Python Tools
for
Coding
and
Feature Learning

Leif Johnson
The University of Texas at Austin
Department of Computer Science
www.cs.utexas.edu/~leif

SciPy 2013
Coding?

Lindsey Bieda: flickr.com/photos/85398171@N00/2280385549 CC-BY-SA
Coding?
Coding?

tatjavanvark.nl/tvv1/pht23.html
Coding!

{frequent, complex} things $\rightarrow$ {short, simple} things.

txtese

- OMG ROFL
- <3 :-)

Telegraphese (Western Union Code Book, 1901)

- Demittebar: After — — was elected.
- Demiurgio: Before — — was elected.
- Demiurgo: Can — — be elected?
- Demnach: Can easily be elected.
- Demoanseen: Cannot be elected.
- Demobilise: Elected by.
Why code things?

Easier classification!

Science!

(Olshausen & Field, 1996)
Take a deep breath
Formal definitions: data

Data are $N$–dimensional vectors, $x \in \mathbb{R}^N$.

Assume we have $M$ of them lying around, $X \in \mathbb{R}^{N \times M}$.

**Setup**

```python
import numpy as np
import numpy.random as rng
import sklearn.decomposition as skd

X = rng.randn(N, M)
```
Formal definitions: codebook

Codebook (aka dictionary) is a matrix $D \in \mathbb{R}^{N \times K}$.

Columns are called basis vectors, basis functions, etc.

Setup

\[
D = \text{rng.randn}(N, K)
\]
Coding: projection

Project $x$ onto the columns of $D$.

$$z = D^T x$$

Preserves dimensionality of data manifold.

```
numpy
Z = np.dot(D.T, X)
```
Project $x$ onto $D$, but threshold the results.

$$z = \max(0, D^\top x - b)$$

Expands dimensionality of data manifold!

**NumPy**

```python
Z = np.dot(D.T, X)
Z[Z - b < 0] = 0
```

**scikit-learn**

```python
Z = skd.sparse_encode(X.T, D.T, alpha=b, algorithm='threshold')
```
Coding: matching pursuit (Mallat & Zhang, 1996)

Repeatedly remove best-matching basis vector from residual.

\[ c = D^\top r_t; \quad i = \arg\max |c|; \quad z_i = c_i; \quad r_{t+1} \leftarrow r_t - c_i D_i \]

**SPAMS**

\[ \text{af} = \text{np.asfortranarray} \]
\[ Z = \text{spams.omp}(\text{af}(X), \text{af}(D), 3) \]

**scikit-learn**

\[ Z = \text{skd.sparse_encode}( \text{X.T, D.T, algorithm=’omp’, n_nonzero_coefs=3}) \]
Coding: sparse coding \cite{Tibshirani1996, Efron2004}

Compute an optimal code to reconstruct the data.

\[
    z = \arg\min_w \|Dw - x\|_2^2 + \lambda_1 \|w\|_1
\]

**scipy**

\[
    Z = \text{scipy.optimize.minimize}(\lambda Z: 0.1 \times \text{abs}(Z).\text{sum()} + \\
    \text{np.linalg.norm(np.dot(D, Z) - X)} + \text{np.zeros((M, K)))}
\]

**scikit-learn**

\[
    Z = \text{skd.sparse_encode}(\text{X.T, D.T, alpha=0.1})
\]
So much for coding!

Shenghung Lin: flickr.com/photos/40764207@N00/290865741 CC-BY-NC-ND
How do we get the codebook?

Build it by hand?
How do we get the codebook?

Learn it from data!

(Olshausen & Field, 1996)  (Smith & Lewicki, 2006)
PCA

Compute maximal-variance orthonormal vector space.

\[ X = U \Sigma V^\top; \quad D = U \]

Standard, but never overcomplete. Possible structure mismatch.

**scikit-learn**

```python
m = skd.PCA()
m.fit(X.T)
D = m.components_.T
```
Samples

Sample a codebook from the data!

Very fast, but not very good unless $K \gg 0$.

```python
numpy

idx = np.arange(len(X))
rng.shuffle(idx)
D = X[idx[:K]]
```
**K-means**  (MacQueen, 1967)

Compute cluster centroids that minimize total distortion.

\[
\ell(D) = \sum_{i=1}^{M} \|X_i - D_{\gamma(X_i)}\|_2^2
\]

where \(\gamma(s) = \arg\min_u \|s - D_u\|_2^2\)

**scipy**

```python
D = scipy.cluster.vq.kmeans(X.T, 8).T
```

**scikit-learn**

```python
D = sklearn.cluster.k_means(X.T, 8)[0].T
```
Sparse coding (Mairal et al., 2009)

Use the sparse coding cost function to optimize \( D \) and \( Z \).

\[
\ell(D, Z) = \|DZ - X\|_2^2 + \lambda_1 \|Z\|_1
\]

**SPAMS**

\[
D = \text{spams.trainDL}(\text{af}(X), k=8, \lambda_1=0.1)
\]

**scikit-learn**

\[
Z, D, _ = \text{skd.dict_learning}(X^T, 8)
\]

\[
Z = Z^T
\]

\[
D = D^T
\]
Restricted Boltzmann Machines (Hinton, 2002)

Compute features that preserve distribution of data.

Given $x$, propagate activity up:

$$p(z|x) \sim \text{Bernoulli}(\sigma(D^\top x))$$

Given $z$, propagate activity down:

$$p(x|z) \sim \text{Bernoulli}(\sigma(Dz))$$
Restricted Boltzmann Machines (Hinton, 2002)

Learn features using Contrastive Divergence.

\[ \Delta D \propto [x_0 z_0^\top] - [x_\infty z_\infty^\top] \approx [x_0 z_0^\top] - [x_1 z_1^\top] \]

- MORB: modular RBM toolkit
- lmj.rbm: example code using numpy
Restricted Boltzmann Machines (Hinton, 2002)

Theano: a very useful tool, but a bit of a head-warp!

```python
import theano, theano.tensor as TT

x = TT.matrix()
 rng = TT.shared_randomstreams.RandomStreams()

D = theano.shared(rng.randn(N, K))

z_mean = TT.nnet.sigmoid(TT.dot(D.T, x))
 z = rng.uniform(size=z_mean.shape, lo=0, hi=1) < z_mean
xhat_mean = TT.nnet.sigmoid(TT.dot(D, z))
xhat = rng.uniform(size=xhat_mean.shape, lo=0, hi=1) < xhat_mean

updown = theano.function([x], [z_mean, z, xhat_mean, xhat])
```
Autoencoders (e.g., Vincent et al., 2008)

Compute features that minimize a nonlinear cost function.

\[ \hat{x} = Dz = D\sigma(D^\top x) \]

Think of this like an “unfolded” RBM.
**Autoencoders** (e.g., Vincent et al., 2008)

Compute features that minimize a nonlinear cost function.

\[ \ell(D) = \| D\sigma(D^\top x - x) \|_2^2 + \alpha \| D \|_2^2 + \beta \| \sigma(D^\top x) \|_1 \]

\[ \alpha \text{ controls the size of the codebook weights.} \]

```python
lmj.nn

exp = lmj.nn.Experiment(
    lmj.nn.Autoencoder, optimize='hf',
)
exp.train(X.T)
D = exp.network.weights[0].get_value()
```
**Autoencoders** (e.g., Vincent et al., 2008)

Compute features that minimize a nonlinear cost function.

\[
\ell(D) = \|D\sigma(D^\top x) - x\|_2^2 + \alpha\|D\|_2^2 + \beta\|D\sigma(D^\top x)\|_1
\]

\(\alpha\) controls the size of the codebook weights.

```python
lmj.nn

exp = lmj.nn.Experiment(
    lmj.nn.Autoencoder, optimize='hf',
    weight_l2=0.1)
exp.train(X.T)
D = exp.network.weights[0].get_value()
```
Autoencoders (e.g., Vincent et al., 2008)

Compute features that minimize a nonlinear cost function.

\[ \ell(D) = \| D \sigma(D^T x) - x \|_2^2 + \alpha \| D \|_2^2 + \beta \| \sigma(D^T x) \|_1 \]

\( \beta \) controls the sparsity of the code.

```python
lmj.nn

exp = lmj.nn.Experiment(
    lmj.nn.Autoencoder, optimize='hf',
    weight_l2=0.1, hidden_l1=0.1)
exp.train(X.T)
D = exp.network.weights[0].get_value()
```
Autoencoders (e.g., Vincent et al., 2008)

Compute features that minimize a nonlinear cost function.

\[ \ell(D) = \| D \sigma(D^T(x + \epsilon)) - x \|_2^2 + \alpha \| D \|_2^2 + \beta \| \sigma(D^T x) \|_1 \]

\( \epsilon \) can be used to add noise to \( x \) (denoising autoencoder).

```python
lmj.nn

exp = lmj.nn.Experiment(
    lmj.nn.Autoencoder, optimize='hf',
    weight_l2=0.1, hidden_l1=0.1, input_noise=0.1)
exp.train(X.T)
D = exp.network.weights[0].get_value()
```
Coding vs Learning

Coates & Ng, 2011

Many learning algorithms produce qualitatively similar results!

Coding algorithm plays an important role (for classifiers).
Package pointers

Scikit-Learn: scikit-learn.org
Theano: deeplearning.net/tutorial

Sparse coding
• scikit-learn.org/modules/decomposition.html
• spams-devel.gforge.inria.fr

Restricted Boltzmann Machines
• github.com/lmjohns3/py-rbm
• github.com/benanne/morb
• github.com/deeplearningais/CUV

Neural Networks
• github.com/lmjohns3/theano-nets
• pythonhosted.org/neurolab
• pybrain.org
Thank you!