

# Example: Faces

---

Labeled Faces in the Wild Home



- ❑ Face images can be high-dimensional
  - We will use  $50 \times 37 = 1850$  pixels
- ❑ But, there may be few degrees of freedom
- ❑ Can we reduce the dimensionality of this?
- ❑ Data Labelled Faces in the Wild project
  - <http://vis-www.cs.umass.edu/lfw>
  - Large collection of faces (13000 images)
  - Taken from web articles about 10 years ago

# Loading the Data

- ❑ Built-in routines to load data is scikit-learn
- ❑ Can take several minutes the first time (Be patient)

Image size = 50 x 37 = 1850 pixels  
Number faces = 1288  
Number classes = 7

```
from sklearn.datasets import fetch_lfw_people
lfw_people = fetch_lfw_people(min_faces_per_person=70, resize=0.4)

2016-11-14 14:15:30,862 Downloading LFW metadata: http://vis-www.cs.umass.edu/lfw/pairsDevTrain.txt
2016-11-14 14:15:30,958 Downloading LFW metadata: http://vis-www.cs.umass.edu/lfw/pairsDevTest.txt
2016-11-14 14:15:31,028 Downloading LFW metadata: http://vis-www.cs.umass.edu/lfw/pairs.txt
2016-11-14 14:15:31,294 Downloading LFW data (~200MB): http://vis-www.cs.umass.edu/lfw/lfw-funneled.tgz
2016-11-14 14:20:10,056 Decompressing the data archive to C:\Users\Sundeep\scikit_learn_data\lfw_home\lfw_funneled
2016-11-14 14:22:08,605 Loading LFW people faces from C:\Users\Sundeep\scikit_learn_data\lfw_home
2016-11-14 14:22:09,735 Loading face #00001 / 01288
2016-11-14 14:22:13,640 Loading face #01001 / 01288
```

# Plotting the Data

- ❑ Some example faces
- ❑ You may be too young to remember them all



```
def plt_face(x):
    h = 50
    w = 37
    plt.imshow(x.reshape((h, w)), cmap=plt.cm.gray)
    plt.xticks([])
    plt.yticks([])

I = np.random.permutation(n_samples)
plt.figure(figsize=(10,20))
nplt = 4;
for i in range(nplt):
    ind = I[i]
    plt.subplot(1,nplt,i+1)
    plt_face(X[ind])
    plt.title(target_names[y[ind]])
```

# Computing the SVD

```
npix = h*w  
Xmean = np.mean(X, 0)  
Xs = X - Xmean[None, :]
```

❑ Note efficient use of python broadcasting

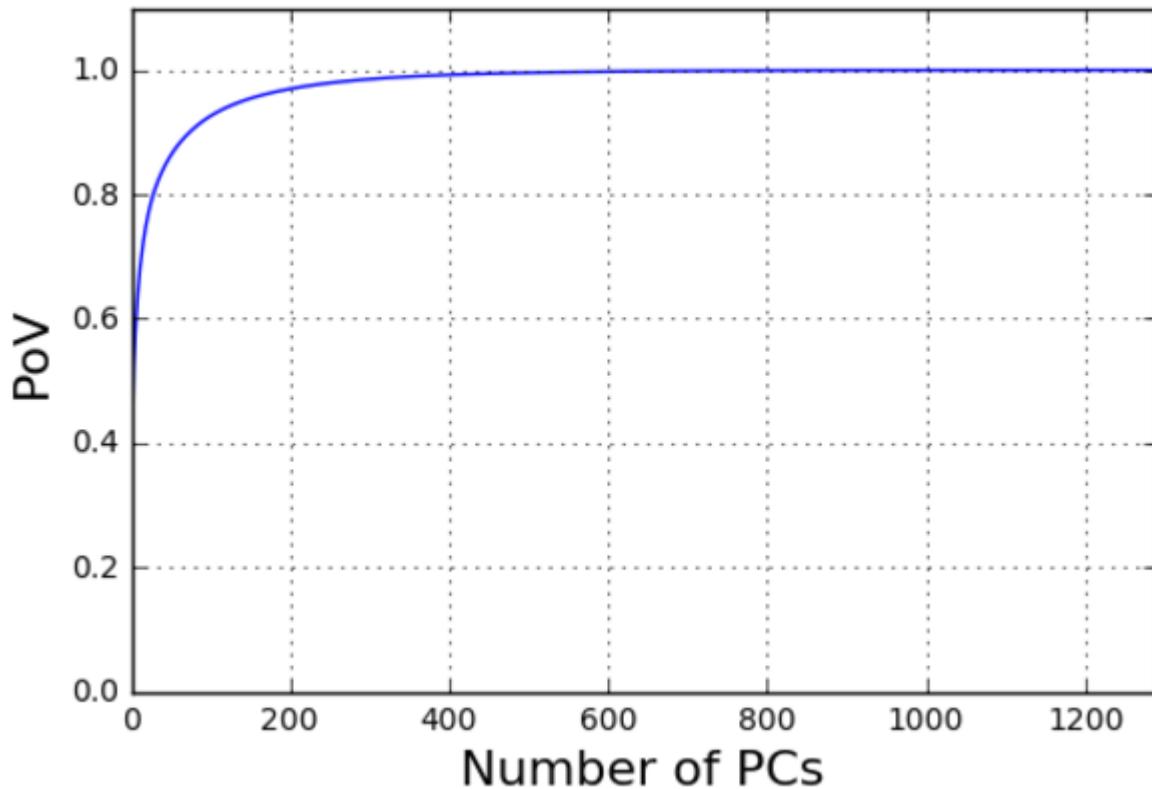
Then, we compute an SVD. Note that in python the SVD re

```
U,S,V = np.linalg.svd(Xs, full_matrices=False)
```

❑ SVD:

- Use full\_matrices (avoids computing zero SVs)
- Matrix V is what we call  $V^T$   
(Different from MATLAB)

# Finding the PoV



- ❑ Most variance explained in about 400 components
- ❑ Some reduction

```
lam = S**2  
PoV = np.cumsum(lam)/np.sum(lam)  
  
plt.plot(PoV)  
plt.grid()  
plt.axis([1,n_samples,0, 1.1])  
plt.xlabel('Number of PCs', fontsize=16)  
plt.ylabel('PoV', fontsize=16)
```

# Plotting Approximations

```
nplt = 2          # number of faces to plot
ds = [0,5,10,20,100] # number of SVD approximations

# Select random faces
inds = np.random.permutation(n_samples)
inds = inds[:nplt]
nd = len(ds)

# Set figure size
plt.figure(figsize=(1.8 * (nd+1), 2.4 * nplt))
plt.subplots_adjust(bottom=0, left=.01, right=.99, top=.90, hspace=.35)

# Loop over figures
iplt = 0
for ind in inds:
    for d in ds:
        plt.subplot(nplt,nd+1,iplt+1)
        Xhati = (U[ind,:,:d]*S[None,:,:d]).dot(V[:,d,:]) + Xmean
        plt_face(Xhati)
        plt.title('d={0:d}'.format(d))
        iplt += 1

    # Plot the true face
    plt.subplot(nplt,nd+1,iplt+1)
    plt_face(X[ind,:])
    plt.title('Full')
    iplt += 1
```

❑ Selection of figure sizes for subplot

❑ Note: Efficient computing of approx



# Plotting the Approximations

---

