 0.0001, 0.001, 0.01, 0.1, 1, 10. In other words, feature grouping is used as pre-processing and not included in training with SGD. The best performance was achieved with $\lambda = 0.0001$ at all noise levels as shown in Figure 2(c) (detailed results at different noise levels in Figure 8).

Comparison of feature grouping with the feature dropout and $\ell_2$ regularizers: We compare feature grouping against other regularizers with the hyper-parameters that achieved the best performance. Figures 3(b) shows the comparison at noise level $\sigma = 0.5$ as a function of computation time. Feature grouping achieves higher accuracy sooner than the other regularizers. Figures 10 and 11 show...
similar comparison for other noise levels as a function of number of epochs and computation time, respectively. Figure 3(a) displays the performance comparison as a function of the standard deviation of the noise. In order to gain insights for the learning behavior, we visualize the learned weights in Figure 4 for a single Olivetti face (class) with noise level $\sigma = 0.5$. The learned weights from feature grouping demonstrate more structure than the other methods despite the high level of noise. We illustrate ReNA clustering and reconstruction using those clusters on a few samples from the Olivetti dataset in Figure 9. Empirically, feature-grouping regularization is not very sensitive to the choice of $b$, the number of projection matrices $\Phi$ and $r$, the number of samples used to compute each $\Phi$ (Table 3). We use $r = 20$, $b = 1000$. When the noise is very large, prediction performance degrades slowly as the number of clusters $k$ increases (Table 4 and Figure 14). We set $k$ to 819, 20% of total number of features, as this is a choice well suited to a wide set of situations [16].

3.2 Neuroimaging Dataset from Human Connectome Project

The Human Connectome Project (HCP) consortium has released a large openly-accessible fMRI dataset. In this study 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 general linear model (GLM) was fitted to each fMRI dataset [5]. 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 in our work as described in Table 1. Note that these tasks are different than those that were used by Bzdok et al. [11]. We made the classification problem more challenging by choosing several stimuli from the same cognitive paradigm. fMRI data are resampled to 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. We used logistic regression as a classification algorithm. Figure 13 shows an example of clusters extracted from feature averaging for 100 randomly selected samples of the fMRI data used in this study. Figure 13(a) shows an original fMRI map for a given subject and a cognitive task and Figure 13(b) shows its approximation using the clusters shown in Figure 13(c). We randomly selected 100 subjects for test and the rest 400 subjects for training from the HCP data set. The test data includes 1 964 samples with at least 95 samples from each target class whereas the training set has 7 785 samples.

Each feature grouping matrix is computed over $r = 100$ randomly picked samples from the training set, yielding $k = 27 080$ clusters. For each epoch during training a projection matrix was randomly selected.
picked from the bank of $b = 500$ feature grouping matrices. We used ADAM [17] with learning rate 0.0001 as a stochastic optimizer where the exponential decay rate parameter for the first moment is set to 0.9; for the second moment it is set to 0.999. To rule out convergence issues, we performed 2000 passes over the training data with mini-batches of 250 samples. Convergence curves show that this number of epochs is likely an overkill in practical settings. We repeated the experiments for 20 different seeds for training and report the average and standard deviation of the performances on test data. We used Python 3.5 for implementation [26] using open-source libraries scikit-learn [27], nibabel [9], nilearn [11], joblib [36] and numpy [39].

We compared our approach with feature dropout method and the standard $\ell_2$ regularizer approach where no stochastic regularizer is used. Table 2 gives the average accuracy at convergence, along with standard error of the mean for various regularizers. The parameter $\lambda$ represents the $\ell_2$ penalty and $p$ represents the dropout probability, respectively. A choice of 1 appears as the best value for the $\ell_2$ regularization parameter. Similarly, feature dropout regularizer with probability 0.5 performs the best among regularizers with feature dropout.

Yet, the best performing regularization scheme was feature grouping, as also highlighted in the table. We also plotted the learning curves for accuracy for different regularizers in Figure 5 where feature grouping also reaches good accuracy faster. Note that we applied a smoothing filter only for visualization purposes. As it can be seen both in Table 2 and Figure 5 feature grouping outperforms the alternative approaches for the neuroimaging dataset.

### 4 Discussion and Conclusion

We introduced a stochastic regularizer that captures the structure of data, e.g. neighboring pixels of an image. The regularizer is based on feature averaging, randomized inside a stochastic gradient descent. Structured regularization is central to high-dimensional small-sample problems. We focused analysis and experiments on logistic regression, since the regularizer only affects the feature domain. However, this approach can be applied to any architecture trained by stochastic gradient descent, including deep neural networks. On very high-dimensional data, architectures are likely to comprise a bottleneck layer after the input, hence high-dimensional structured regularization should be necessary only for the input layer.

On a faces dataset with a controlled amount of noise, we showed that this structured regularization becomes very beneficial in strong noise settings. This result carries over to real-world high-dimensional and noisy data, namely brain imaging which often suffers from overfitting. Experiments indicate that feature grouping in an SGD leads to good prediction accuracy in less epochs and shorter time than $\ell_2$ and feature dropout regularizers, even for low noise levels, where the benefit to final prediction accuracy is minor. We also showed that the learned weights have more structure with feature grouping.

On real world neuroimaging, the accuracy improvement is small, yet significant. Other regularization approaches also add only marginal improvements with respect to the un-regularized estimator. The reason for this might be the high dimensional and noisy structure of the fMRI data set. Another interesting observation was that while feature dropout performed reasonably well for the neuroimaging dataset, it performed poorly for the faces dataset. This might be related to the different noise structures of the datasets. Inherent noise in fMRI dataset is likely to be different from the additive Gaussian noise we embedded to Olivetti dataset in our simulations.

Unlike the feature dropout method, our approach for stochastic regularization leverages the image structure and takes statistical organization of the data into account. We observed that this approach increases the performance of the statistical estimations in both datasets used in this study. Our approach has linear computational complexity with respect to the dimension of the data, as opposed to traditional structured penalties that require an inner loop for each gradient step on the loss [4].

The approach proposed here points to natural future developments. Our findings suggest that the use of structured random matrices in regularization should be more explored. We gave insights on regularization with randomized feature grouping, but more comprehensive analysis using the statistical properties of clustering matrices would be useful. Beyond logistic regression, the approach should be applied to more general neural-network architectures and tested on other data that present a strong stationary structure, such as NMR spectra in chemistry. Last, beyond ReNA, other clustering
approaches could be used, including other agglomerative approaches [23], or density-based methods. Finally, fitting complex models has been a challenge so far in Neuroimaging applications: linear models are still the most used, and often the most performant. We hope that such structured regularization, combined with more expressive models [11] will help extract more information from neuroimaging data.

References

[1] Alexandre Abraham, Fabian Pedregosa, Michael Eickenberg, Philippe Gervais, Andreas Mueller, Jean Kossaifi, Alexandre Gramfort, Bertrand Thirion, and Gaël Varoquaux. Machine learning for neuroimaging with scikit-learn. Frontiers in neuroinformatics, 8:14, 2014.

[2] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. IEEE transactions on pattern analysis and machine intelligence, 34(11):2274–2282, 2012.

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

[4] Francis Bach, Rodolphe Jenatton, Julien Mairal, and Guillaume Obozinski. Structured sparsity through convex optimization. Statistical Science, pages 450–468, 2012.

[5] Deanna M Barch, Gregory C Burgess, Michael P Harms, Steven E Petersen, Bradley L Schlaggar, Maurizio Corbetta, Matthew F Glasser, Sandra Curtiss, Sachin Dixit, Cindy Feldt, et al. Function in the human connectome: task-fmri and individual differences in behavior. Neuroimage, 80:169–189, 2013.

[6] Ella Bingham and Heikki Mannila. 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, pages 245–250. ACM, 2001.

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

[8] Xavier Bouthillier, Kishore Konda, Pascal Vincent, and Roland Memisevic. Dropout as data augmentation. arXiv preprint arXiv:1506.08700, 2015.

[9] Matthew Brett, Michael Hanke, Ben Cipollini, Marc-Alexandre Côté, Chris Markiewicz, Stephan Gerhard, Eric Larson, Gregory R Lee, Yaroslav Halchenko, Erik Kastman, et al. nibabel: 2.1.0. Zenodo, 2016.

[10] Peter Bühlmann, Philipp Rütimann, Sara van de Geer, and Cun-Hui Zhang. Correlated variables in regression: clustering and sparse estimation. Journal of Statistical Planning and Inference, 143(11):1835–1858, 2013.

[11] Danilo Bzdok, Michael Eickenberg, Olivier Grisel, Bertrand Thirion, and Gaël Varoquaux. Semi-supervised factored logistic regression for high-dimensional neuroimaging data. In Advances in neural information processing systems, pages 3348–3356, 2015.

[12] Timothy I Cannings and Richard J Samworth. Random-projection ensemble classification. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 79(4):959–1035, 2017.

[13] Robert J Durrant and Ata Kabán. Random projections as regularizers: Learning a linear discriminant ensemble from fewer observations than dimensions. 2013.

[14] Miguel García-Torres, Francisco Gómez-Vela, Belén Melián-Batista, and J Marcos Moreno-Vega. High-dimensional feature selection via feature grouping: A variable neighborhood search approach. Information Sciences, 326:102–118, 2016.

[15] A HOPPER. The orl face database. at&t (olivetti) research laboratory cambridge, 1992.
[16] Andrés Hoyos-Idrobo, Gaël Varoquaux, Jonas Kahn, and Bertrand Thirion. Recursive nearest agglomeration (rena): fast clustering for approximation of structured signals. *arXiv preprint arXiv:1609.04608*, 2016.

[17] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[18] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[19] Nick Lim and Robert J Durrant. Linear dimensionality reduction in linear time: Johnson-lindenstrauss-type guarantees for random subspace. *arXiv preprint arXiv:1705.06408*, 2017.

[20] Andrew McCallum, Kamal Nigam, and Lyle H Ungar. 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*, pages 169–178. ACM, 2000.

[21] Charles E McCulloch and John M Neuhaus. *Generalized linear mixed models*. Wiley Online Library, 2001.

[22] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[23] Daniel Müllner. Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378*, 2011.

[24] Benson Mwangi, Tian Siva Tian, and Jair C Soares. A review of feature reduction techniques in neuroimaging. *Neuroinformatics*, 12(2):229–244, 2014.

[25] Steven J Nowlan and Geoffrey E Hinton. Simplifying neural networks by soft weight-sharing. *Neural computation*, 4(4):473–493, 1992.

[26] Travis E Oliphant. Python for scientific computing. *Computing in Science & Engineering*, 9(3), 2007.

[27] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[28] Sergey M Plis, Devon R Hjelm, Ruslan Salakhutdinov, Elena A Allen, Henry J Bockholt, Jeffrey D Long, Hans J Johnson, Jane S Paulsen, Jessica A Turner, and Vince D Calhoun. Deep learning for neuroimaging: a validation study. *Frontiers in neuroscience*, 8, 2014.

[29] Mostafa Rahmani and George Atia. Data dropout in arbitrary basis for deep network regularization. *arXiv preprint arXiv:1712.00891*, 2017.

[30] Salah Rifai, Xavier Glorot, Yoshua Bengio, and Pascal Vincent. Adding noise to the input of a model trained with a regularized objective. *arXiv preprint arXiv:1104.3250*, 2011.

[31] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[32] Heung-Il Suk, Chong-Yaw Wee, Seong-Whan Lee, and Dinggang Shen. State-space model with deep learning for functional dynamics estimation in resting-state fmri. *NeuroImage*, 129:292–307, 2016.

[33] Anbupalam Thalamuthu, Indranil Mukhopadhyay, Xiaojing Zheng, and George C Tseng. Evaluation and comparison of gene clustering methods in microarray analysis. *Bioinformatics*, 22(19):2405–2412, 2006.
[34] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[35] David C Van Essen, Stephen M Smith, Deanna M Barch, Timothy EJ Behrens, Essa Yacoub, Kamil Ugurbil, WU-Minn HCP Consortium, et al. The wu-minn human connectome project: an overview. *Neuroimage*, 80:62–79, 2013.

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

[37] Nguyen Xuan Vinh, Sarah Erfani, Sakrapee Paisitkriangkrai, James Bailey, Christopher Leckie, and Kotagiri Ramamohanarao. Training robust models using random projection. In *Pattern Recognition (ICPR), 2016 23rd International Conference on*, pages 531–536. IEEE, 2016.

[38] Stefan Wager, Sida Wang, and Percy S Liang. Dropout training as adaptive regularization. In *Advances in neural information processing systems*, pages 351–359, 2013.

[39] Stéfan van der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.

[40] Rui Xu and Donald Wunsch. Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–678, 2005.

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

[42] Peng Zhao, Guilherme Rocha, and Bin Yu. The composite absolute penalties family for grouped and hierarchical variable selection. *The Annals of Statistics*, pages 3468–3497, 2009.
| Cognitive Task | Stimuli | Description |
|----------------|---------|-------------|
| **Working Memory** | 2BK-0BK BODY-AVG FACE-AVG PLACE-AVG TOOL-AVG | remembering two pictures back versus the current one presented body parts versus other visual objects presented faces versus other visual objects presented places versus other visual objects presented tools versus other visual objects |
| **Gambling** | PUNISH REWARD | loss trials when asked to guess a range of a number. reward trials when asked to guess a range of a number. |
| **MOTOR** | LF-AVG LH-AVG RF-AVG RH-AVG T-AVG | Left foot movement versus other movements Left hand movements versus other movements Right foot movements versus other movements Right hand movements versus other movements Tongue movements versus other movements |
| **Language** | MATH STORY | complete addition and subtraction problems asked questions about topic of the story |
| **Relational** | MATCH REL | decide if objects match find differences between objects |
| **Emotion** | FACES SHAPES | decide which two faces match decide which two shapes match |
| **Social** | RANDOM TOM | presented video clips where objects moved randomly presented video clips where objects interacted |

Table 1: Cognitive tasks and contrasts used for supervised classification for the neuroimaging data set

| Regularizer | Acc |
|-------------|-----|
| None        | 0.931 ± 0.0011 |
| \( \ell_2 \) with \( \lambda = 1 \) | 0.932 ± 0.0011 |
| \( \ell_2 \) with \( \lambda = 10 \) | 0.931 ± 0.0012 |
| \( \ell_2 \) with \( \lambda = 100 \) | 0.923 ± 0.0017 |
| \( \ell_2 \) with \( \lambda = 1000 \) | 0.896 ± 0.0030 |
| Feature Grouping | 0.938 ± 0.0010 |
| Feature Grouping + \( \ell_2 \) with \( \lambda = 10 \) | 0.937 ± 0.0020 |
| Feature Grouping + \( \ell_2 \) with \( \lambda = 100 \) | 0.927 ± 0.0026 |
| Feature Dropout with \( p = 0.1 \) | 0.933 ± 0.0011 |
| Feature Dropout with \( p = 0.3 \) | 0.933 ± 0.0010 |
| Feature Dropout with \( p = 0.5 \) | 0.935 ± 0.0010 |

Table 2: Average and standard deviation of accuracy results for different regularizers for the Neuroimaging dataset

5 Appendix

Figure 6: Accuracy performance comparison of feature dropout regularizer with different dropout probabilities for varying noise level of Olivetti face dataset.
Figure 7: Accuracy performance comparison of L2 regularizer with different regularizer parameter $\lambda$ for varying noise level of Olivetti face dataset.

Figure 8: Accuracy performance comparison of L2 regularizer with different regularizer parameter $\lambda$ after reducing dimension with feature grouping for varying noise level of Olivetti face dataset.

$\sigma = 0.3$

$\sigma = 0.5$

Table 3: Average and standard deviation of accuracy results of feature grouping for Olivetti dataset with $\sigma = 0.5$ for different $r$ and $b$ values.

| $r$ : number of samples used for each $\Phi$ | $b$ : number of projection matrices | Accuracy       |
|--------------------------------------------|-----------------------------------|---------------|
| 50                                         | 100                               | 0.756 ± 0.010 |
| 50                                         | 500                               | 0.758 ± 0.013 |
| 50                                         | 1500                              | 0.757 ± 0.013 |
| 100                                        | 100                               | 0.747 ± 0.013 |
| 100                                        | 500                               | 0.752 ± 0.013 |
| 100                                        | 1500                              | 0.756 ± 0.012 |
| 200                                        | 100                               | 0.729 ± 0.013 |
| 200                                        | 500                               | 0.730 ± 0.012 |
| 200                                        | 1500                              | 0.731 ± 0.011 |
| 20                                         | 1000                              | **0.763 ± 0.012** |

Figure 9: (a) Original and reconstructed images randomly sampled from the test dataset of Olivetti faces (b) Clusters extracted from training dataset using feature grouping with $k = 819$ (20 % of number of features).
Figure 10: Performance comparison of feature grouping against the other regularizers using best parameters, for varying noise level of Olivetti face dataset as a function of computational time.

Figure 11: Performance comparison of feature grouping against the other regularizers using best parameters, for varying noise level of Olivetti face dataset as a function of epochs.

| $k$                  | Accuracy  | Total Training Time (sec) |
|----------------------|-----------|---------------------------|
| 40 (1% of features)  | 0.762 ± 0.01 | 1331                     |
| 204 (5% of features) | 0.821 ± 0.01 | 1963                     |
| 409 (10% of features)| 0.803 ± 0.01 | 2049                     |
| 819 (20% of features)| 0.763 ± 0.01 | 2340                     |
| 1638 (40% of features)| 0.647 ± 0.01 | 2810                     |
| 3276 (80% of features)| 0.447 ± 0.01 | 4148                     |

Table 4: Average and standard deviation of accuracy results of feature grouping for Olivetti dataset with $\sigma = 0.5$ for different numbers of clusters ($k$).
Figure 12: Visualization of the weights learned for $\sigma = 0.5$ for sampled Olivetti faces.

Figure 13: (a) Original fMRI map of a random subject while performing right foot movement (RF-AVG in table 1) (b) Compressed fMRI map of the same subject (the clusters were computed on a different data set) (c) Clusters extracted from randomly selected 100 fMRI samples in the dataset using feature grouping with $k = 54,161$ (20% of the number of features).

Figure 14: Average and standard deviation of accuracy results of feature grouping for Olivetti dataset with $\sigma = 0.5$ for different amount of dimensionality reduction (results in Table A).