cs178-hw5s

March 16, 2016

1 Homework 5 solutions (raw)

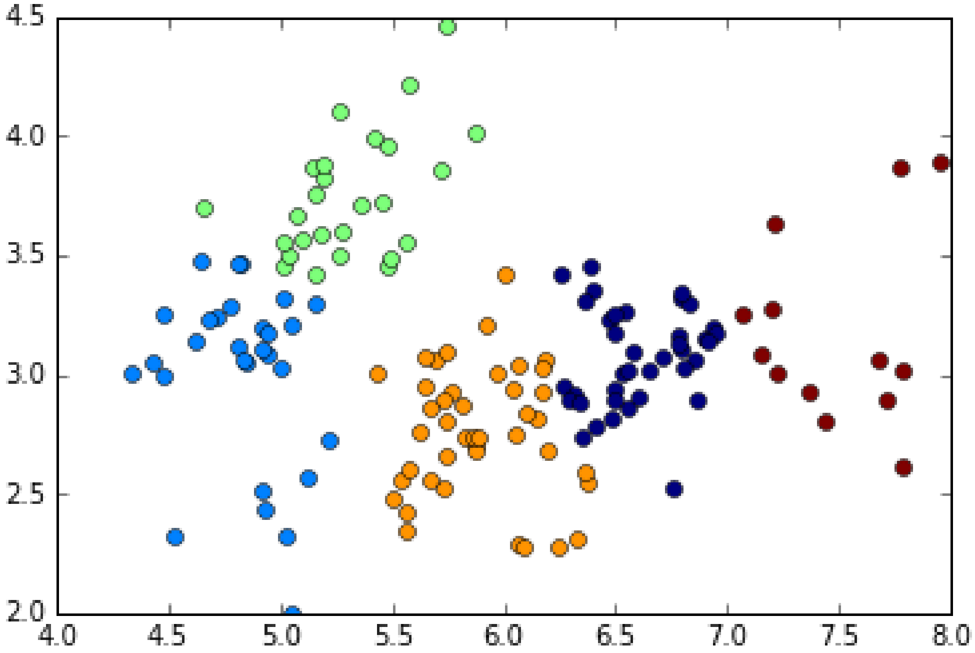
```
In [1]: import numpy as np
        np.random.seed(0)
        import mltools as ml
        import matplotlib.pyplot as plt   # use matplotlib for plotting with inline plots
        %matplotlib inline
```

```
In [13]: import mltools.cluster as clust
         reload(clust);
```

```
In [14]: iris = np.genfromtxt("data/iris.txt",delimiter=None)
         Y = iris[:,-1]
         X = iris[:,0:2]
         print X.shape
```

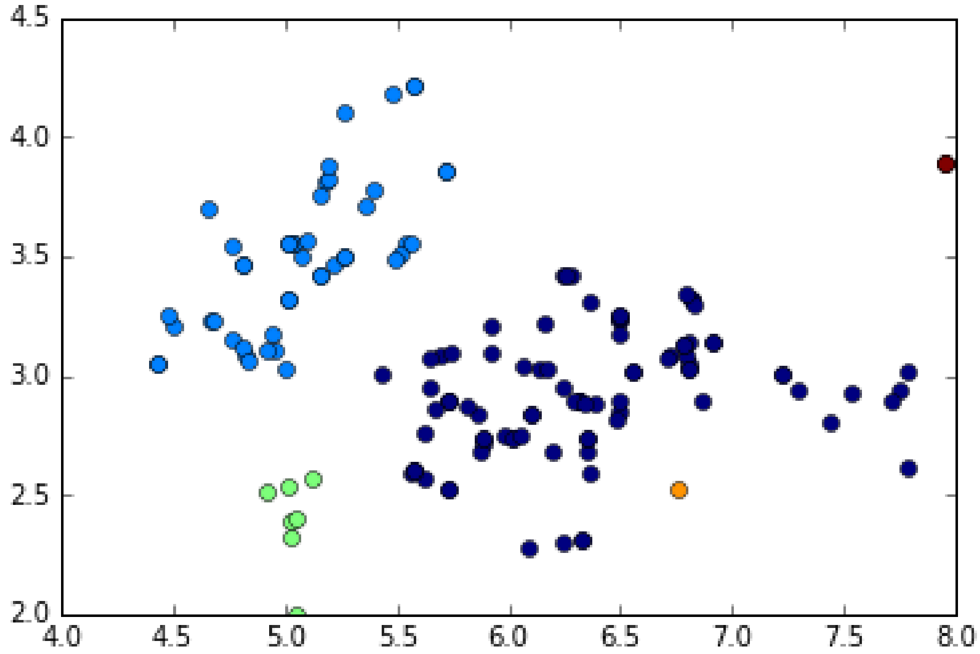
```
(148, 2)
```

```
In [16]: ssd = np.inf
         for it in range(10):
             Zi,mui,ssdi = clust.kmeans(X,K=5,init='random')
             if ssdi < ssd:
                 Z,mu,ssd = Zi,mui,ssdi
         # Now, plot the data and their cluster ID as color:
         ml.plotClassify2D(None,X,Z)

         # same thing again k=20
```

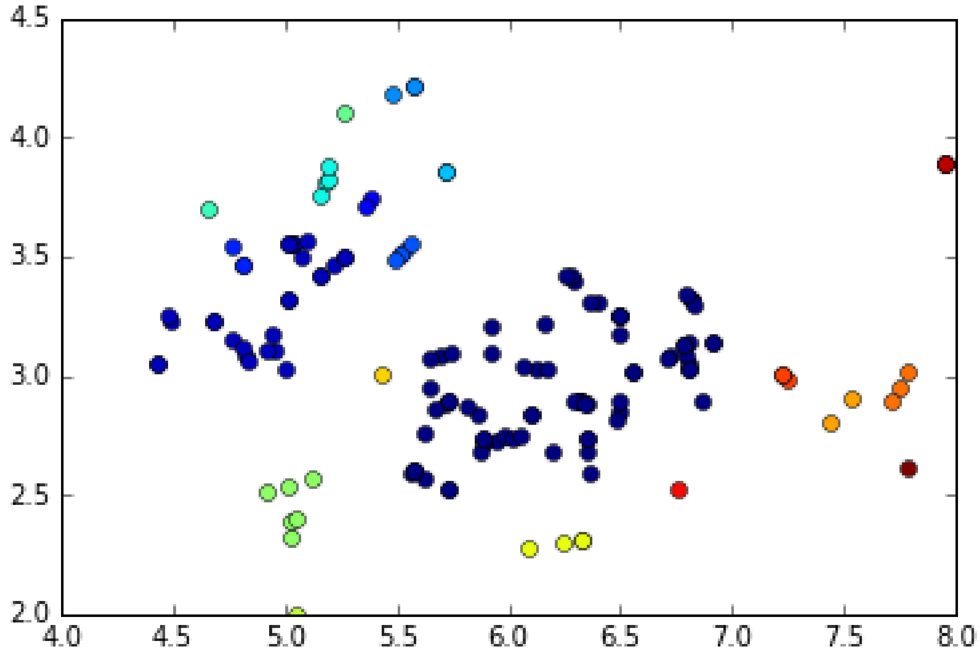


```
In [31]: Z,dend = clust.agglomerative(X, K=5, method='min')
         plt.figure()
         print "Single linkage, K=5"
         ml.plotClassify2D(None,X,Z)
         plt.show()

         Z,dend = clust.agglomerative(X, K=20, method='min')
         plt.figure()
         print "Single linkage, K=20"
         ml.plotClassify2D(None,X,Z)
         plt.show()

         Z,dend = clust.agglomerative(X, K=5, method='max')
         plt.figure()
         print "Complete linkage, K=5"
         ml.plotClassify2D(None,X,Z)
         plt.show()

         Z,dend = clust.agglomerative(X, K=20, method='max')
         plt.figure()
         print "Complete linkage, K=20"
         ml.plotClassify2D(None,X,Z)
         plt.show()
```

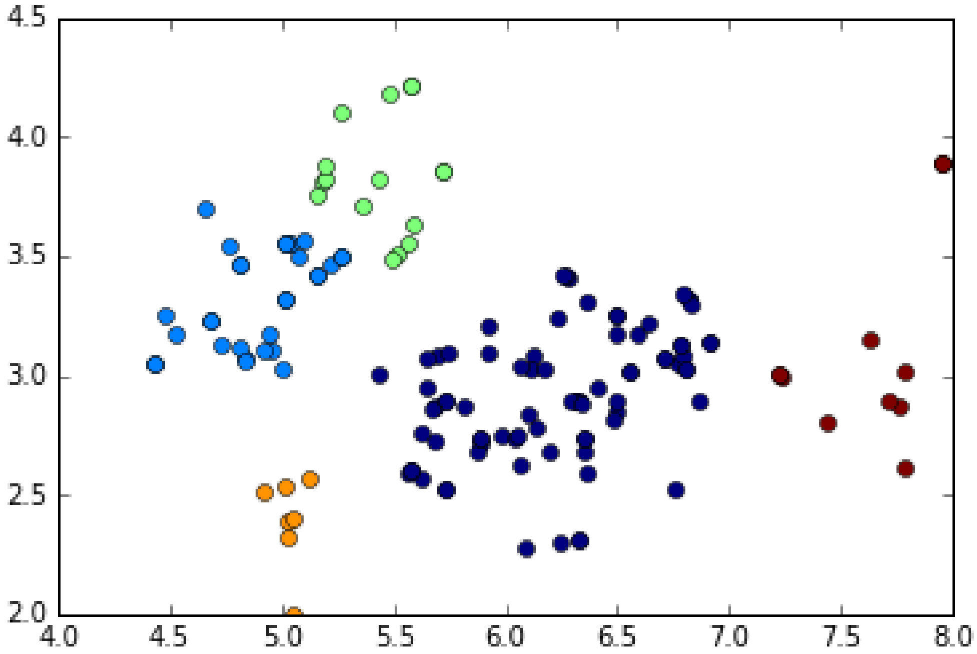
```
Single linkage, K=5
```



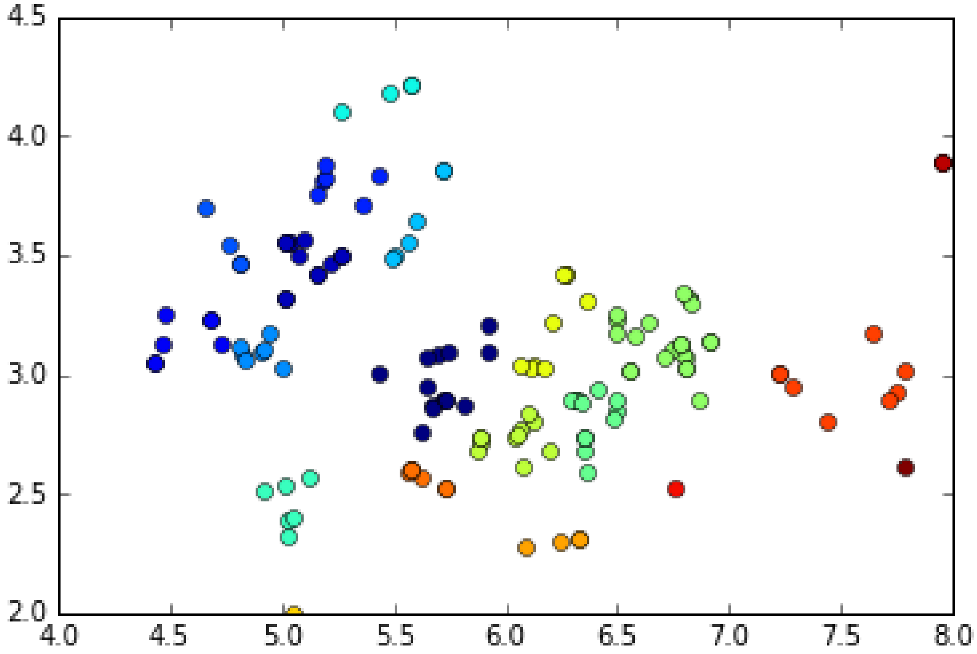
Single linkage, K=20



Complete linkage, K=5

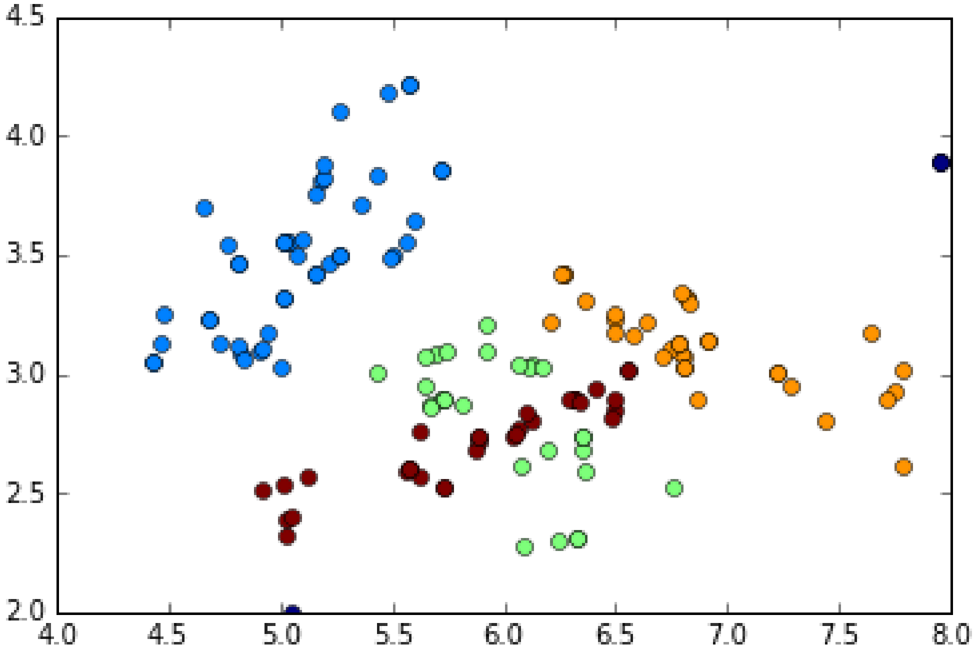


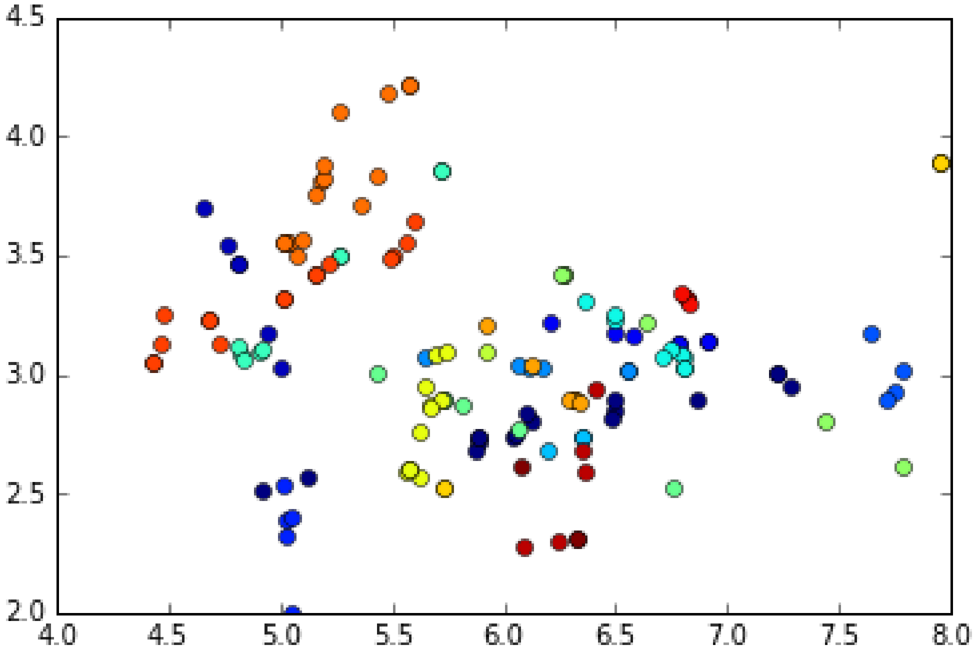
Complete linkage, K=20



```
In [46]: Z,params,prob,LL = clust.gmmEM(X,K=5,init='k++')
         ml.plotClassify2D(None,X,Z)
```

```
plt.show()

Z,params,prob,LL = clust.gmmEM(X,K=20,init='k++')
ml.plotClassify2D(None,X,Z)
plt.show()
```



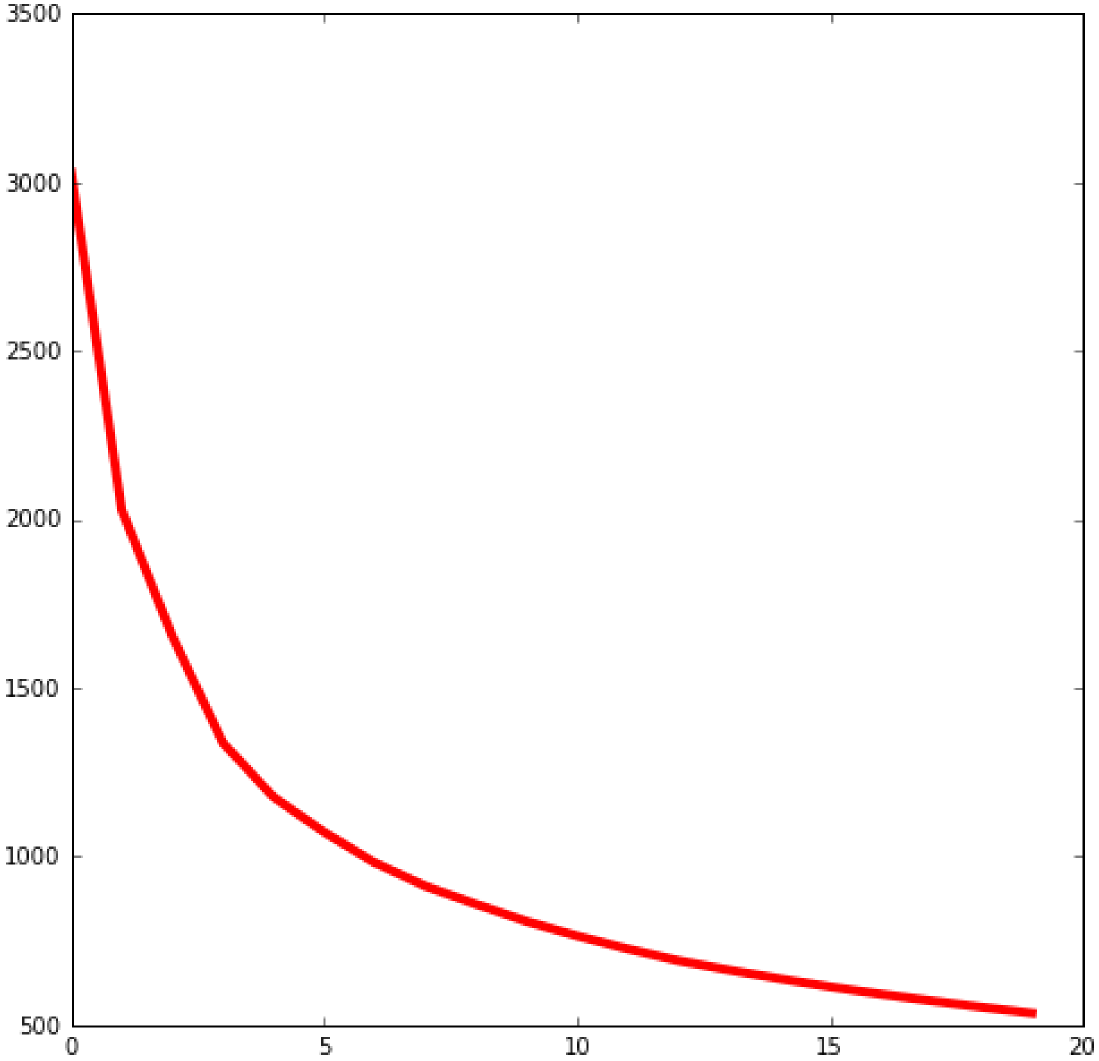


1.1 Eigenfaces

```
In [29]: import scipy.linalg
         # (a)--(b) : Load the data and take the SVD:
         X = np.genfromtxt("data/faces.txt",delimiter=None)   # load face dataset
         mu = X.mean(axis=0, keepdims=True)
         X0 = X - mu     # remove the mean
         U,S,V = scipy.linalg.svd(X0, False)  # ??? puts singular vectors in descending order
         W = U.dot( np.diag(S) );
         print U.shape, S.shape, V.shape
```

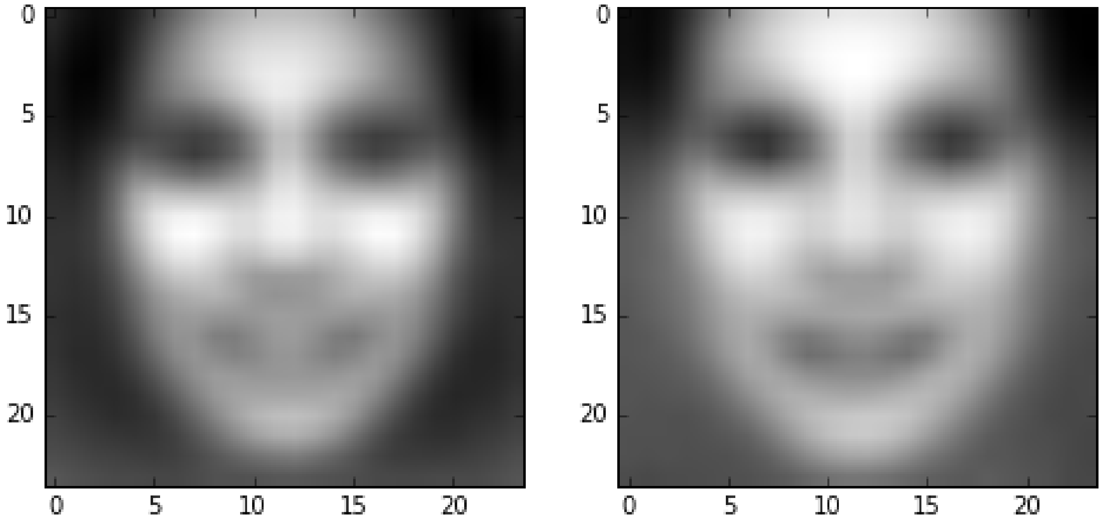
```
(4916, 576) (576,) (576, 576)
```

```
In [30]: # (c) Let's look at the reconstruction error as a function of the number of components
         err = [None]*20
         for k in range(20):
             Xhat0 = W[:,:k].dot( V[:k,:] )
             err[k] = ((X0-Xhat0)**2).mean()

         plt.plot(range(20),err,'r-',linewidth=4);
```

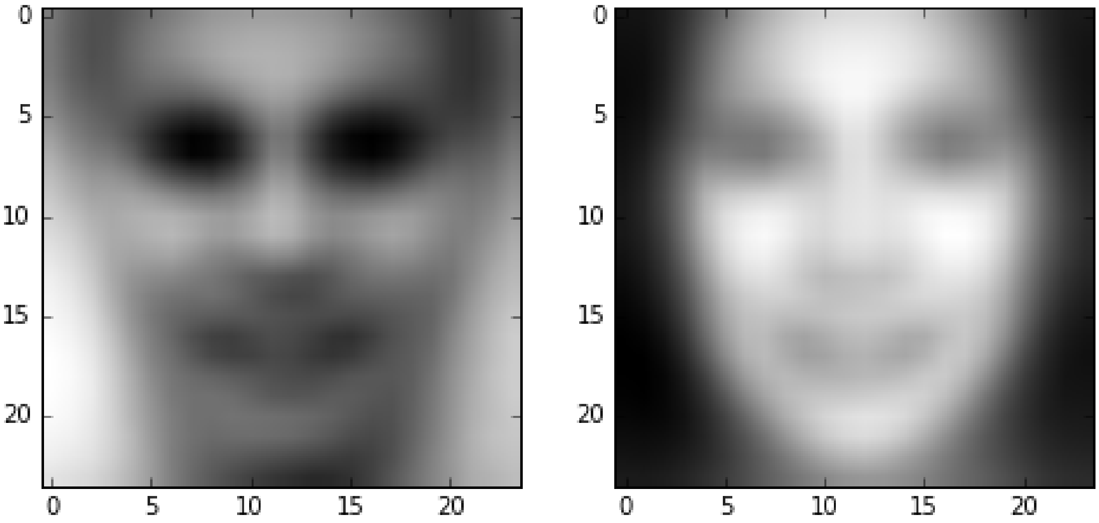


```
In [31]: # (d) Let's look at how the faces vary along each principal component
         for k in range(5):
             alpha = 2*np.median( np.abs( W[:,k] ));
             im1 = np.reshape(mu + alpha*V[k,:], (24,24));
             im2 = np.reshape(mu - alpha*V[k,:], (24,24));
             # TODO: subplots
             plt.figure();
             f,(ax1,ax2) = plt.subplots(1,2);
             ax1.imshow(im1.T, cmap="gray");
             ax2.imshow(im2.T, cmap="gray");
```

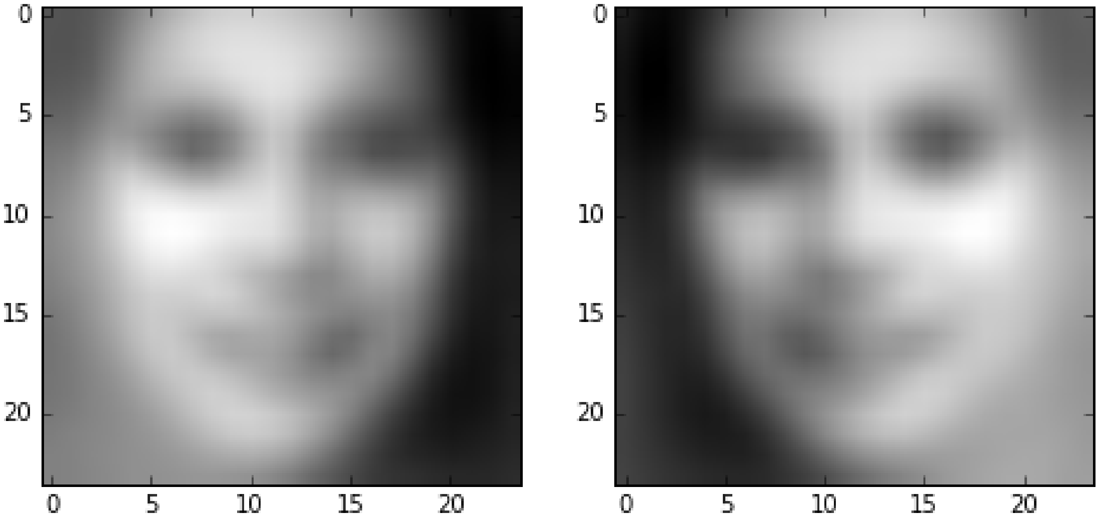
```
<matplotlib.figure.Figure at 0x1159e7a10>
```



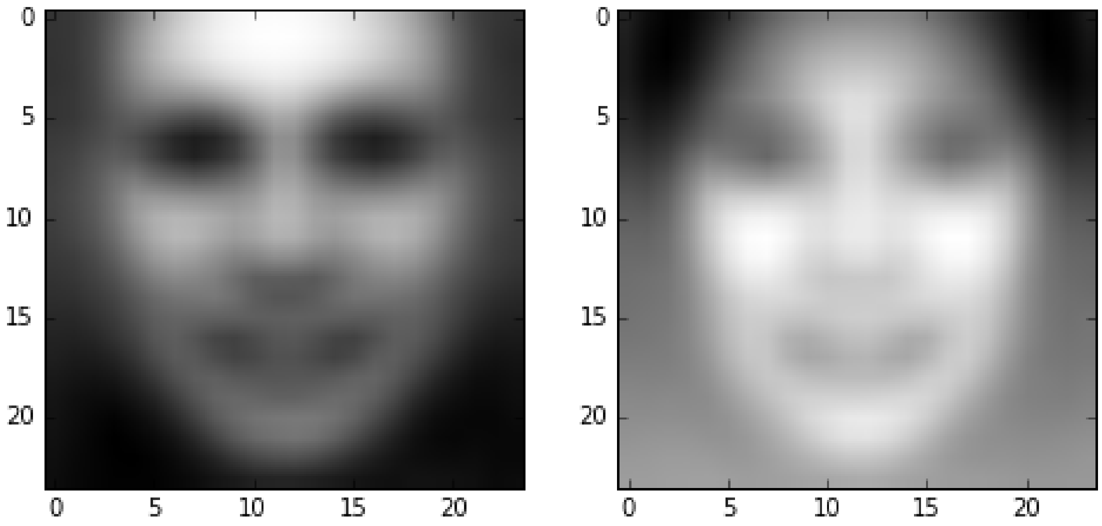
```
<matplotlib.figure.Figure at 0x119197150>
```


0
5
10
15
20
0
5
10
15
20
0
5
10
15
20
0
5
10
15
20

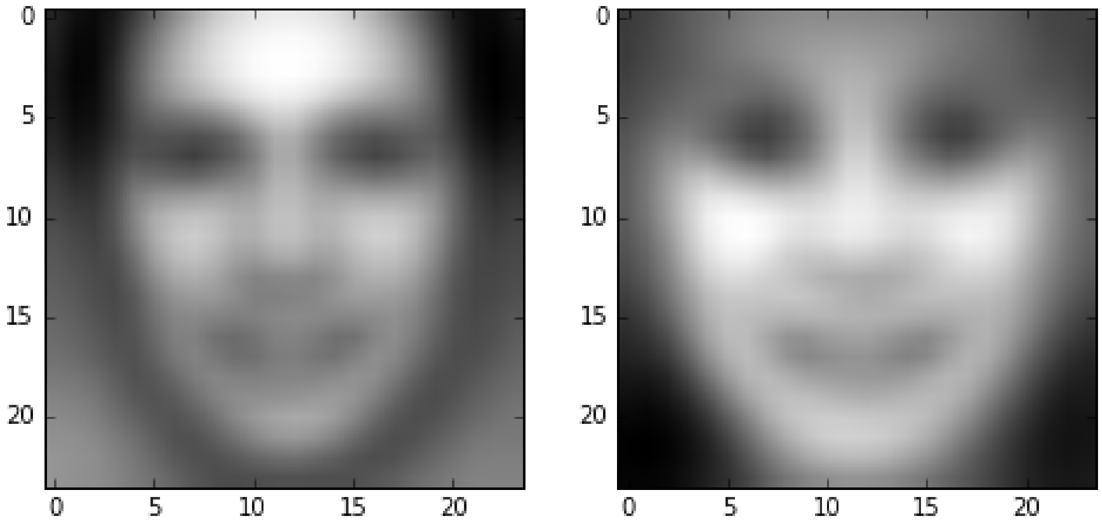
```
<matplotlib.figure.Figure at 0x119d83b50>
```


0
5
10
15
20
0
5
10
15
20
0
5
10
15
20
0
5
10
15
20

```
<matplotlib.figure.Figure at 0x115a6c110>
```

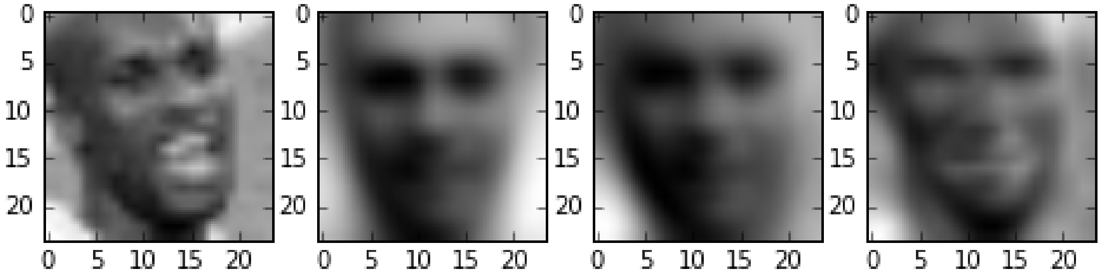


```
<matplotlib.figure.Figure at 0x115b8a410>
```

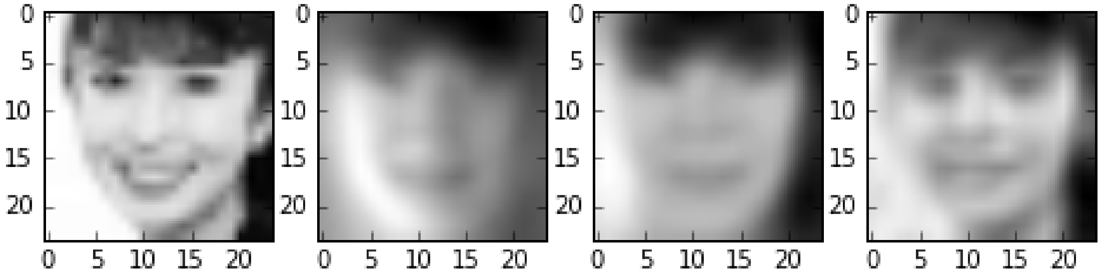


```
In [32]: # (e) Reconstruct two faces using a few components
         for i in [16,24,35,70]:
             im = X[i,:];
             im = np.reshape(im, (24,24));
             plt.figure()
             f,ax = plt.subplots(1,4);
             ax[0].imshow(im.T, cmap="gray");
             for j,k in enumerate([5,10,50]):
                 im = mu + W[i,0:k].dot( V[0:k,:] );
                 im = np.reshape(im, (24,24));
                 ax[j+1].imshow(im.T, cmap="gray");
```

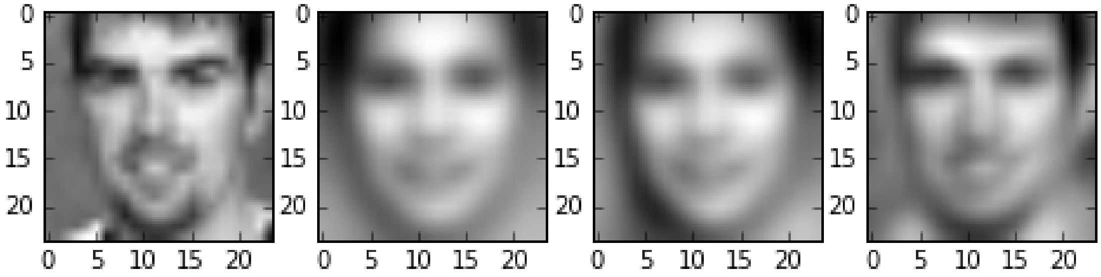
```
<matplotlib.figure.Figure at 0x11a24b310>
```


0
5
10
15
20

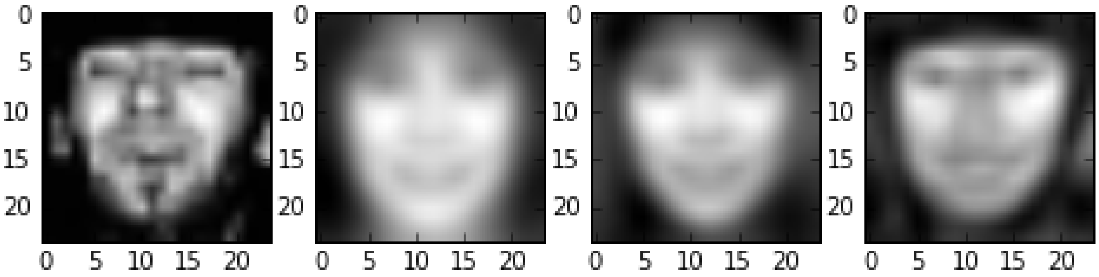
```
<matplotlib.figure.Figure at 0x115ad8710>
```


0
5
10
15
20

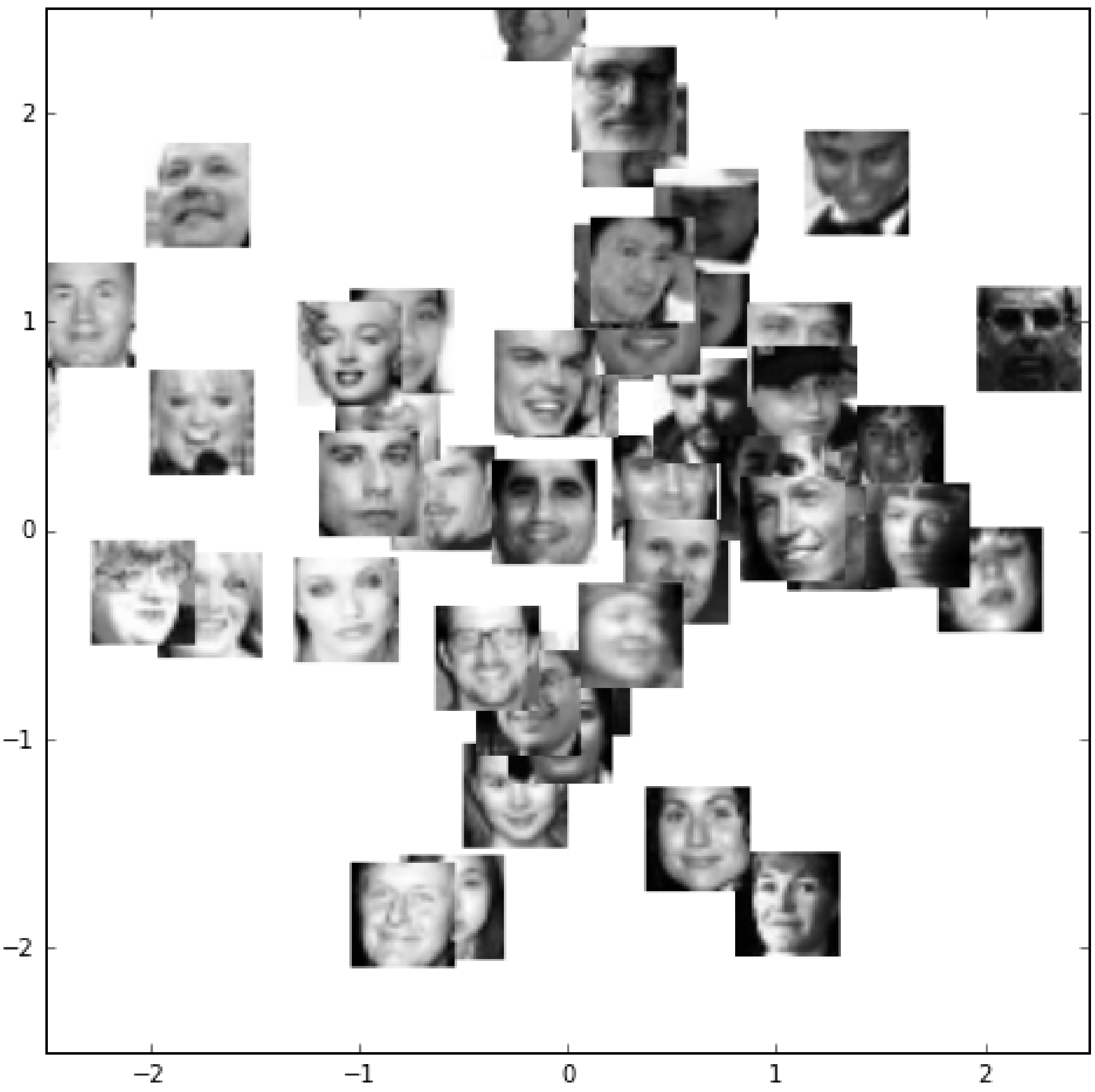
```
<matplotlib.figure.Figure at 0x118eab910>
```


0
5
10
15
20

```
<matplotlib.figure.Figure at 0x10d27dbd0>
```



```
In [27]: # (f) Let's plot some of the faces and see what the first two dimensions look like...
         import mltools.transforms

         idx = np.floor( 4916*np.random.rand(50) ); # pick some data
         idx = idx.astype('int')

         plt.rcParams['figure.figsize'] = (8.0, 8.0)

         coord,params = ml.transforms.rescale(W[:,0:2])
         for i in idx:
             loc = (coord[i,0],coord[i,0]+.5, coord[i,1],coord[i,1]+.5)
             plt.imshow(np.reshape(X[i,:],(24,24)).T , cmap="gray", extent=loc );
             plt.axis( (-2.5,2.5,-2.5,2.5) )
```



```
In [ ]:
```