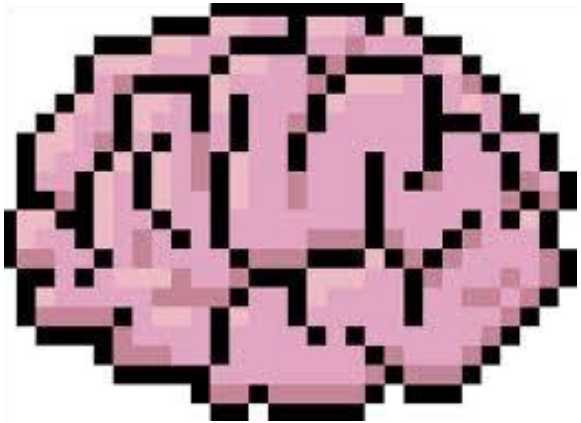
Decoding and encoding mental representations of objects in the brain

Syed Hussain Ather
BIOF509 Spring 2018



Motivation

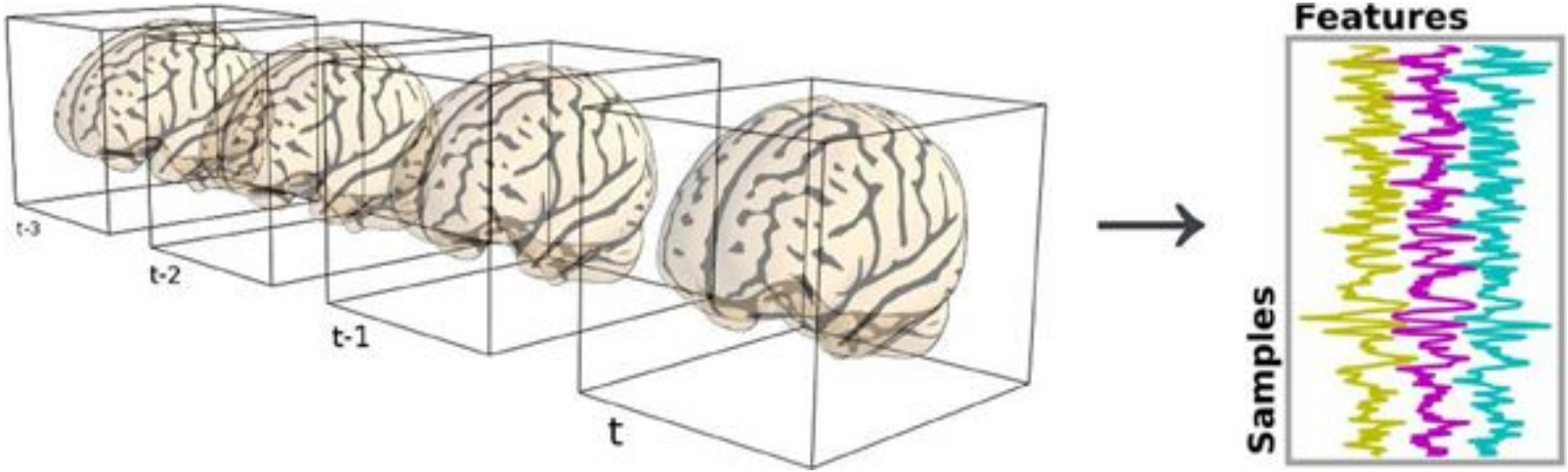
- Modeling neuroimaging data high-dimensional, complicated
- Supervised learning to decode images to relate brain images to behavioral or clinical observations
- Sci-kit learn can be used for this analysis
- Make predictions that can be cross-validated

NiLearn

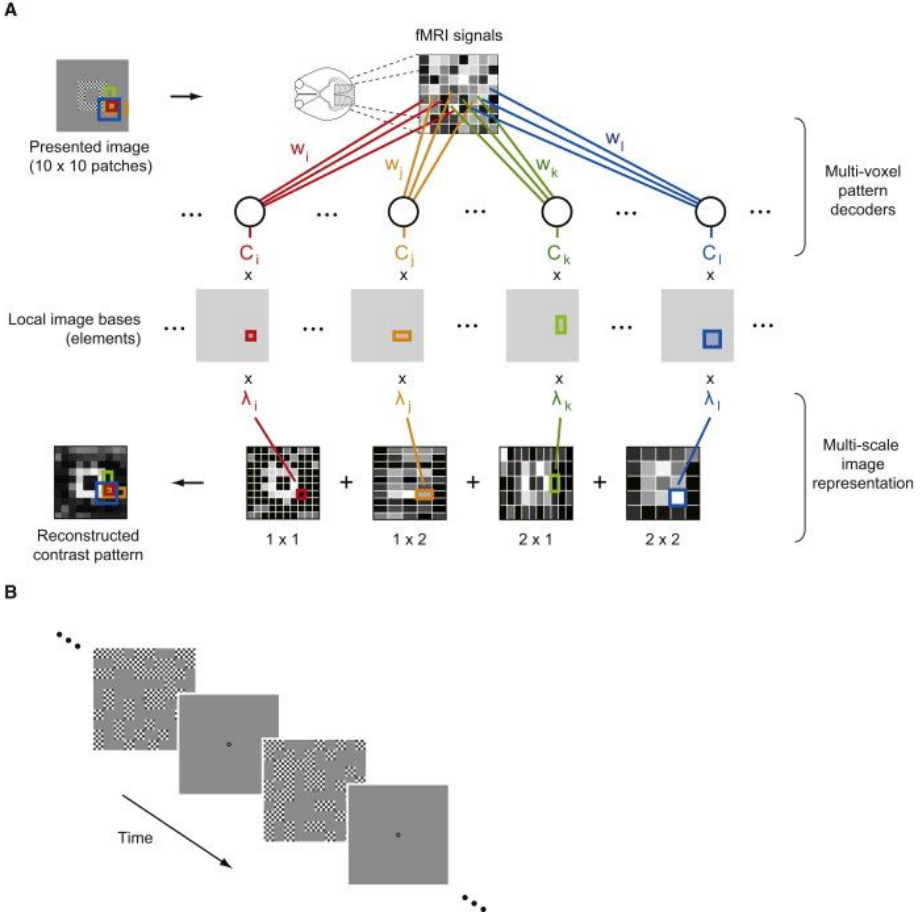
- simple interfaces for people to apply machine learning to neuroimaging data
- best visualizations for raw data and processed results
- built on scikit-learn

Preparing the dataset

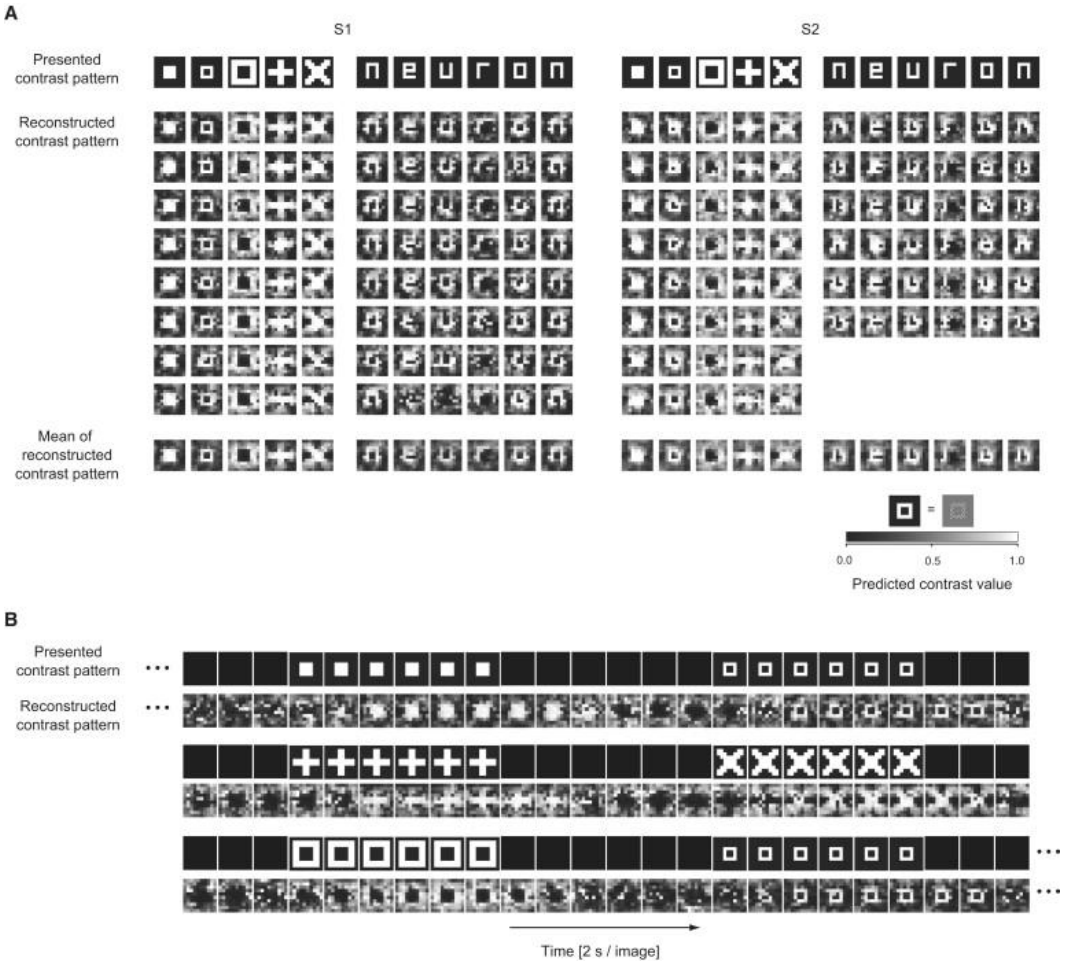
- NiftiMasker (4 dimensions -> 3 dimensions)
- Lose spatial structure
- Discard uninformative locations



Visual image reconstruction

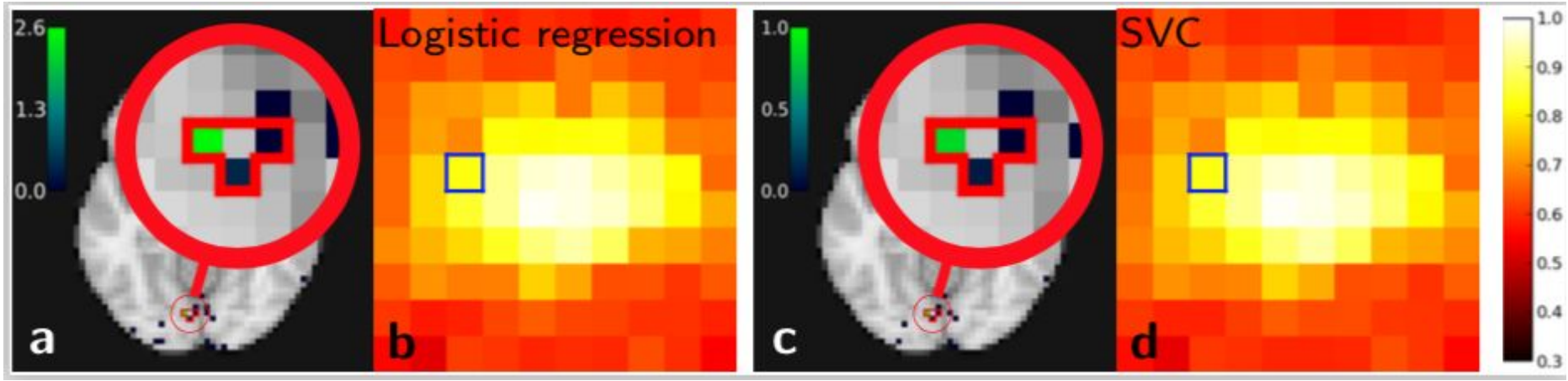


Reconstruction results



Decoding

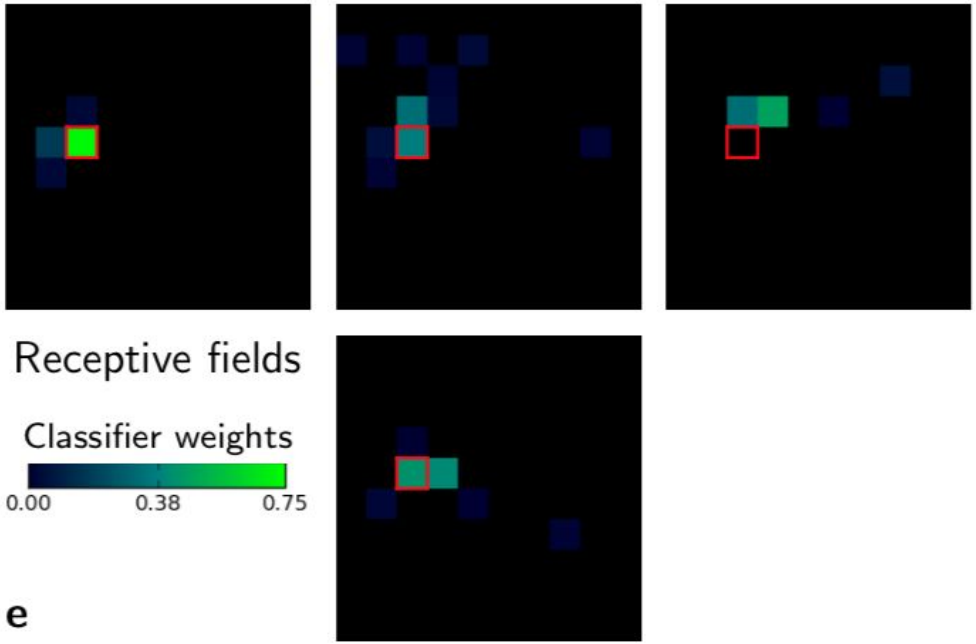
Learning a model that predicts behavioral or phenotypic variables from fMRI data

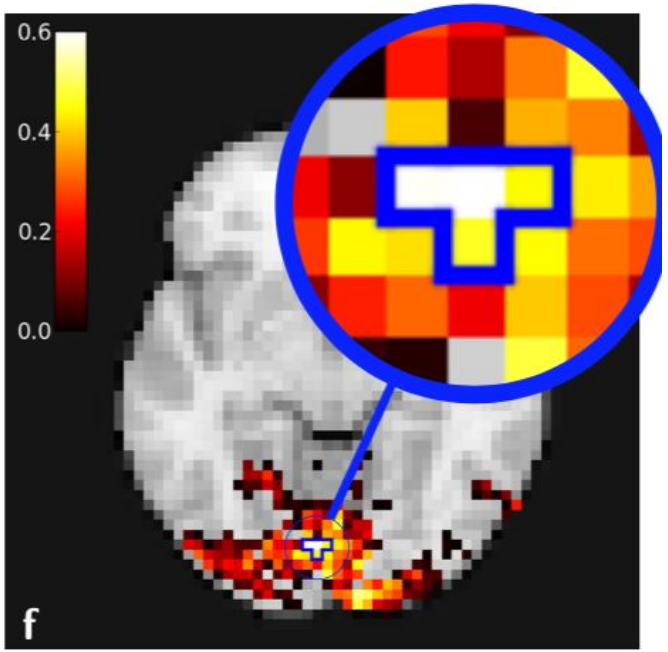


Cross validation:
Logistic Regression mean accuracy: 0.675474
SVC L1 mean accuracy: 0.678149
SVC L2 mean accuracy: 0.635706

Encoding

Predicting the imaging data given external variables, such as stimuli descriptors





Decoding

```
# Logistic Regression
sys.stderr.write("\tLogistic regression...")
t0 = time.time()
cache_path = os.path.join('miyawaki', 'lr_coef.npy')
if not os.path.exists(cache_path):
    lr = LR(penalty='l1', C=0.05)
    lr.fit(X_train, y_train[:, i_p])
    np.save(cache_path, lr.coef_)
lr_coef = np.load(cache_path)
sys.stderr.write(" Done (%.2fs)\n" % (time.time() - t0))

# Support Vector Classifier
sys.stderr.write("\tSupport vector classifier...")
t0 = time.time()
cache_path = os.path.join('miyawaki', 'svc_coef.npy')
if not os.path.exists(cache_path):
    svc = LinearSVC(penalty='l1', dual=False, C=0.01)
    svc.fit(X_train, y_train[:, i_p])
    np.save(cache_path, svc.coef_)
svc_coef = np.load(cache_path)
sys.stderr.write(" Done (%.2fs)\n" % (time.time() - t0))
```

Encoding

```
### Very simple encoding using ridge regression

from sklearn.linear_model import Ridge
from sklearn.cross_validation import KFold

print("Do ridge regression")
estimator = Ridge(alpha=100.)
cv = KFold(len(y_train), 10)
predictions = [
        Ridge(alpha=100.).fit(y_train.reshape(-1, 100)[train], X_train[train]
            ).predict(y_train.reshape(-1, 100)[test]) for train, test in cv]

print("Scoring")
scores = [1. - (((X_train[test] - pred) ** 2).sum(axis=0) /
           ((X_train[test] - X_train[test].mean(axis=0)) ** 2).sum(axis=0))
for pred, (train, test) in zip(predictions, cv)]
```

Sources

Miyawaki et. al. 2004. “Visual Image Reconstruction from Human Brain Activity using a Combination of Multiscale Local Image Decoders” *Neuron*.