Machine learning for fMRI in Python: inverse inference with scikit-learn

G. Varoquaux[†⋆§] A. Gramfort[⋆‡§], F. Pedregosa[⋆§], V. Michel[⋆§], B. Thirion[⋆§],
⋆ Parietal Project, INRIA, France † Unicog, INSERM U992, France
‡ Martinos Center, MGH, Harvard Med. School, USA § Neurospin, CEA, France
scikit-learn-general@lists.sourceforge.net

Project vision: the goals

Building-blocks for fMRI statistical learning

- Bridge the gap between machine-learning research and brain imaging
- Well **documented** and **easy to use** by non-expert
- High numerical and prediction **performance**
- Building blocks for fMRI analysis: multivariate brain mapping, resting state data analysis

Technical choices

- Python: general-purpose, high-level language
- BSD license : reuse even in commercial settings

Numpy
+Scipy
+Matplotlib

NiPy
+ NiBabel

Methods

Supervised: fMRI inverse inference

- SVM, Lasso, Elastic-Net, Logistic Regression (L1 & L2), LDA, Naive Bayes, KNN, Gaussian Process
- Feature selection: recursive feature elimination (RFE)

Unsupervised: Clustering, Signal processing

- K-means, Gaussian Mixture Models, Mean-shift, Hierarchical clustering
- PCA, ICA, Sparse PCA, NMF

Model selection

- Cross-validation to automatically set parameters or compare methods in parallel

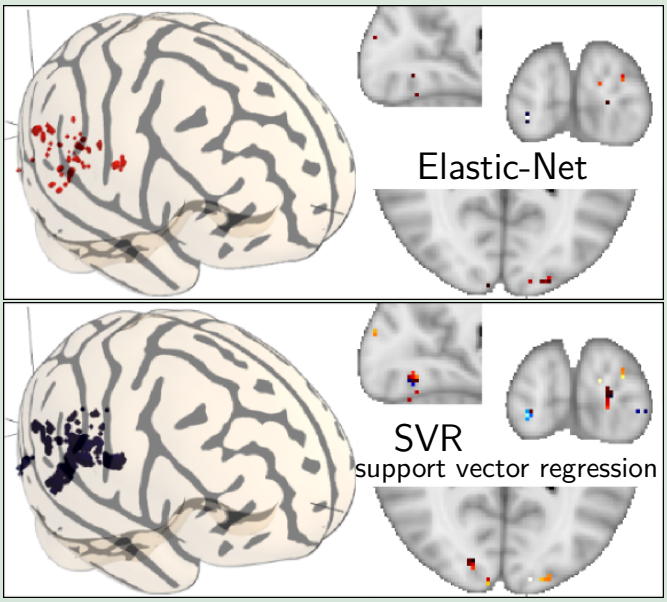
http://nisl.github.com **http://scikit-learn.sf.net**

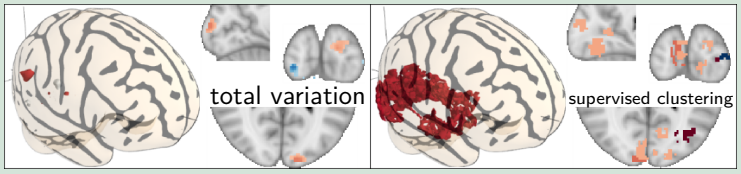
Example code of fMRI inverse inference

```
y, session = np.loadtxt("attributes.txt").astype("int").T
X = ni.load("bold.nii.gz").get_data()
mask = ni.load("mask.nii.gz").get_data()

# Process the data in order to have a two-dimensional design matrix
# X of shape (n_samples, n_features).
X = X[mask!=0].T

# Detrend data on each session independently
for s in np.unique(session):
    X[session==s] = signal.detrend(X[session==s], axis=0)

# Remove volumes corresponding to rest
X, y, session = X[y!=0], y[y!=0], session[y!=0]
n_samples, n_features = X.shape
n_conditions = np.size(np.unique(y))

# Define the prediction function to be used: Support Vector
# Classification, with a linear kernel and C=1
clf = SVC(kernel='linear', C=1.)

# Define the dimension reduction to be used: 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)

### We combine the dimension reduction and the prediction function
anova_svc = Pipeline([('anova', feature_selection), ('svc', clf)])

# Define the cross-validation scheme used for validation:
# a LeaveOneLabelOut on the session, which corresponds
# to a leave-one-session-out
cv = LeaveOneLabelOut(session)

# Compute the prediction accuracy for the different sessions
cv_scores = cross_val_score(anova_svc, X, y, cv=cv, n_jobs=-1,
                            verbose=1, iid=True)

### Return the corresponding mean prediction accuracy
classification_accuracy = np.sum(cv_scores) / float(n_samples)
print "Classification accuracy: %f" % classification_accuracy
```

Illustration

Prediction across subjects of the size of an object seen (10 subjects, 3 sizes)





References

- Scikit-learn (Pedregosa JMLR MLOSS 2010)
- Inverse inference (Michel Patt Rec 2011: **Supervised clustering**, Michel TMI 2011: **Total variation**, HBM poster 555)
- fMRI tutorial (Gramfort `http://nisl.github.com`)
- Resting state (Varoquaux 2010 Neuroimage, Varoquaux 2010 NIPS 2010, HBM poster 662)