FReM – scalable and stable decoding with fast regularized ensemble of models
Andrés Hoyos-Idrobo, Gaël Varoquaux, Yannick Schwartz, Bertrand Thirion

To cite this version:

HAL Id: hal-01615015
https://hal.archives-ouvertes.fr/hal-01615015
Submitted on 11 Oct 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
FReM – scalable and stable decoding with fast regularized ensemble of models

Andrés Hoyos-Idrobo\textsuperscript{a,b,}*, Gaël Varoquaux\textsuperscript{a,b}, Yannick Schwartz\textsuperscript{a,b}, Bertrand Thirion\textsuperscript{a,b}

\textsuperscript{a}Parietal project-team, INRIA Saclay-Île de France
\textsuperscript{b}CEA/Neurospin bât 145, 91191 Gif-Sur-Yvette

Abstract

Brain decoding relates behavior to brain activity through predictive models. These are also used to identify brain regions involved in the cognitive operations related to the observed behavior. Training such multivariate models is a high-dimensional statistical problem that calls for suitable priors. State of the art priors –eg small total-variation– enforce spatial structure on the maps to stabilize them and improve prediction. However, they come with a hefty computational cost. We build upon very fast dimension reduction with spatial structure and model ensembling to achieve decoders that are fast on large datasets and increase the stability of the predictions and the maps. Our approach, fast regularized ensemble of models (FReM), includes an implicit spatial regularization by using a voxel grouping with a fast clustering algorithm. In addition, it aggregates different estimators obtained across splits of a cross-validation loop, each time keeping the best possible model. Experiments on a large number of brain imaging datasets show that our combination of voxel clustering and model ensembling improves decoding maps stability and reduces the variance of prediction accuracy. Importantly, our method requires less samples than state-of-the-art methods to achieve a given level of prediction accuracy. Finally, FreM is highly parallelizable, and has lower computation cost than other spatially-regularized methods.

Keywords: fMRI; supervised learning; decoding; bagging; MVPA

1. Introduction: decoding needs stability

Decoding models predict stimuli or behavior from brain images. These models have become a standard tool in neuroimaging data analysis (Haynes and Rees, 2006; Norman et al., 2006; Varoquaux and Thirion, 2014). In clinical applications, they can be used to perform diagnosis or prognosis (Demirci et al., 2008; Fan et al., 2008). They are also used as evidence of the link between distributed activity patterns and an observed behavior (Haxby et al., 2001). Additionally, decoding used on a large variety of cognitive processes grounds a form of reverse inference (Poldrack, 2011; Schwartz et al., 2013). An appeal of decoding procedures is that they avoid multiple voxel-wise test and perform an omnibus test: “Can one predict the behavioral outcome from brain activity?”

Identifying the brain activity patterns that drive prediction of behavior is crucial for brain mapping and understanding (Gramfort et al., 2013; Mourão-Miranda et al., 2005). However achieving reliable and stable decoder maps is challenging due to the dimensionality of the problem: the number of samples is small –hundreds or less– whereas the number of features is typically the number of voxels in the brain –up to hundreds of thousands. Linear models, e.g. linear support vector machines (SVM), are often used (Pereira et al., 2009), as they have shown a good performance in a small-sample regime. In addition, their classification/regression weights form brain maps used for interpretation of the discriminative pattern (Mourão-Miranda et al., 2005).

However, the high dimensionality of the problem leads to multiple weight maps yielding the same predictive power, and some form of regularization has to be applied (Hastie et al., 2000). In across-subject settings, complex spatial and sparse penalties such as total-variation (TV) (Baldassarre et al., 2012; Michel et al., 2011) and Graph-net (Grosenick et al., 2013) help the decoder to capture the important brain regions shared across subjects. TV and its variants are considered as the state-of-the-art regularizers for brain images, as they handle local correlations present in the data. The main drawback of spatially-structured sparsity as in TV and related penalties is their computational cost.

A much cheaper alternative to these structured estimators is to use spatially-constrained clustering algorithms to perform voxel grouping. In decoding, voxel grouping is often used as part of the pipeline for stability selection of correlated voxels (Gramfort et al., 2012; Varoquaux et al., 2012; Wang et al., 2015). Additionally, it helps to improve the conditioning of the estimation problem. However, voxel grouping introduces high bias, as the patterns are constrained by the clusters shape.

One way to mitigate this bias is to use model aggregation or ensembling. These approaches have been
used to reduce the variability of the output of the decoder (Kuncheva and Rodriguez, 2010; Kuncheva et al., 2010a; Zhou, 2012). The central idea is to build a decoder by averaging the output of several “good” models. In particular, averaging linear models boils down to averaging weight maps. One way to estimate multiple models is to use bootstrap resampling to generate different training sets to fit the decoder, and then aggregate them. This approach is known as Bagging\(^1\) (Breiman, 1996). It is easy to run in parallel, training each model independently. Yet, naïve application of bagging to neuroimaging data induces high computational cost as the data are high dimensional, and parameters have to be set by internal cross-validation.

Decoding calls not only for hyperparameter selection, but also for model validation. Both tasks require a measure of the predictive power of the decoder. In practice, one runs two cross-validation loops –one inside the other– where each loop assesses prediction accuracy respectively for model selection and validation. Thus, investigators often train the decoder many times. These repeated calculations entail computational costs that limit day-to-day work on standard workstations. This is particularly problematic for more advanced decoders such as those with spatial regularizations that are beneficial to neuroimaging data (e.g. Grosenick et al., 2013; Michel et al., 2011; Mohr et al., 2015). In the face of growing data size, to enable good validation and ease of use on most hardware, a good decoder should be sparing on computation resources.

**Contributions.** Here, we propose a fast scheme to train regularized ensembles of models, FReM. It reduces the variance of the weight maps of the decoder, while ensuring high prediction accuracy. The core of this approach is to average the estimator with the best predictive power per loop inside the nested cross-validation. To benefit from spatial regularization while keeping fast run times, we show how an optional voxel-clustering can be included in the ensembling, bringing stable spatial patterns. We perform a series of classification experiments on several MRI datasets to demonstrate that ensembling regularized models gives state-of-the-art decoders. In particular, we show that they compare favorably to existing decoders in terms of prediction performance, weight-map stability, and computation time.

2. **Background and prior art**

2.1. **Brain decoding**

In neuroimaging, a decoder is a predictive model that, given \(n\) brain images, fits an external variable \(y\). In practice, we arrange \(n\) observed brain images composed of \(p\) voxels in a matrix \(X \in \mathbb{R}^{n \times p}\). Linear predictive models, at the core of most decoders in neuroimaging, are then written (Hastie et al., 2009):

\[
y = f(Xw + \epsilon),
\]

where \(y\) denotes a target variable giving the experimental condition or health status of subjects, \(f\) represents the decision function in the classification; \(w \in \mathbb{R}^p\) denotes the weight vector/map, and \(\epsilon \in \mathbb{R}^n\) is a random error term.

In spite of a recently growing effort on the accumulation of neuroimaging data (Poldrack and Gorgolewski, 2015), the number \(n\) of samples per-class remains in the order of a few hundreds, whereas \(p\) can be hundreds of thousands of voxels (\(p \gg n\)). In this high-dimensional setting, there are many equivalent solutions and some form of regularization or prior is necessary to restrict model complexity. A standard approach relies on solving the following optimization problem:

\[
\hat{w}(\lambda) = \arg \min \{ \mathcal{L}(y, X; w) + \lambda \Omega(w) \}, \quad \lambda > 0,
\]

where \(\mathcal{L}\) is a data-fidelity term, a loss function that measures the quality of the estimator (e.g. logistic or hinge loss); \(\Omega\) denotes the penalty/regularization term, and \(\lambda\) is the parameter that controls the amount of regularization.

Two of the most often used penalties are: 1) the \(\ell_2\)-norm, that penalizes large \(w\) coefficients, and yields non-sparse solutions; 2) the \(\ell_1\)-norm, that promotes a small number of non-zero \(w\) coefficients, and yields sparse solutions (Tibshirani, 1994).

Nevertheless, as neuroimaging data exhibit strong correlations between the columns of \(X\), the \(\ell_1\)-penalty yields unstable solutions as it tends to arbitrarily select only one among the correlated variables (Varoquaux et al., 2012; Yu, 2013). One way to tackle this is the use of additional spatially-informed penalties as Graphnet (Grosenick et al., 2013) or TV (total variation) (Eickenberg et al., 2015; Michel et al., 2011).

2.2. **Model validation and selection**

In high-dimensional settings, the number of candidate models is much larger than the number of samples. Therefore, we use regularization to constrain the complexity of the solution, and this penalization is controlled by the \(\lambda\) regularization parameter. The ensuing problem is then to find an optimal value for \(\lambda\) (i.e. finding the best bias-variance trade-off), yielding a model that exploits the richness of the data. One typically uses the predictive power of the decoder to choose the right amount of regularization.

**Hyperparameters selection.** In general, the setting of the hyperparameter is a data-specific choice, as it is governed by the amount of data and their signal-to-noise ratio (SNR). The most common approach to set it is to use cross-validation to measure the predictive power for various amounts of regularization and retain the value that maximizes the predictive power across several cross-validation folds (Varoquaux et al., 2017). To assess predictive power in addition, the standard scheme is *nested...*
cross-validation, that consists of two cross-validation loops run one inside the other: an outer loop is used to assess the predictive power of the decoder, and an inner/nested loop is used to set the hyperparameter(s) (see Fig. 1): the train set is used to fit the decoder, while the test and validation sets are used to measure its ability to generalize to new data.

In most of the non-parametric approaches to select a regularization parameter, a suitable and finite set of $l$ hyperparameters, $\lambda \in [\lambda_1, \ldots, \lambda_l]$ is first defined. For each cross-validation fold, one fits the decoder with all hyperparameters, and measures their prediction error on the test set. Then, one choses the $\lambda_i$ value that maximizes the predictive power across folds.

2.3. Ensembling methods for better stability

Another desired characteristic of decoding algorithms is the stability to data perturbations. Stability is defined as the amount of change in the output of the decoder as a function of small changes in the data on which the decoder is trained (Shalev-Shwartz et al., 2010). These small changes or perturbations usually mean either deleting one example or replacing it with another one. Stability implies that small variations in the data yield a commensurate variation in the prediction and the weight map of the decoder.

In neuroimaging, another approach to improve the stability of the estimators is to train the base estimator on several random partitions of the feature space, then select representative features according to a consensus. These partitions can be defined using various criteria, for instance: i) random voxels selection (Kuncheva et al., 2010b; Rondina et al., 2014); ii) using clustering (Varoquaux et al., 2012). Yet, these decoders need to fit more models, to accumulate selection statistics, and hence entail excessive computational costs given the number of models to fit.

Model averaging. Model selection can run into some issues due to instability in the choice of the model, as any perturbation of the original data entails the selection of a completely different hyperparameter (Arlot and Celisse, 2010). Model averaging mitigates this problem by aggregating the output of several suitable models (Nemirovski, 2000). This method improves the predictive power of the base estimator, reducing the variance if the models are sufficiently uncorrelated (see Appendix A).

In particular, averaging linear models boils down to:

$$\hat{w}_{bagg}(\lambda) = \frac{1}{b} \sum_{j=1}^{b} \hat{w}^{(j)}(\lambda),$$

where $b$ denotes the number of models to average; it is often chosen as 50 or 100, depending on the sample size and on the computation cost to train the estimator $\hat{w}$ (Bühlmann and Yu, 2002). The weight map of the bagged estimator\(^3\) displays less variance if the weight maps of the base estimators are sufficiently uncorrelated. Note that $\lambda$ has to be set, requiring another nested loop of cross-validation (see Fig. 2a). Hence, choosing the parameter of the aggregated model is computationally expensive.

3. Materials and methods

3.1. Dimension reduction by feature agglomeration

In neuroimaging, dimension reduction is routinely used to alleviate problems due to high-dimensionality. A common way to select features is univariate feature screening, which uses a score (e.g. statistical test, correlation) to remove non-predictive variables. In particular, the method proposed by Doehmatob et al. (2015) operates as follows: The data $(X, y)$ are standardized so that $y$ has unit variance and zero mean, likewise each row of the design matrix $X$. Additionally, $X$ is smoothed with a Gaussian filter for the screening of voxels, but not during the fitting of the estimator. Then, for each voxel $j$ one computes the absolute dot-product of $y$ and the $j$th column of $X$, $|X^j y|$. Finally, one selects the voxels with the highest $|X^j y|$ values.

However, this method does not take into account the spatial structure of brain images. Instead, we can reduce the dimension of the data by grouping similar neighboring voxels, moving from the voxel-space to a parcel-space. To do this, we can use anatomical/functional atlases or data-driven approaches.

Here we rely on a voxel grouping approach, where we use a fraction of the training data to train a clustering algorithm, finding suitable groups of features or parcellations. Then, we use these parcels on the remaining data to work at a parcel level. Formally, we define a feature-grouping matrix $\Phi \in \mathbb{R}^{p \times k}$, where $k \ll p$, and each column has a constant value with support at each parcel. We normalize each column to have unit $\ell_2$-norm (Hoyos-Idrobo et al.,

\[^3\]The bagged estimator is a Monte-Carlo approximation of $E[w]$, \[^4\]The feature-grouping matrix is orthogonal.
response inhibition on openfMRI ds009 (Poldrack et al., 2013). We use the trial-by-trial (Z-score) maps computed in a first-level GLM to perform all across-subject predictions. Additionally, we predict the gender from VBM maps using the OASIS dataset (Marcus et al., 2007).

Standard preprocessing and first-level analysis were applied using SPM. The data were variance-normalized and spatially smoothed at 6 mm FWHM for fMRI data and 2 mm FWHM for VBM data.

**Experimental setup.** In all classification tasks, we use nested cross-validation for an accurate measure of the predictive power. We repeatedly split the data into a validation set and a decoding set. We choose validation sets of 20% of the data, respecting the sample dependence structure (leaving out subjects or sessions). We set 10 folds for the outer cross validation loop.

As is standard practice in fMRI decoding (Pereira et al., 2009), we use univariate feature selection on the training set to select 20% of voxels and train the decoder on the selected features. We compare several decoders, split into two groups:

- **Non-ensembles:** Graph-net (Grosenick et al., 2013), TV-$\ell_1$ (Michel et al., 2011), Log-enet (Zou and Hastie, 2005)$^6$, SVM-$\ell_1$, and SVM-$\ell_2$.

- **Ensembles:** FReM of SVM-$\ell_1$, SVM-$\ell_2$, both, with and without clustering. These estimators are fitted using the proposed scheme—see Algorithm 1.

We use scikit-learn (Pedregosa et al., 2011) for the Log-enet, and the SVM with $\ell_1$ and $\ell_2$ penalties. We use nilearn (Abraham et al., 2014) for Graph-net and TV-$\ell_1$. When clustering is applied, we set the number $k$ of clusters to 10% of the number $p$ of voxels$^7$. We rely on the fast agglomerative clustering presented in Hoyos-Idrobo et al. (2016). In brief, this algorithm iteratively performs 1-nearest neighbor grouping, reduces the graph at each iteration, then averages the input features and repeats the process until it reaches the desired number of clusters.

Regarding the ensembles of models, we use 50% of decoding sets to train the decoder$^8$.

In the first experiment, we empirically validate the performance of various decoders on different discrimination tasks. In particular, we measure the prediction score,...

---

$^5$This approximation can be seen as the application of an anisotropic smoothing.

$^6$We don’t use an SVM with elastic net penalty, as none of the solvers have an implementation, and using our own implementation would lead to unfair comparisons.

$^7$We consider a useful dimension reduction range, $k \in \left[ \frac{p}{20}, \frac{p}{4} \right]$. This regime gives a good trade-off between computational efficiency and data fidelity (Hoyos-Idrobo et al., 2016).

$^8$In the standard bootstrap the whole dataset is resampled. However, it can be approximated with a subsampling of 50% of the data (Dümbgen et al., 2013; Præstgaard and Wellner, 1993; Shah and Samworth, 2013)
computation time, and the correlation across folds (stability). In a second experiment, we explore the training speedup of decoders in a multi-core setting. Then we evaluate the similarity between weight maps obtained using all the data and the ones obtained for different sample sizes (small-sample recovery).

5. Results: evaluating decoder performance

5.1. Comparing FReM to bagging

We explore the performance of FReM and bagging on two discriminative tasks: discrimination of famous and scrambled faces from the Henson (2006) dataset, and discrimination of response inhibition on openfMRI ds009 (Foldrack et al., 2013). The regularization parameters of bagging are set by 10-fold inner cross validation.

Fig. 3 summarizes the relative performance of FReM and bagging with respect their base estimators across discriminative tasks. FReM of $\ell_2$-penalized models with and without clustering display better weight map stability than bagging of SVM-$\ell_2$. FReM of $\ell_1$-penalized models without clustering has worst prediction accuracy, with a loss of 2\% with respect to bagging. Regarding FReM with clustering, it has the same prediction accuracy as bagging, and it displays better weight map stability. For both penalties, FReM yields computation speedups, > 5 times faster, while preserving the gains in prediction accuracy.

5.2. Benchmarking decoders

For all discriminative conditions, we measure the prediction error on several left-out validation sets to assess the predictive power of the decoders. Additionally, we measure the correlation between the weight maps obtained
in each cross-validation fold, and the computation time required to train the decoder. To perform this analysis, we separate the datasets into two types: within-subject and across-subject. Throughout this experiment, we set the number $b$ of estimators used in FReM to 50. This choice is discussed in Fig. 8.

Fig. 4 summarizes the relative performance with respect to the mean across decoders per discriminative task. In within-subject settings, all sparse methods have good prediction performance. Decoding using the standard SVM with both $\ell_1$ and $\ell_2$ penalty is fast, but the weight maps are less stable than the ones found by sparse structured methods – i.e. Graph-net and TV-$\ell_1$. However, these complex penalties come with higher computation costs. As expected, using FReM reduces the variance of the prediction, while increasing the stability of the weight maps. This effect is enhanced when including a clustering step. The computation time of FReM with or without clustering is less than that of structured sparse classifiers.

For the discriminative task across subjects, FReM consistently improves prediction accuracy as well as the stability of the weight maps, while keeping a computation cost less than structured sparse classifiers. In addition, the use of spatial clustering has a beneficial impact on the spatial stability. In all the presented cases, FReM improves stability of the weight maps of the base estimator, while preserving the prediction accuracy. Note however that, for the combination of the SVM-$\ell_2$ and clustering, it does not display any additional benefit.

Table 1 shows the comparison between each decoder and the decoder displaying the best prediction accuracy, namely FReM of SVM-$\ell_1$ for within-subject problems, and FReM of SVM-$\ell_2$ with clustering for across-subjects problems. These results confirm the above observations.

Experiments on simulated data. Unlike brain imaging datasets, simulations open the door to measuring the actual support of a decoder, as well as its prediction accuracy. Briefly, we generate data with 2 classes and 1728 voxels with temporally auto-correlated Gaussian noise (more details in Appendix B). The dataset contains a decoding set of 200 samples. We choose validation sets of 20% of the data. We select 10 folds for the outer cross validation loop. We assess the support recovery of each decoder by building the precision-recall curve. This curve is generated by comparing the ground truth weight maps with the coefficients of the decoder after applying different thresholds.

Fig. 5 shows the prediction accuracy of each decoder across cross-validation folds on simulated data. FReM SVM-$\ell_1$ with and without clustering have the best prediction accuracy on simulated data, followed by SVM-$\ell_1$, and Graph-net. TV-$\ell_1$ display slightly worst performance. Log-enet, SVM-$\ell_2$, FReM SVM-$\ell_2$ with and without clustering have a low predictive performance. FReM of $\ell_1$ models increase the prediction accuracy, whereas for $\ell_2$ models this remains similar.

Regarding the computation time, SVM-$\ell_2$, FReM of SVM-$\ell_2$ with and without clustering are the fastest methods to train, followed by SVM-$\ell_1$, FReM of SVM-$\ell_1$ with and without clustering. Log-enet, and Graph-net are slightly slower, whereas TV-$\ell_1$ is at least 4 times slower than Graph-enet. On simulated data, FReM requires the same computation time as its base estimator.

Fig. 6 displays the precision-recall curve, which serves as an indicator of support recovery of the underlying spatial activation map. FReM of SVM-$\ell_1$ with clustering, FReM of SVM-$\ell_2$ with clustering, and TV-$\ell_1$ display the best support recovery. They are followed by Graph-net, SVM-$\ell_2$, and FReM SVM-$\ell_2$. In this experiment, Log-enet and SVM-$\ell_1$ both fail to recover the support of the activation signal. We can see that FReM consistently improves the support recovery of base estimators. In particular, FReM of SVM-$\ell_1$ with clustering displays the best trade-off between prediction accuracy and support recovery. It has a predictive performance similar to Graph-net, and a support recovery close to TV-$\ell_1$.

5.3. Delineating brain regions

An important question regarding brain decoders is whether they segment well the brain regions that support the decoding. The validation of this question is hard, yet there is evidence that relying on ensembles of models is a good approach (Leung and Barron, 2006; Zhou, 2012). Fig. 7 displays the decoder maps for the face-recognition tasks. For these tasks, we expect prediction to be driven
Relative prediction accuracy, weight stability and computation time for different classification tasks. Values are displayed relative to the mean over all the classifiers. a) FReM improves prediction accuracy, and when applied with clustering it also reduces the variability. The ensembles of models consistently improve the weights stability, with a computation time smaller than TV-ℓ1 and Graph-net.

b) Across-subjects discrimination

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Relative prediction score</th>
<th>Relative weight stability</th>
<th>Relative computation time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph-net</td>
<td>1.4e-3 &gt; 1.6e-4 &gt; 7.8e-14 &lt;</td>
<td>1.4e-33 &gt; 1.5e-33 &gt; 1.7e-3 &lt;</td>
<td>1.2e-6 &gt; 1.4e-3 &lt; 2.1e-3 &lt;</td>
</tr>
<tr>
<td>TV-ℓ1</td>
<td>6.3e-5 &gt; 8.3e-3 &gt; 7.8e-14 &lt;</td>
<td>1.4e-33 &gt; 1.4e-33 &gt; 1.7e-3 &lt;</td>
<td>1.4e-33 &gt; 1.4e-3 &lt; 1.4e-32 &gt;</td>
</tr>
<tr>
<td>SVM-ℓ2</td>
<td>1.4e-9 &gt; 8.9e-12 &gt; 7.8e-14 &gt;</td>
<td>1.4e-3 &lt; 1.4e-3 &lt; 1.4e-3 &lt;</td>
<td>1.4e-3 &lt; 1.4e-3 &lt; 1.4e-32 &gt;</td>
</tr>
<tr>
<td>SVM-ℓ1</td>
<td>2.4e-3 &gt; 7.8e-14 &gt; 7.8e-14 &gt;</td>
<td>SVM-ℓ2 &gt; SVM-ℓ1 &gt; SVM-ℓ0</td>
<td>SVM-ℓ2 &gt; SVM-ℓ1 &gt; SVM-ℓ0</td>
</tr>
<tr>
<td>SVM-ℓ0</td>
<td>SVM-ℓ2 &gt; SVM-ℓ1 &gt; SVM-ℓ0</td>
<td>SVM-ℓ2 &gt; SVM-ℓ1 &gt; SVM-ℓ0</td>
<td>SVM-ℓ2 &gt; SVM-ℓ1 &gt; SVM-ℓ0</td>
</tr>
<tr>
<td>SVM-ℓ2 + clustering</td>
<td>SVM-ℓ2 &gt; SVM-ℓ1 &gt; SVM-ℓ0</td>
<td>SVM-ℓ2 &gt; SVM-ℓ1 &gt; SVM-ℓ0</td>
<td>SVM-ℓ2 &gt; SVM-ℓ1 &gt; SVM-ℓ0</td>
</tr>
<tr>
<td>SVM-ℓ1 + clustering</td>
<td>SVM-ℓ2 &gt; SVM-ℓ1 &gt; SVM-ℓ0</td>
<td>SVM-ℓ2 &gt; SVM-ℓ1 &gt; SVM-ℓ0</td>
<td>SVM-ℓ2 &gt; SVM-ℓ1 &gt; SVM-ℓ0</td>
</tr>
<tr>
<td>SVM-ℓ0 + clustering</td>
<td>SVM-ℓ2 &gt; SVM-ℓ1 &gt; SVM-ℓ0</td>
<td>SVM-ℓ2 &gt; SVM-ℓ1 &gt; SVM-ℓ0</td>
<td>SVM-ℓ2 &gt; SVM-ℓ1 &gt; SVM-ℓ0</td>
</tr>
<tr>
<td>SVM-ℓ2</td>
<td>SVM-ℓ1 &gt; SVM-ℓ0</td>
<td>SVM-ℓ2 &gt; SVM-ℓ1 &gt; SVM-ℓ0</td>
<td>SVM-ℓ2 &gt; SVM-ℓ1 &gt; SVM-ℓ0</td>
</tr>
<tr>
<td>SVM-ℓ1</td>
<td>SVM-ℓ2 &gt; SVM-ℓ1 &gt; SVM-ℓ0</td>
<td>SVM-ℓ2 &gt; SVM-ℓ1 &gt; SVM-ℓ0</td>
<td>SVM-ℓ2 &gt; SVM-ℓ1 &gt; SVM-ℓ0</td>
</tr>
<tr>
<td>SVM-ℓ0</td>
<td>SVM-ℓ2 &gt; SVM-ℓ1 &gt; SVM-ℓ0</td>
<td>SVM-ℓ2 &gt; SVM-ℓ1 &gt; SVM-ℓ0</td>
<td>SVM-ℓ2 &gt; SVM-ℓ1 &gt; SVM-ℓ0</td>
</tr>
<tr>
<td>SVM-ℓ2 + clustering</td>
<td>SVM-ℓ2 &gt; SVM-ℓ1 &gt; SVM-ℓ0</td>
<td>SVM-ℓ2 &gt; SVM-ℓ1 &gt; SVM-ℓ0</td>
<td>SVM-ℓ2 &gt; SVM-ℓ1 &gt; SVM-ℓ0</td>
</tr>
<tr>
<td>SVM-ℓ1 + clustering</td>
<td>SVM-ℓ2 &gt; SVM-ℓ1 &gt; SVM-ℓ0</td>
<td>SVM-ℓ2 &gt; SVM-ℓ1 &gt; SVM-ℓ0</td>
<td>SVM-ℓ2 &gt; SVM-ℓ1 &gt; SVM-0</td>
</tr>
<tr>
<td>SVM-ℓ0 + clustering</td>
<td>SVM-ℓ2 &gt; SVM-ℓ1 &gt; SVM-0</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
</tr>
<tr>
<td>Graph-net</td>
<td>SVM-ℓ2 &gt; SVM-ℓ1 &gt; SVM-ℓ0</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
</tr>
<tr>
<td>TV-ℓ1</td>
<td>SVM-ℓ2 &gt; SVM-ℓ1 &gt; SVM-ℓ0</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
</tr>
<tr>
<td>SVM-ℓ2</td>
<td>SVM-ℓ1 &gt; SVM-ℓ0</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
</tr>
<tr>
<td>SVM-ℓ1</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
</tr>
<tr>
<td>SVM-ℓ0</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
</tr>
<tr>
<td>SVM-ℓ2 + clustering</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
</tr>
<tr>
<td>SVM-ℓ1 + clustering</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
</tr>
<tr>
<td>SVM-ℓ0 + clustering</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
</tr>
<tr>
<td>SVM-ℓ2</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
</tr>
<tr>
<td>SVM-ℓ1</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
</tr>
<tr>
<td>SVM-ℓ0</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
</tr>
<tr>
<td>SVM-ℓ2 + clustering</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
</tr>
<tr>
<td>SVM-ℓ1 + clustering</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
</tr>
<tr>
<td>SVM-ℓ0 + clustering</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
<td>SVM-2 &gt; SVM-1 &gt; SVM-0</td>
</tr>
</tbody>
</table>

Table 1: Comparison of performance: Each decoder is compared with a reference. The values correspond to Bonferroni-corrected p-values obtained by paired Wilcoxon rank test. The direction in the parenthesis denotes the sign of the mean difference, and bold text denotes a significant results (p < 10^{-10}). Results without smoothing are presented in Table C.2.

by the functional areas of the visual cortex (Grill-Spector and Malach, 2004). Indeed, the maps outline regions in known visual areas –e.g. the fusiform face area (PPA). In both within-subject and across-subject datasets, the
SVM-ℓ₁ maps are unstructured, and even if using FReM of this model improves the stability of the weight maps, these maps remain scattered with a large number of small clusters. However, the use of clustering yields less and larger clusters, with maps that are qualitatively similar to TV-ℓ₁ maps. Graph-net and SVM-ℓ₂ display similar behavior, yielding various small clusters around large clusters of activation. In the case of SVM, the use of FReM can reduce the number of small clusters. The combination with clustering enhances this effect. Note that setting the threshold to visualize regions is difficult task as the noise level is unknown.

5.4. Setting the number of estimators to ensemble

The choice of the number b of estimators to combine also affects the stability of the weight maps and the running time. The number b can be interpreted as a “smoothing” parameter (Bühlmann and Yu, 2002), and the computation time that one is willing to pay to train a decoder. We measure the performance of ensembles of classifiers on three different datasets and across 10 folds of cross-validation. In practice, we often use a number of estimators between 50 and 100, but to verify if the model converges, we consider here a range from 10 to 640 estimators.

Fig. 8 shows that for ensembles of classifiers, prediction accuracy does not depend on the number of estimators, whereas the computation time is almost linear (t ∝ b, where γ ≈ 1). We use the running time as a constraint to finally set the number of estimators to 50, as the weight stability of non-sparse classifiers are at least 95% of the asymptotic optimum. In addition, this is a good compromise between stability and computation cost. Hence, we use this number of estimators throughout all experiments.

6. Results: parallel computing of brain decoders

One important feature of ensembling models is scalability, as these methods can be trained in parallel in a multicore, shared-memory environment. This corresponds to current standard workstations, which frequently have a large number of CPUs. Here, we measure the training time of various decoders across 5 folds of cross-validation. We perform face-discrimination tasks on two datasets with different sizes.

Fig. 9 shows that, in general, there is not an ideal decrease in the computation time as more CPUs are added. The SVM with ℓ₁ and ℓ₂ penalty are the fastest. In contrast, TV-ℓ₁ is the slowest, followed by Graph-net. In both datasets, FReM displays most of the speed up at 10 CPUs, and reaches a minimum at 20. These methods are much

---

9Parallel computation was run using joblib https://pythonhosted.org/joblib/ to use multiple cores on a single computer with Python “multiprocessing”. Benchmarks were done on Intel Xeon E5-2697 CPUs, clocked at 2.7GHz, with 12 cores per CPUs, on a single Linux (Ubuntu 16.04) computer.
a) Within-subject discrimination

Task: face vs house

b) Across-subjects discrimination

Task: face vs shape

Figure 7: Qualitative comparison of decoder weight maps: Weight maps for different discriminative tasks on the Haxby and HCP datasets. The maps are thresholded at the 99 percentile for visualization purposes. In both dataset, (top) illustration of weight maps for the face-recognition task; (bottom) outlines of the other tasks. The weight maps obtained with TV-L1 and FReM methods with clustering display a prediction driven by the functional areas of the visual mosaic, such as: primary visual areas, lateral occipital complex, the face and place specific regions in the fusiform gyrus. An example of the difference of outlines between FReM and its base estimator see Figure D.3.
faster that Graph-net. The combination of FReM with clustering does not increase computation cost as we use a fast clustering algorithm.

7. Results: small-sample recovery behavior of decoders

In fMRI, despite growing efforts in data accumulation (Essen et al., 2012; Poldrack et al., 2013), the sample size remains small in comparison with the number of voxels. Therefore, an important aspect of the brain decoders is their sample complexity - i.e. the number of samples required to bound the estimation error. Yet, assessing the recovery of weight maps is difficult, as we do not have access to the asymptotic result. To bypass this problem, we measure the similarity between the weight maps obtained with different sample sizes and the ones obtained using the whole dataset. This gives us an intuition on the small-sample recovery capacity.

Fig. 10 shows that across datasets, FReM of SVM-ℓ₂ with and without clustering, and FReM of SVM-ℓ₁ with clustering are consistently the best. In contrast, the SVM-ℓ₁ and Log-enet fail to recover the final weight maps. TV-ℓ₁, Graph-net, and SVM-ℓ₂ have a good performance on both datasets. FReM of SVM-ℓ₁ outperforms these methods on the within-subject discrimination, as the weight similarity rapidly increases. On across-subject datasets, FReM of SVM-ℓ₁ with clustering has almost the same performance as TV-ℓ₁, Graph-net, and SVM-ℓ₂, differing only after using 80% of the data for training.

8. Discussion and conclusion: using FReM

We have introduced a fast strategy to train regularized ensembles of models, FReM, that improves the stability of brain decoders. This scheme is summarized as follows: i) For each fold of the nested cross-validation loop, we select the estimator with the best predictive power; ii) we build an estimator by storing the models for all folds of cross-validation and averaging them. This approach differs from the stability selection methods (Meinshausen and Bühlmann, 2010; Varoquaux et al., 2012), as here we use the amplitude of predictive weight maps for the model aggregation, and not only their support.

Using FReM. The predictive power of FReM is not very dependent on the number of estimators used during the aggregation step. On the other hand, the stability of the resulting decoder improves as more estimators are used. On across-subjects datasets, the use of clustering improves the stability of sparse methods. In terms of computation time, this scheme displays an almost linear complexity in the number of estimators. Thus, setting the number of estimators is an arbitrary choice, it depends only on the computation resources available.

Comparing decoders. In both within and across subjects datasets, FReM has shown an improvement of the performance of the base estimator. This strategy reduces the variance of predictive power and increases the stability of weight maps of the base estimator. It also improves the small-sample behavior of the base estimators, boosting the
Figure 10: **Small-sample recovery behavior of decoders**: Consistently the slowest methods.

Acknowledgment

This project has received funding from the European Union’s Horizon 2020 Framework Programme for Research and Innovation under Grant Agreement No 720270 (Human Brain Project SGA1).

References


Appendix A. Bagging reduces the variance of the base model

A learning problem can be cast as identifying the best hypothesis in a space of hypotheses. Statistical issues arise when the sample size is small compared to the size of the hypothesis space. In this setting, there are several hypotheses that give the same prediction accuracy on training data. One can reduce the risk of choosing a "wrong" estimator by averaging the output of several of these estimators (Dietterich, 2000).

Bagging in regression (Breiman, 1996): We present Breiman’s proof of bagging regressors to demonstrate its benefits. Let $D$ be a training set which contains a sample of independent $(x,y)$ drawn from the distribution $P$. Define $h(x,D)$ to be prediction function based on the sample $D$. Let $h_{agg}(x) = \mathbb{E}_D[h(x,D)]$ be an aggregate of prediction functions.

Take $x$ to be a fixed predictor and $y$ an output value. Then

$$\mathbb{E}_D[(y - h(x,D))^2] \geq \mathbb{E}_D[y - h(x,D)]^2 \quad \text{(by Jensen’s inequality)}$$

$$= y^2 - 2y\mathbb{E}_D[h(x,D)] + \mathbb{E}_D[h(x,D)]^2$$

$$= (y - h_{agg}(x))^2.$$ 

Hence, the averaged predictor has lower mean-squared error than the base predictor. This improvement depends on how unequal $\mathbb{E}_D[h(x,D)]^2 \geq \mathbb{E}_D[h(x,D)]^2$. The more the $h(x,D)$ vary with respect to each other, the more improvement the aggregation may produce.

To understand the effect of averaging, let us assume the extreme case where the models created by sampling are i.i.d. Let the aggregation be the mean of the predicted values $h_{agg}(x) = \frac{1}{b} \sum_{i=1}^{b} h_i(x,D)$, where $h_i(x,D)$ denotes the prediction function based on the sample $D$. The predictions are i.i.d., and the variance of each of them is defined by $\mathbb{E}_D[(y - h_i(x,D))^2] = \sigma^2$. Hence, the variance of the aggregated estimator is:

$${\mathbb{E}_D[(y - h_{agg}(x))^2]}= \frac{1}{b^2} \mathbb{E}_D \left[ \left( \frac{1}{b} \sum_{i=1}^{b} y - h_i(x,D) \right)^2 \right]$$

$$= \frac{1}{b^2} \sum_{i=1}^{b} \mathbb{E}_D[(y - h_i(x,D))^2] \quad \text{(by independence)}$$

$$= \frac{1}{b} \sigma^2.$$ 

We can see that averaging decreases the error as $\sqrt{b}$. Note that the i.i.d. case studied here is the most favorable case.

Appendix B. Experiments on Simulated data

Appendix B.1. Dataset simulation

We use the same approach presented in Michel et al. (2011) to simulate data satisfying the Eq. 1. The matrix $X$ consists of $n = 100$ images, and $p = 1728$ voxels (size $12 \times 12 \times 12$). Each image contains a set of five square Regions of Interest (ROIs) (size $2 \times 2 \times 2$), and each of the four ROIs has a fixed weight in $\{-0.6, 0.5, -0.6, 0.5, 0.5\}$. Let us denote $S$ the support of the ROIs (i.e. the 40 resulting voxels of interest), and $w_{i,j,k}$ denotes the weights of the $(i,j,k)$ voxel. The resulting images are smoothed with a Gaussian kernel with a standard deviation of 2 voxels, to mimic the correlation structure observed in real fMRI data. To simulate the spatial variability between images (inter-subject variability, movement artifacts in intra-subject variability), we define a new support of the ROIs, $\hat{S}$ such as, for each image $l$-th, 50% (randomly chosen) of the weights $\mathbf{w}$ are set to zero. Thus, we have $\hat{S} \subset S$. We simulate the target $y$ for the $l$-th image as:

$$y_l = \sum_{(i,j,k) \in \hat{S}} w_{i,j,k} \mathbf{X}_{i,j,k,l} + \epsilon_l,$$  \hspace{1cm} (B.1)

with the signal in the $(i,j,k)$ voxel of the $l$-th image simulated as:

$$\mathbf{X}_{i,j,k,l} \sim N(0,1).$$  \hspace{1cm} (B.2)

$\epsilon \sim N(0, \gamma)$. We choose $\gamma$ in order to have a SNR of 5 dB. Finally, we apply a sign function to Eq. B.1 to obtain a binary targets.

Appendix C. Benchmarking results without smoothing

Results without spatial smoothing of the voxels are given in Fig. C.2. The results obtained with and without spatial smoothing of the voxels are consistent. In within subject settings, structured sparse methods display good prediction accuracy and stability of weight maps. However, the computation time is slow. FReM of SVM-$\ell_1$ yields to a better performance, increasing prediction accuracy and weight map stability, while reducing the computation time.

In between subject settings, FReM over performs other methods. In particular, FReM of SVM-$\ell_2$ with clustering.

Table C.2 shows the comparison between each decoder and the mean performance across models. These results confirm the above observations.

Appendix D. FReM improves brain regions delineation

Fig. D.3 displays the difference between the outlines obtained with FReM and its base model. The difference between the outlined brain maps is characterized by small clusters. As expected, this effect is higher for SVM-$\ell_1$. FReM of SVM-$\ell_1$ with clustering leads to greater brain activation areas that are not found by the base estimator. On the other hand, FReM of SVM-$\ell_2$ with clustering displays a marginal difference with its non-clustered counterpart.
Figure B.1: **Weight maps of the decoder:** Weight maps for different decoders on simulated data. The values in parenthesis correspond to the Pearson correlation to ground truth. Note that the maps are unthresholded.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>prediction score</th>
<th>weight stability</th>
<th>computation time</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Graph-net (ρ = 0.62)</strong></td>
<td>2e-6 &gt;</td>
<td>8.1e-8 &gt;</td>
<td>7.8e-14 &lt;</td>
</tr>
<tr>
<td><strong>TV-ℓ₁ (ρ = 0.72)</strong></td>
<td>1 &gt;</td>
<td>5.8e-2 &gt;</td>
<td>7.8e-14 &lt;</td>
</tr>
<tr>
<td>Log-enet</td>
<td>1 &gt;</td>
<td>7.8e-14 &gt;</td>
<td>7.8e-14 &gt;</td>
</tr>
<tr>
<td>SVM-ℓ₁</td>
<td>4.5e-10 &gt;</td>
<td>1.8e-12 &gt;</td>
<td>7.8e-14 &gt;</td>
</tr>
<tr>
<td>SVM-ℓ₂</td>
<td>1.7e-2 &gt;</td>
<td>8.4e-14 &gt;</td>
<td>7.8e-14 &gt;</td>
</tr>
<tr>
<td>SVM-ℓ₁</td>
<td>1.8e-11 &gt;</td>
<td>1 &gt;</td>
<td>1 &gt;</td>
</tr>
<tr>
<td>SVM-ℓ₂ + clustering</td>
<td>4.6e-6 &gt;</td>
<td>9.5e-3 &lt;</td>
<td>1 &gt;</td>
</tr>
<tr>
<td>SVM-ℓ₁ + clustering</td>
<td>1 &gt;</td>
<td>0.2 &lt;</td>
<td>0.5 &gt;</td>
</tr>
<tr>
<td><strong>SVM-ℓ₁ (ρ = 0.36)</strong></td>
<td>1 &gt;</td>
<td>5.8e-2 &gt;</td>
<td>7.8e-14 &lt;</td>
</tr>
<tr>
<td>SVM-ℓ₂</td>
<td>4.5e-10 &gt;</td>
<td>1.8e-12 &gt;</td>
<td>7.8e-14 &gt;</td>
</tr>
<tr>
<td>SVM-ℓ₁</td>
<td>1.7e-2 &gt;</td>
<td>8.4e-14 &gt;</td>
<td>7.8e-14 &gt;</td>
</tr>
<tr>
<td>SVM-ℓ₂ + clustering</td>
<td>1 &gt;</td>
<td>5.8e-2 &gt;</td>
<td>7.8e-14 &lt;</td>
</tr>
<tr>
<td>SVM-ℓ₁ + clustering</td>
<td>4.5e-10 &gt;</td>
<td>1.8e-12 &gt;</td>
<td>7.8e-14 &gt;</td>
</tr>
<tr>
<td><strong>SVM-ℓ₁ (ρ = 0.33)</strong></td>
<td>1 &gt;</td>
<td>5.8e-2 &gt;</td>
<td>7.8e-14 &lt;</td>
</tr>
<tr>
<td>SVM-ℓ₂</td>
<td>4.5e-10 &gt;</td>
<td>1.8e-12 &gt;</td>
<td>7.8e-14 &gt;</td>
</tr>
<tr>
<td>SVM-ℓ₁</td>
<td>1.7e-2 &gt;</td>
<td>8.4e-14 &gt;</td>
<td>7.8e-14 &gt;</td>
</tr>
<tr>
<td>SVM-ℓ₂ + clustering</td>
<td>1 &gt;</td>
<td>5.8e-2 &gt;</td>
<td>7.8e-14 &lt;</td>
</tr>
<tr>
<td>SVM-ℓ₁ + clustering</td>
<td>4.5e-10 &gt;</td>
<td>1.8e-12 &gt;</td>
<td>7.8e-14 &gt;</td>
</tr>
<tr>
<td><strong>SVM-ℓ₁ + clustering (ρ = 0.43)</strong></td>
<td>1 &gt;</td>
<td>5.8e-2 &gt;</td>
<td>7.8e-14 &lt;</td>
</tr>
<tr>
<td><strong>SVM-ℓ₂ + clustering (ρ = 0.41)</strong></td>
<td>4.5e-10 &gt;</td>
<td>1.8e-12 &gt;</td>
<td>7.8e-14 &gt;</td>
</tr>
<tr>
<td><strong>SVM-ℓ₁ + clustering (ρ = 0.65)</strong></td>
<td>1 &gt;</td>
<td>5.8e-2 &gt;</td>
<td>7.8e-14 &lt;</td>
</tr>
<tr>
<td><strong>FReM: SVM-ℓ₁ + clustering (ρ = 0.53)</strong></td>
<td>1 &gt;</td>
<td>5.8e-2 &gt;</td>
<td>7.8e-14 &lt;</td>
</tr>
</tbody>
</table>

Table C.2: **Comparison of performance:** Each decoder is compared with a reference. The values correspond to Bonferroni-corrected p-values obtained by paired Wilcoxon rank test. The direction in the parenthesis denotes the sign of the mean difference, and bold text denotes a significant results (p < 10⁻¹⁰).
Figure C.2: **Relative performance**: Relative prediction accuracy, weight stability and computation time for different classification tasks. Values are displayed relative to the mean over all the classifiers.
Within-subject discrimination

Figure D.3: **Difference on delineated brain regions**: Difference on weight maps of various decoders.