Problem 1:

1.1:

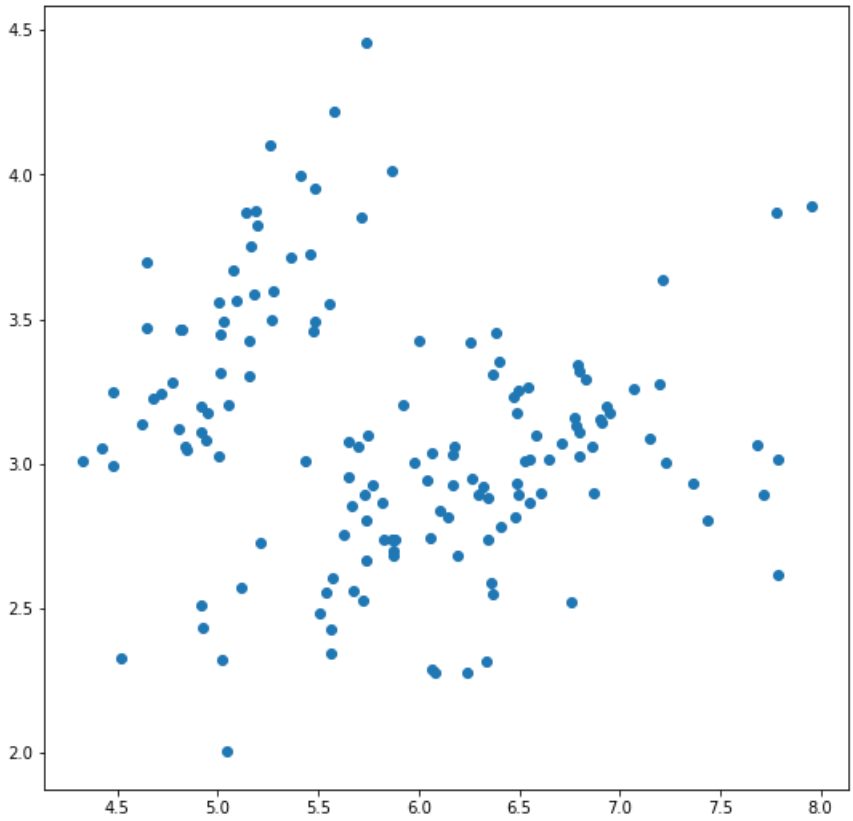
In [42]:

```
import numpy as np
import mltools as ml
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from scipy.linalg import svd

data = np.genfromtxt('data/iris.txt', delimiter = None)

X = data[:,0:2]
Y = data[:,-1]

f, ax = plt.subplots(1, 1, figsize=(9, 9))
ax.scatter(X[:,0],X[:,1])
plt.show()
```



I believe there is two clusters because there is a white space between the top portion of points and the bottom ones. I think these two group of points are the only ones that are close enough together to be considered clusters.

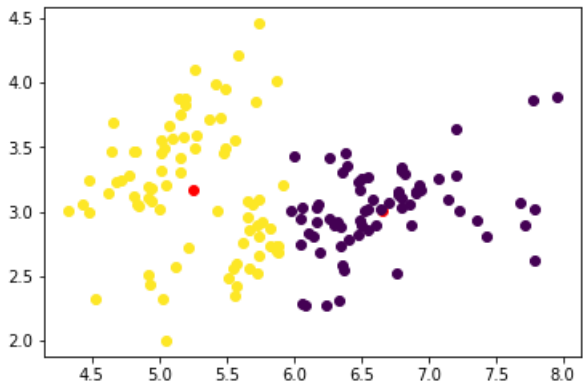
1.2:

In [32]:

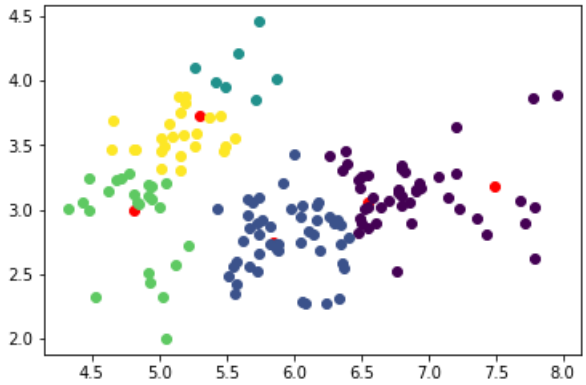
```
ks = [2, 5, 20]

for k in ks:
    print("k = " + str(k))
    c = KMeans(k).fit(X)
    u = ml.cluster.kmeans(X,k)[0]
    ml.plotClassify2D(None, X, u)
    plt.scatter(c.cluster_centers_[:,0], c.cluster_centers_[:,1], c = 'red')
    plt.show()
```

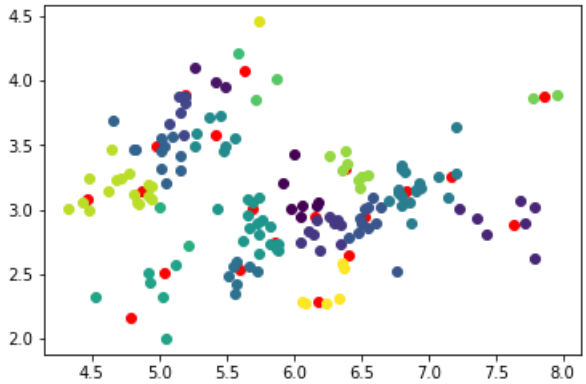
k = 2



k = 5



k = 20



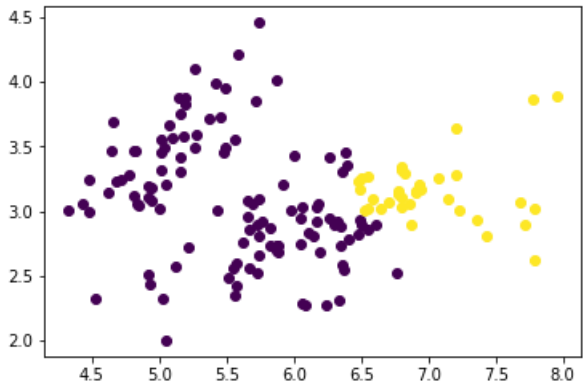
1.3:

In [34]:

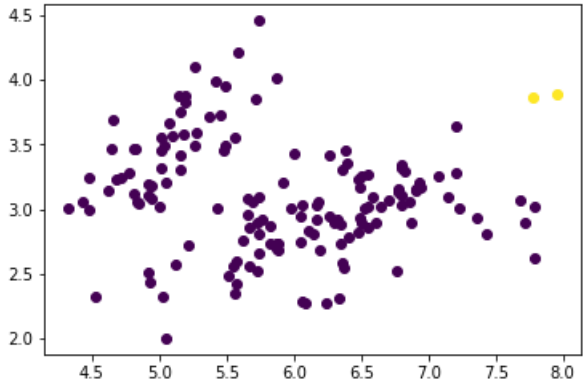
```
clusters = [2, 5, 20]
complete = "max"
single = "min"

for i in clusters:
    print("K = " + str(i))
    print("Complete Linkage")
    ml.plotClassify2D(None, X, ml.cluster.agglomerative(X, i, method = complete)[0] )
    plt.show()
    print("Single Linkage")
    ml.plotClassify2D(None, X, ml.cluster.agglomerative(X, i, method = single)[0] )
    plt.show()
```

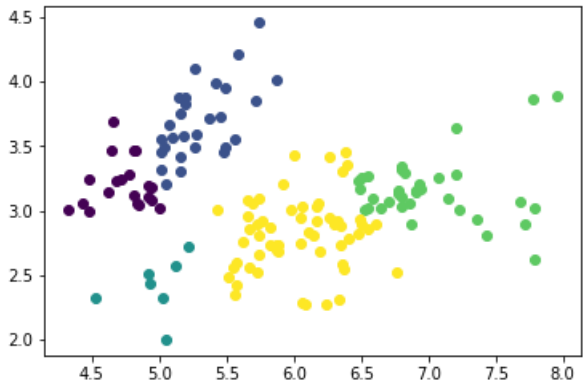
K = 2
Complete Linkage



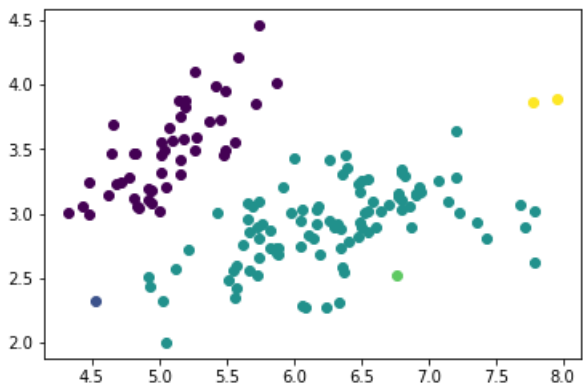
Single Linkage



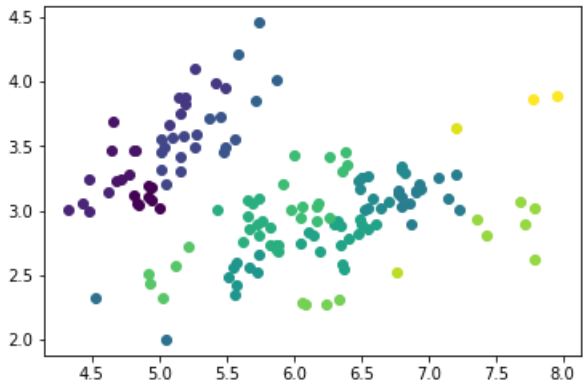
K = 5
Complete Linkage



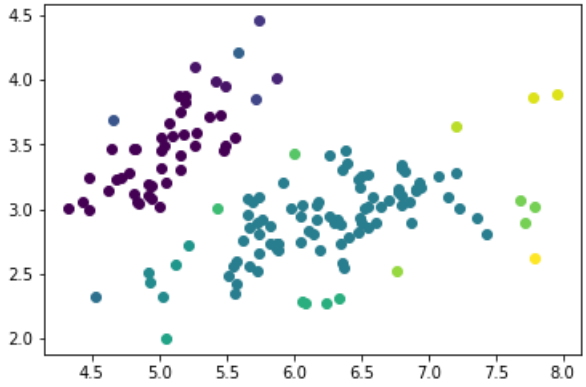
Single Linkage



```
K = 20
Complete Linkage
```




```
Single Linkage
```



1.4:

The kmeans output is similar to the agglomerative clustering algorithm in the sense that clusters are formed around the same general area amongst most outputs. I do notice that kmeans made many more clusters than agglomerative clustering. As k increases, kmeans output had many clusters. Single linkagea gglomerative clustering outputs seem to make the biggest clusters out of all the outputs. Complete linkage agglomerative clustering are very similar to kmeans outputs but seem to still have less clusters.

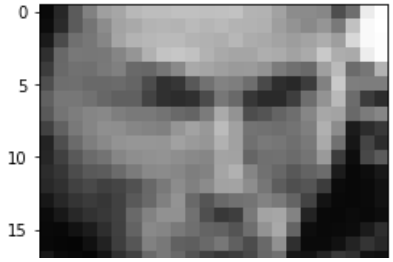
Problem 2:

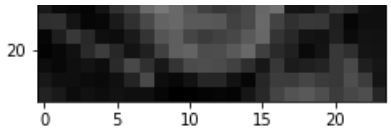
2.1:

In [52]:

```
X = np.genfromtxt("data/faces.txt", delimiter=None) # load face dataset plt.figure()
# pick a data point i for display
img = np.reshape(X[i,:],(24,24)) # convert vectorized data to 24x24 image patches
plt.imshow( img.T , cmap="gray") # display image patch; you may have to squint
plt.show()

X0 = X - np.mean(X, axis = 0)
print("Data Zero Mean: ")
print("X0" + str(x0))
```






```
Data Zero Mean:
X0[[  9.95240033  15.13161107  10.03254679 ... -18.58136697 -84.98494711
  -94.25386493]
 [-64.04759967 -65.86838893 -61.96745321 ... -19.58136697 -57.98494711
  -97.25386493]
 [-80.04759967 -76.86838893 -74.96745321 ... -35.58136697 -38.98494711
  -40.25386493]
 ...
 [-66.04759967 -64.86838893 -64.96745321 ... -72.58136697 -71.98494711
  -68.25386493]
 [-21.04759967 -17.86838893 -15.96745321 ... -69.58136697 -70.98494711
  -72.25386493]
 [-29.04759967 -30.86838893 -28.96745321 ... 152.41863303 150.01505289
  149.74613507]]
```

2.2:

In [63]:

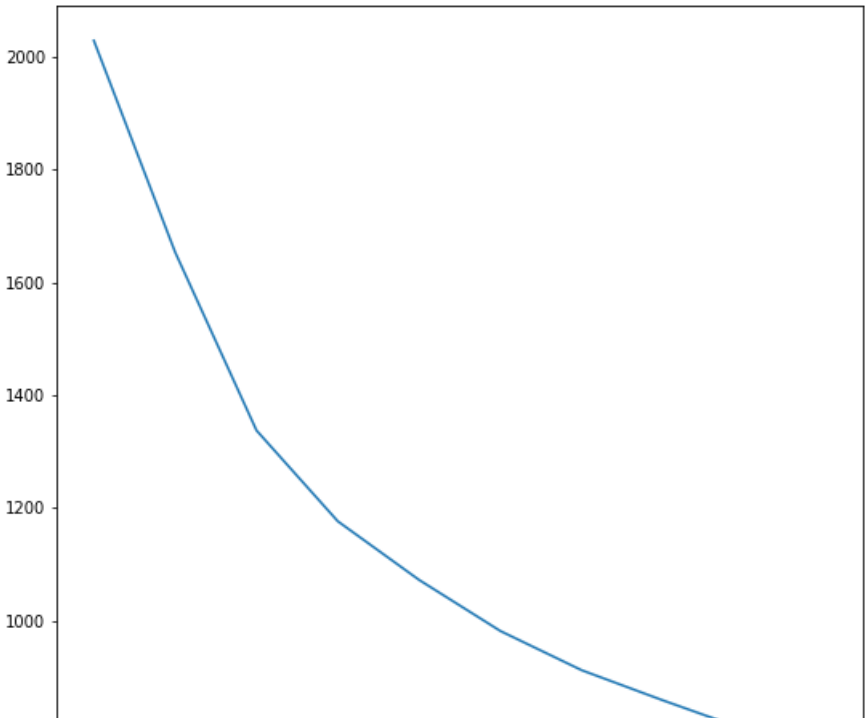
```
U, s, V = svd(X0, full_matrices=False)
W = np.dot(U, np.diag(s))
print 'Shape of W = (%d, %d)' % W.shape, 'Shape of V = (%d, %d)' % V.shape
```

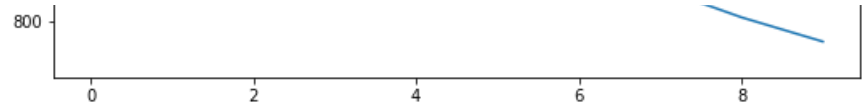
```
Shape of W = (4916, 576) Shape of V = (576, 576)
```

2.3:

In [59]:

```
K_range = [1,2,3,4,5,6,7,8,9,10]
plot_mse = []
for K in K_range:
    X0hat = np.dot(W[:, :K],V[:K,:])
    mean = np.mean((X0 - X0hat)**2 )
    plot_mse.append(mean)
f, ax = plt.subplots(1, 1, figsize=(9, 9))
ax.plot(plot_mse)
plt.show()
```





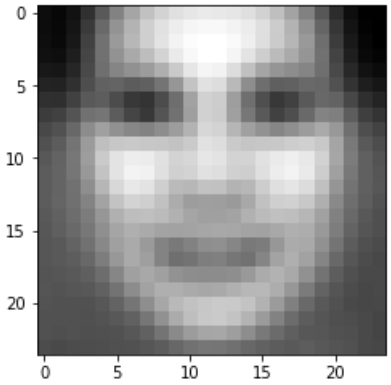
2.4:

In [77]:

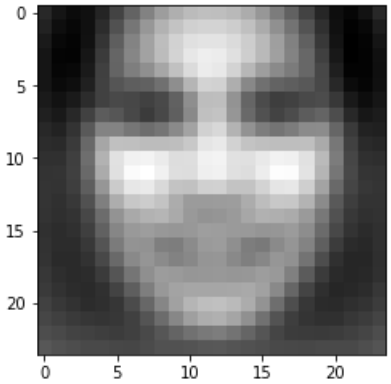
```
mu = np.mean(X, axis = 0)

for j in range(3):
    print("J = " + str(j))
    a = 2 * np.median(np.abs(W[:, j]))
    p1 = mu - a * V[j]
    p2 = mu + a * V[j]
    im1 = np.reshape(p1, [24,24])
    print("μ-α V[j,:]")
    plt.imshow(im1.T, cmap="gray")
    plt.show()
    im2 = np.reshape(p2, [24,24])
    print("μ+α V[j,:]")
    plt.imshow(im2.T, cmap="gray")
    plt.show()
```

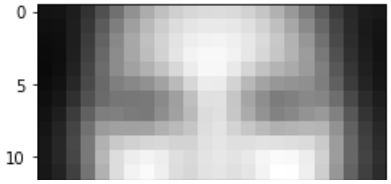
J = 0
μ-α V[j,:]

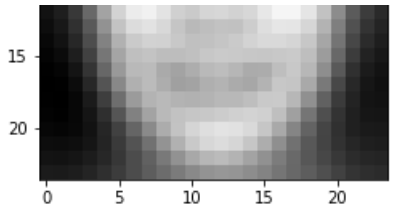


μ+α V[j,:]

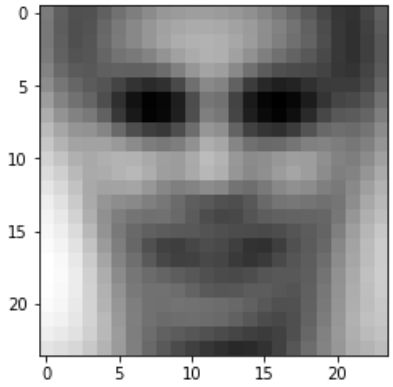


J = 1
μ-α V[j,:]

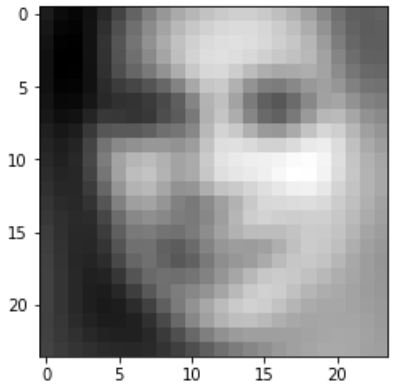




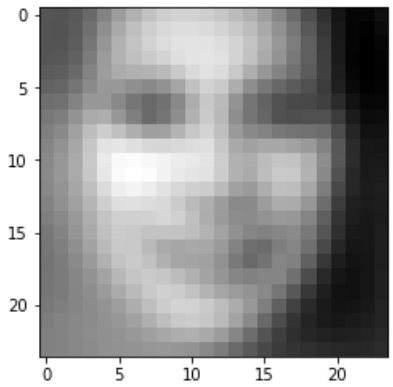
μ+α V[j,:]



J = 2
μ-α V[j,:]



μ+α V[j,:]



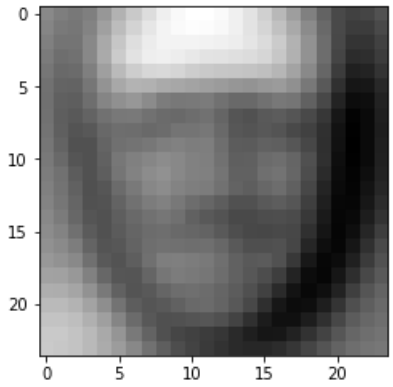
2.5:

In [88]:

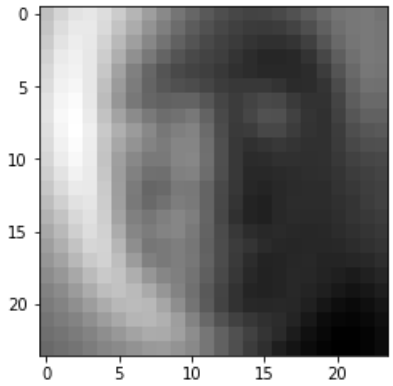
```
K_ = [5,10,50,100]
for k in K_:
    print("K = " + str(k))
    X0hat = np.dot(W[:, :k],V[:k,:])
    print("Face 1")
```

```
    print("Face 1")
    im1 = np.reshape(X0hat[0], [24,24]) #face1
    plt.imshow(im1.T , cmap="gray")
    plt.show()
    print("Face 2")
    im2 = np.reshape(X0hat[1], [24,24]) #face2
    plt.imshow(im2.T , cmap="gray")
    plt.show()
```

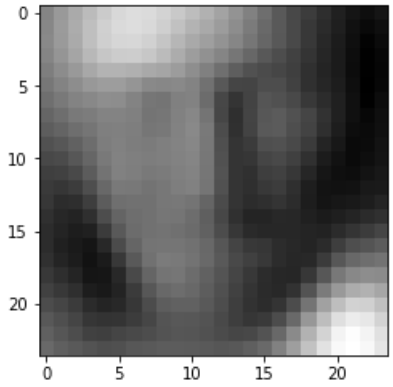
```
K = 5
Face 1
```



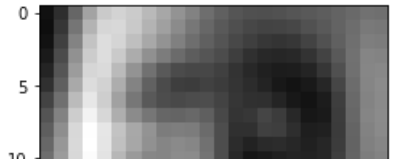

```
Face 2
```

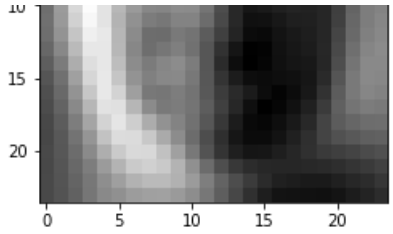



```
K = 10
Face 1
```

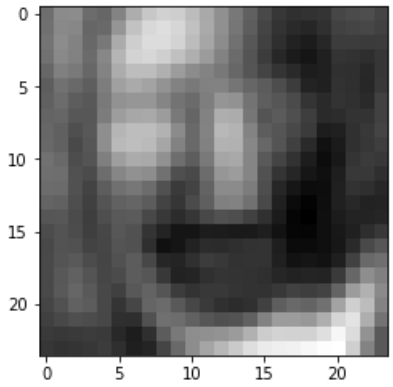



```
Face 2
```

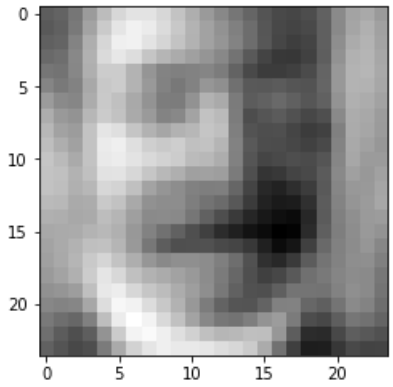




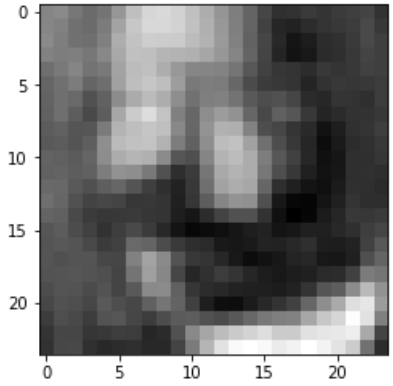
K = 50
Face 1



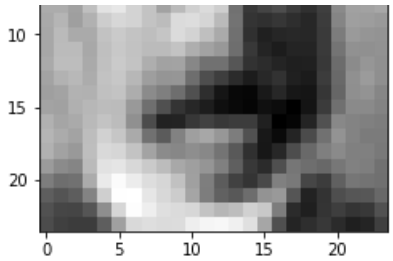
Face 2



K = 100
Face 1



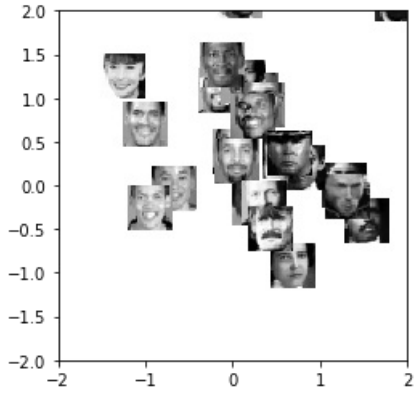
Face 2



1.6:

In [92]:

```
idx = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25] # pick some data
(randomly or otherwise); an array of integer indices
import mltools.transforms
coord,params = ml.transforms.rescale( W[:,0:2] ) # normalize scale of "W" locations
plt.figure();
for i in idx:
    # compute where to place image (scaled W values) & size
    loc = (coord[i,0],coord[i,0]+0.5, coord[i,1],coord[i,1]+0.5)
    img = np.reshape( X[i,:], (24,24) ) # reshape to square
    plt.imshow( img.T , cmap="gray", extent=loc ) # draw each image
    plt.axis( (-2,2,-2,2) ) # set axis to a reasonable scale
plt.show()
```



Problem 3:

Statement of Collaboration:

I watched this video, https://www.youtube.com/watch?v=RdT7bhm1M3E, to better understand Hierarchical Agglomerative Clustering. I also looked at the discussion python notebook named "svd". I also looked on piazza and read over the discussions over how outputs looked to others.