Frontiers in Neurosciences 28June 2013

Machine Learning for Neuroimaging with Scikit-Learn

Alexandre Abraham [1,2,*], **Philippe Gervais** [1,2], **Fabian Pedregosa** [1,2], **Andreas Muller, Jean Kossaifi, Michael Eickenberg, Alexandre Gramfort, Bertrand Thirion** [1,2] **and Gaël Varoquaux** [1,2]

[1] *Parietal Team, INRIA Saclay-Île-de-France, Saclay, France*
[2] *Neurospin, I²BM, DSV, CEA, 91191 Gif-Sur-Yvette, France*

Correspondence*:
Alexandre Abraham
Parietal Team, INRIA Saclay-Île-de-France, Saclay, France,
alexandre.abraham@inria.fr

Research Topic

ABSTRACT

Statistical learning methods are increasingly used to perform neuroimaging analysis. Their main virtue for this type of application is their ability to model high-dimensional datasets, e.g. multivariate analysis of activation images, or capturing inter-subject variability. Supervised learning is typically used in decoding setting to relate brain images to behavioral or clinical observations, while unsupervised learning is typically used to uncover hidden structure in sets of images (e.g. resting state functional MRI) or to find sub-populations in large cohorts of subjects. By considering functional neuroimaging use cases, we illustrate how the Scikit-learn, a Python machine learning library, can be used to perform some key analysis steps. Scikit-learn contains a large set of statistical learning algorithms, both supervised and unsupervised, that can be applied to neuroimaging data after a proper preprocessing. Combined with other Python libraries, neuroimaging data can be loaded, processed and the results can be visualised easily.

Keywords: Machine learning, Statistical Learning, Neuroimaging, Scikit-learn, Python

1 INTRODUCTION

1.1 SCIENTIFIC PYTHON AND NEUROIMAGING ECOSYSTEM

1.1.1 Scipy and Numpy Scipy and Numpy python packages are the basis of scientific computing in Python. They provide efficient and easy to use data representation, linear algebra, statistics, algortihms... They are the elementary bricks we use in all our algorithms.

1.1.2 matplotlib Matplotlib is a plotting library that is part of the scientific python stack. It offers a Matlab-like experience and allows to display plots, images or even 3D plots in a graphical user interface. We have used it to generate all the figures of this paper.

Note that a more convenient interface for matplotlib called pylab allows a procedural use of matplotlib.

1.1.3 nibabel Nibabel is a neuroimaging data loading package. Nibabel can load or save data in the most popular neuroimaging data format. This is indeed an entry point of all our scripts.

1.1.4 nipy

1.1.5 scikit-learn

2 SCIKIT-LEARN CONCEPTS

2.1 ESTIMATOR

2.2 DATA REPRESENTATION

In the scikit learn, and in the world of statistical machine learning, data are usually represented in a 2-dimensional matrix of shape $n\_samples \times n\_features$.

2.3 TRANSFORMER

A tranformer is an object that exposes a `transform` method. If the transformation can be inverted, a method called `inverse_transform` also exists.

2.4 CROSS VALIDATION

It seems more right to me to put it in this part

3 FROM MR VOLUMES TO A DATA MATRIX

As any domain specific data, MR volumes holds particular properties. Understanding them is crucial to be sure to make proper use of the data.

$$\begin{bmatrix} r_x & 0 & 0 & o_x \\ 0 & r_y & 0 & o_y \\ 0 & 0 & r_z & o_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

3.1 DATA PREPARATION

At this point, we suppose that standard preprocessings have been applied to the data. They should be registrated on a common template (MNI for example). However, data is not yet ready to be processed by the scikit-learn. In fact, preprocessed data may have different shapes. Moreover, it is essential to get rid of some remaining scanner artefacts and individual trends.

3.1.1 Detrending Detrending is an essential step when dealing with fMRI data. It removes a best-fit linear trend (in the least square sense) over the time series of each voxel. It is obviously needed when you want to study the correlation between features.

Gal told me not to go deep into the maths, I wonder if talking about least squares is a good idea. Maybe I should say that a constant trend is a mean and a linear trend is simply a linear function

3.2 RESAMPLING

Resampling consists in changing the shape of the data. This is typically needed when dealing when data coming from an heterogenous dataset, as the shape depends on acquisition parameters.

Practically, resampling is an interpolation and thus may alterate the integrity of the data. That is why it should be used carefully. Oversampling (increasing data resolution) leads to higher memory consumption and computation resources. Downsampling is commonly used to reduce the size of the data we want to process.

Typical sizes are 2mm or 3mm resolutions, but the spread of high field MR scanner tends to lower these values.

- Removing confounds is necessary for some treatments

3.3 SIGNAL CLEANING

- Remove high frequency (scanner artefacts)

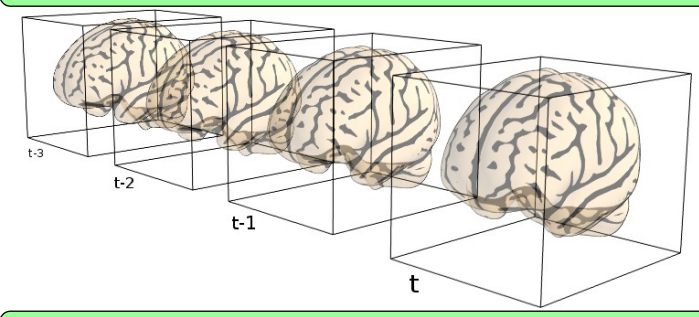
3.4 MASKING

3.4.1 From 4-dimensional image to 2-dimensional array Neuroimaging data are represented in 4 dimensions: 3 dimensions for the scans, which are positioned in a coordinate space, and one dimension for the time. Scikit-learn algorithms, on the other hand, only accept 2-dimensional data: one dimension for the features and one for the samples.

Consequently, in order to use neuroimaging data in the scikit-learn, a conversion is needed. The most simple way to achieve that would be to *flatten* the 3D scans into a 1D array. However, we know that not every voxels in a neuroimaging scan is useful. In particular, outter-brain voxels are of no use and, worse, they can bring spurious noise and scanner artefacts (such as ghosts).

To sort out voxels of interest, we will have to apply a mask on the data. Most of public datasets provide a mask, come of them even provide several, isolating different functional or anatomical brain regions.

ref to Haxby



Should tell here that some algorithms, like logistic regression, do not like colinear features.

3.4.2 Automatically computing a mask The simplest strategy to compute a mask is a binarization by a selected threshold. Due to the nature of the neuroimaging data, there exists some strategy to choose this threshold in order to obtain a decent segmentation.

There is a reference for the method used in Nisl. We should put it there and in the code. Add a figure with an histogram to illustrate.

Multi subject computation is simply done by intersecting subjects maps relatively to a chose threshold.

3.4.3 Conserving geometrical structure Applying a mask on the data obviously remove the 3-dimensional structure of the data. However, some algorithms, like the Ward, need this structural information to run.

- Speak about connectivity graphs / adjacency matrices

3.5 LABEL SHIFTING

Functional MRI measures brain activity by using the Blood-Oxygen-Level-Dependent contrast (BOLD). In fact, like muscles, brain regions consumes more oxygen and nutriments when stimulated. So when a part of the brain starts working, physiological mechanisms induce an oxygen-rich blood flood toward this particular region: this is called haemodynamic response.

However, this reaction takes time, usually around 6 seconds. This is the duration between the event and the reaction observed in the brain. To be able to match these two events, we will sometimes have to shift our data. This can be done by removing the two first scans of the data and the two last values of the labels (to keep an homogeneous length).

```
func = func[2:]
labels = labels[:-2]
```

4 DECODING

The process of predicting behavioral or comportamental data from fMRI scan is called decoding.

4.1 SVM

4.1.1 Haxby dataset For this example and the following, we will use a subset of the Haxby dataset (Haxby et al. (2001)). Haxby dataset is from a study about face and object representation into the brain (in particular in high level visual cortex). It is composed of 12 runs for each of the 6 subjects. Greyscale images representing faces, houses, cats, bottles, scissors, shoes, chairs and random textured were presented in 24 seconds blocks separated by rest periods. The repetition time (TR) between each scan is 2.5s.

To make this example easier, we will work on a subset of this dataset. We will consider only one subject and will try to classify faces versus houses.

4.1.2 Feature selection: ANOVA F-Test Even if the resolution of brain-imaging data seems low (3mm cubes, around 100000 neurones), from a computational point of view, this is a huge. For example, Haxby dataset has a resolution of $64 \times 64 \times 40 = 163840$ voxels. After applying the mask, only 39912 voxels are left, which is still high.

In order to reduce the number of features, we can aggregate them (in regions of interest for example) or we can select only the most relevant ones (those who correlates most with the task). As we expect a lot of feature to be irrelevant for our task, we opt for a feature selection method.

In supervised learning, the most popular feature selection method is the ANalysis Of VAriance (ANOVA) F-Test. This is a generalization of the t-test to more than 2 features. Basically, ANOVA compares several groups to determine if they are similar (ie randomly drawn form the same population, this is the null hypothesis). We use it to compare the distributions of the features values across the classes.

`sklearn.feature_selection` contains a panel of feature selection strategies. One can choose to take a percentile of the features (`SelectPercentile`), or a fixed number of features (`SelectKBest`) for example. All these objects are implemented as transformers. Here we use a fixed number of features and we use the `f_calssif` function (ANOVA F-Test) for scoring.

```
from sklearn.feature_selection import SelectKBest, f_classif

### Define the dimension reduction to be used.
# Here we use a classical univariate feature selection based on F-test,
# namely Anova. We set the number of features to be selected to 500
feature_selection = SelectKBest(f_classif, k=500)
```

4.1.3 Support Vector Classifier A Support Vector Classifier (SVC) is a simple classifier that finds a linear hyperplane that separates the samples. Classifying a new example boils down to seeing on which side of the hyperplane the example is. SVC has the advantage to give reliable results even when the number of dimensions is greater than the number of samples.

The decision is taken based upon a subset of training data called support vectors. We can say that these support vectors holds the information allowing to discriminate the two classes, this is why we will display them and try to see if they match some neuroscientific knowledge.

```
from sklearn.svm import SVC
clf = SVC(kernel='linear', C=1.)

### Look at the discriminating weights
svc = clf.support_vectors_
# reverse feature selection
svc = feature_selection.inverse_transform(svc)
```

4.1.4 Pipeline The workflow described above (feature selection + estimator) is a standard one. In fact, in most cases, the workflow will consist in atomic steps *linked* together (the output of a step is the input of the next one). For this purpose, scikit-learn offers a pipeline object that allows such linking. A pipeline is simply a list of scikit-learn objects through which the input data will be conveyed. The function to call for each object (transform, fit...) depends on its type. This allow the developpers to write a complete processing as a one-liner.

```
from sklearn.pipeline import Pipeline
anova_svc = Pipeline([('anova', feature_selection), ('svc', clf)])
```

4.1.5 Displaying the results

Should we define a visualization function once and for all?

4.2 SEARCHLIGHT

- Present the Searchlight problem
- Say it is less a pain to implement thanks to scikit-learn bricks (estimator and cross_val). Plus it is easily customizable.

4.3 CLASSIFICATION OF M/EEG SENSOR SPACE DATA

4.4 ORTHOGONAL MATCHING PURSUIT

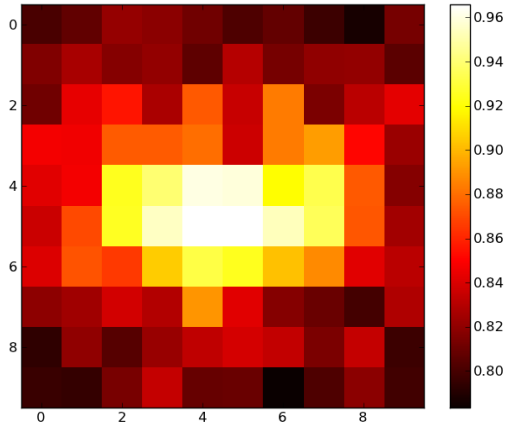
4.4.1 Kamitani dataset Kamitani dataset is based on a visual task like Haxby. In this experiment, several series of 10×10 binary images are presented to two subjects. Our goal will be to use the scikit-learn to learn a correlation between brain activation and voxel color.

Kamitani training set is composed of random images (where black and white pixels are balanced). The testing set is composed of structured images containing geometric shape (square, cross...) and letters (spelling the word *neuro*).

There are two ways to establish a link between brain voxels and image pixels: we can either try to reconstruct image from brain voxel activation, this is called decoding, or we can try to predict brain activation from an image, this is called encoding.

In the present example, we will do both encoding and decoding and see if the results match.

4.4.2 Preprocessing Classical preprocessing (detrend...) have been applied to the data.



4.4.3 Decoding: reconstructing image from brain activity

5 ENCODING

After talking with Michael, he told me that he could make a fairly simple example for encoding, which I think is a plus for the paper. The example will be integrated in Nisl.

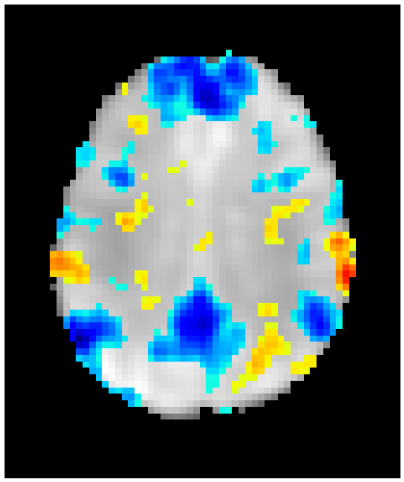
6 FUNCTIONAL CONNECTIVITY

Should we speak of correlation matrices to represent interaction between regions?

6.1 INDEPENDENT COMPONENT ANALYSIS (ICA)

6.1.1 Intuition ICA is a blind source separation method. Its principle is to separate a multivariate signal into several components by maximizing their non-gaussianity. A typical example is the *cocktail party problem* where ICA separates the voices of people using signal from several mikes.

It is historically the reference method to extract networks from resting state fMRI Biswal and Ulmer (1999).



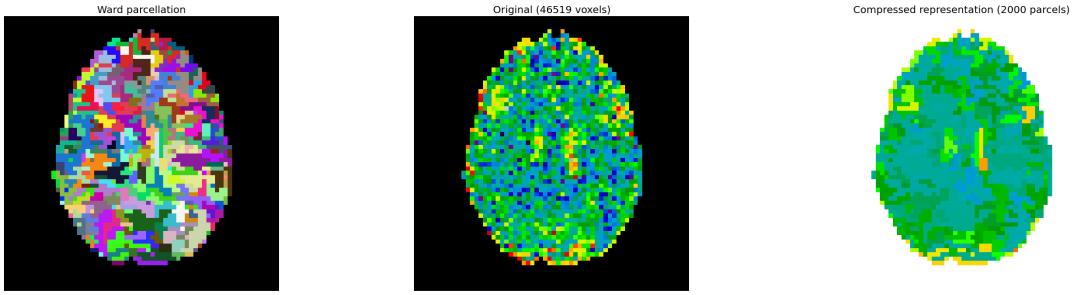
6.1.2 Application

6.2 CLUSTERING

Make an example with Ward Clustering. Indicate then that other algorithms can be used such as KMeans and Spectral clustering and only give results.

We use a PCA here to reduce dimensionality.

Bonus: may be used as dimensionality reduction



DISCLOSURE/CONFLICT-OF-INTEREST STATEMENT

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

ACKNOWLEDGEMENT

Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text.

Funding: Text Text Text Text Text Text Text Text.

SUPPLEMENTAL DATA

Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text.

REFERENCES

Haxby, J. V., Gobbini, I. M., Furey, M. L., Ishai, A., Schouten, J. L., and Pietrini, P. (2001) Distributed and overlapping representations of faces and objects in ventral temporal cortex. *Science* 293 2425.

Biswal, B. and Ulmer, J. (1999) Blind source separation of multiple signal sources of fMRI data sets using independent component analysis. *Journal of computer assisted tomography* 23 265.