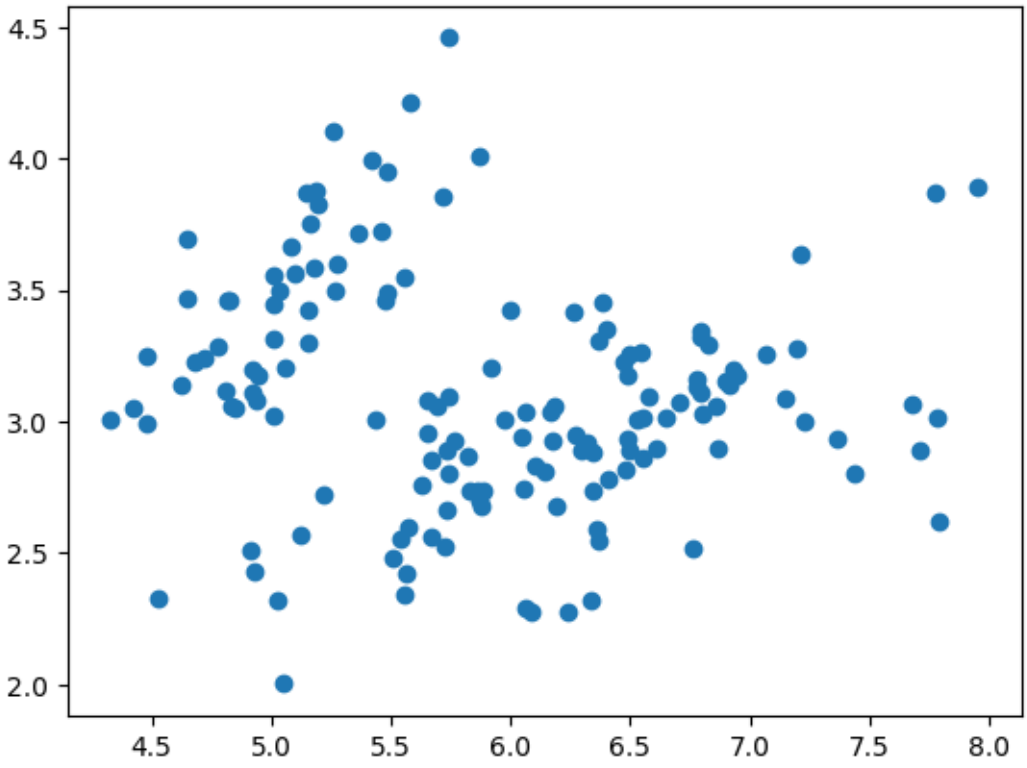
hw5

March 10, 2024

[61]:
```
import numpy as np
import matplotlib.pyplot as plt
import mltools as ml
import sys
```

0.1 Problem 1.1

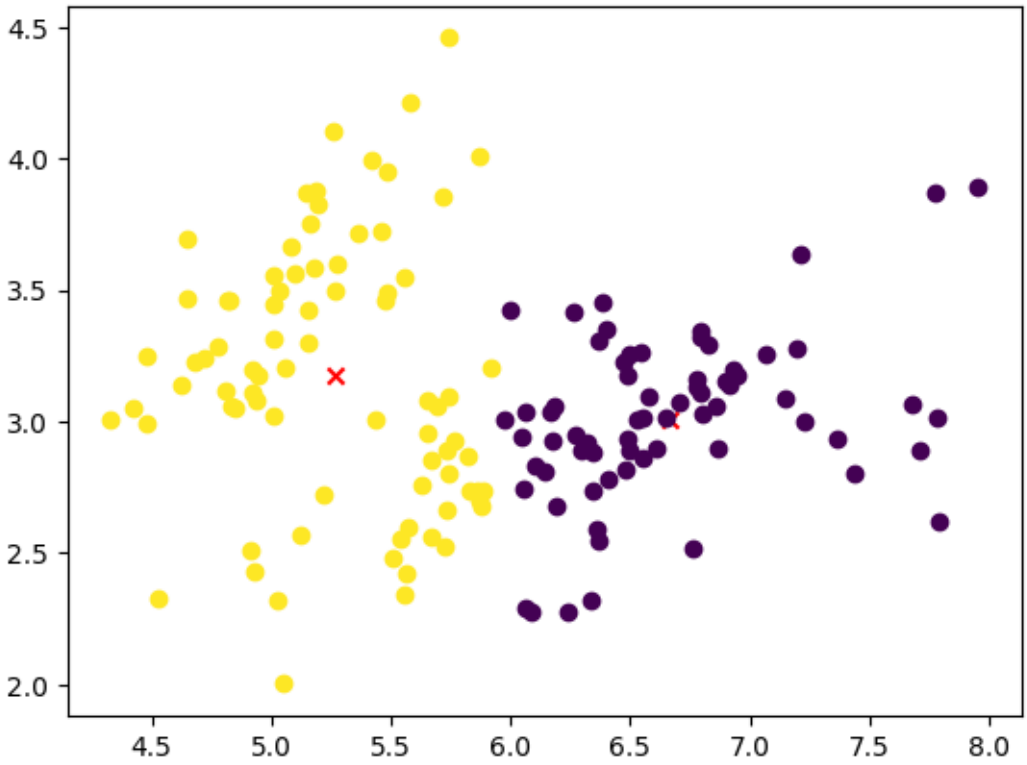
[62]:
```
iris=np.genfromtxt("data/iris.txt",delimiter=None)
X,Y=iris[:,0:2], iris[:,-1]

plt.figure()
plt.scatter(X[:, 0], X[:, 1])
plt.show()
```



The figure shows approximately 4 clusters ## Problem 1.2

[63]:
```
k=2
sumd_best = sys.maxsize
for i in range(10):
    z, c, sumd = ml.cluster.kmeans(X, k)
    if sumd < sumd_best:
        sumd_best = sumd
        z_best = z
        c_best = c
        sumdbest = sumd
ml.plotClassify2D(None, X, z_best)
plt.scatter(c[:,0],c[:,1],c='r',marker='x')
```

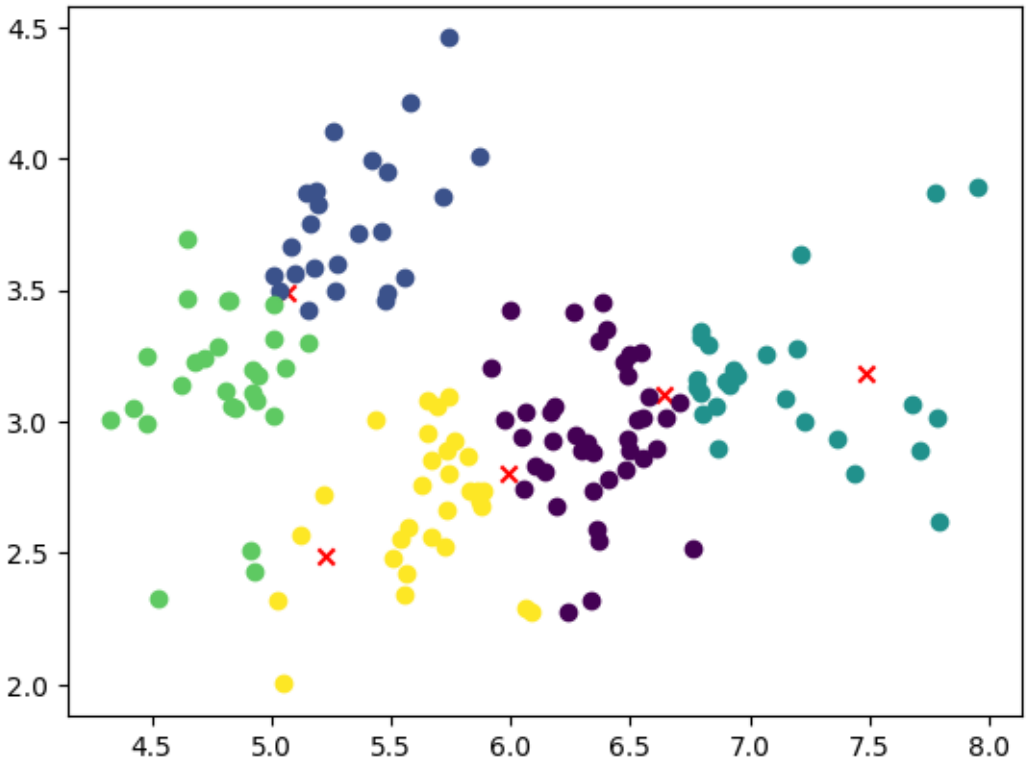
[63]:
```
<matplotlib.collections.PathCollection at 0x7f7a34894ee0>
```



[64]:
```
k=5
sumd_best = sys.maxsize
for i in range(10):
    z, c, sumd = ml.cluster.kmeans(X, k)
    if sumd < sumd_best:
```

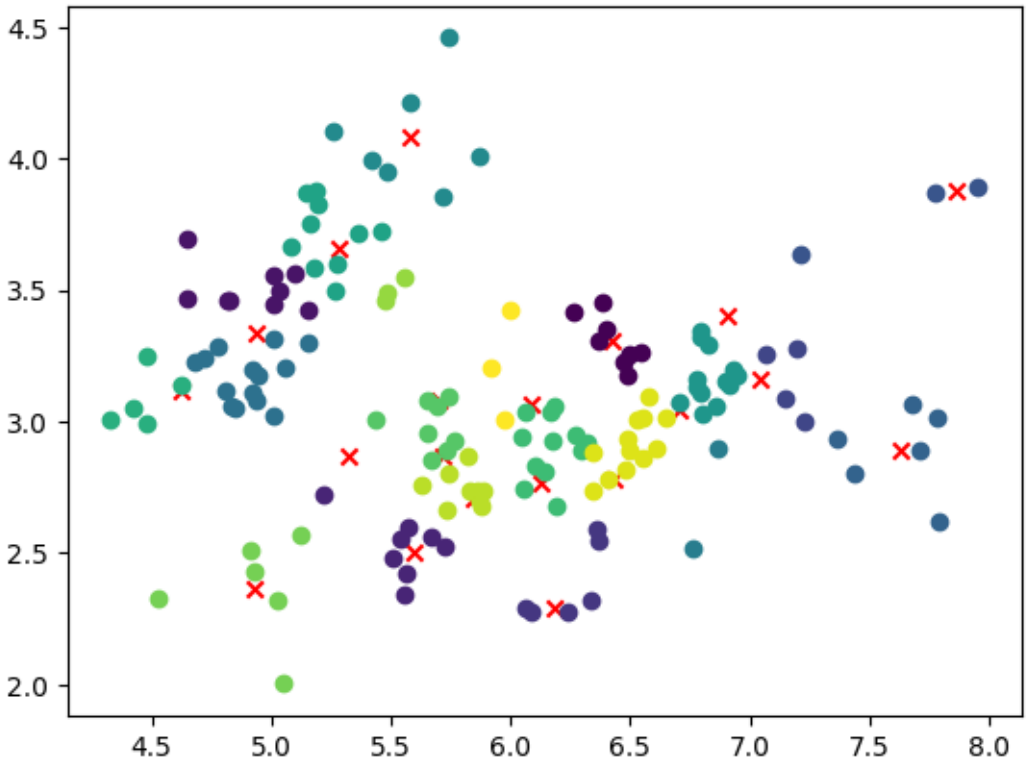
```
        sumd_best = sumd
        z_best = z
        c_best = c
        sumdbest = sumd
ml.plotClassify2D(None, X, z_best)
plt.scatter(c[:,0],c[:,1],c='r',marker='x')
```

[64]: <matplotlib.collections.PathCollection at 0x7f7a234b2b50>



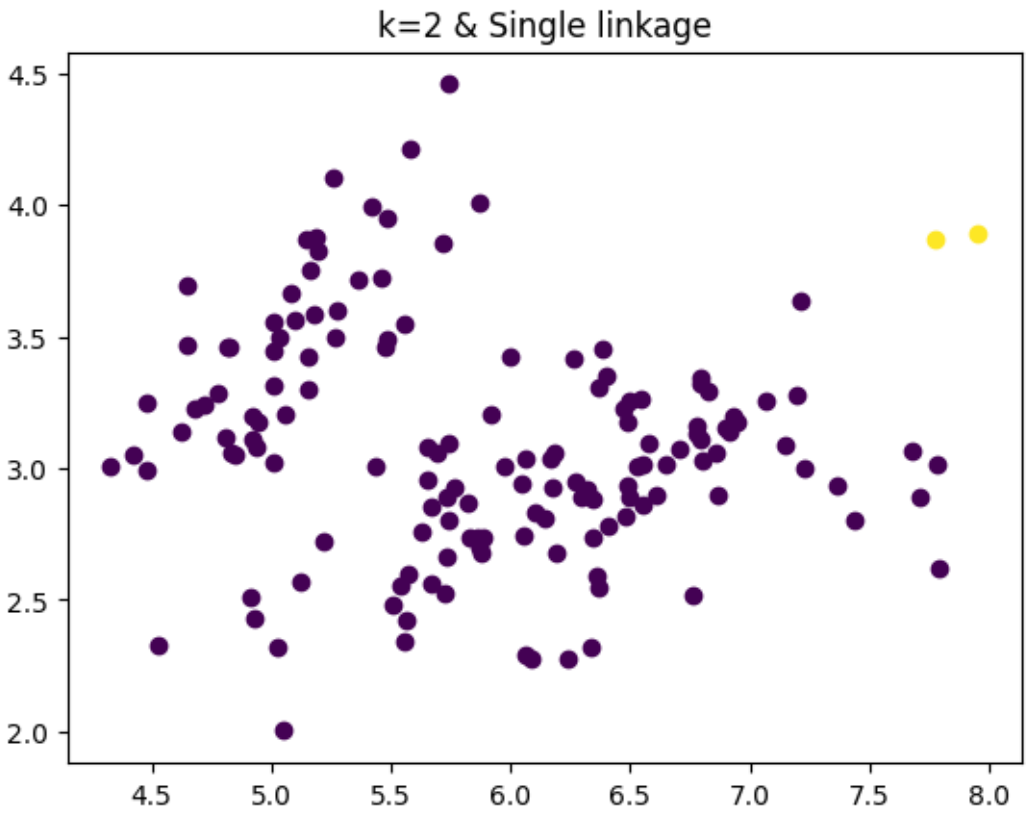
[65]:
```
k=20
sumd_best = sys.maxsize
for i in range(15):
    z, c, sumd = ml.cluster.kmeans(X, k)
    if sumd < sumd_best:
        sumd_best = sumd
        z_best = z
        c_best = c
        sumdbest = sumd
ml.plotClassify2D(None, X, z_best)
plt.scatter(c[:,0],c[:,1],c='r',marker='x')
```

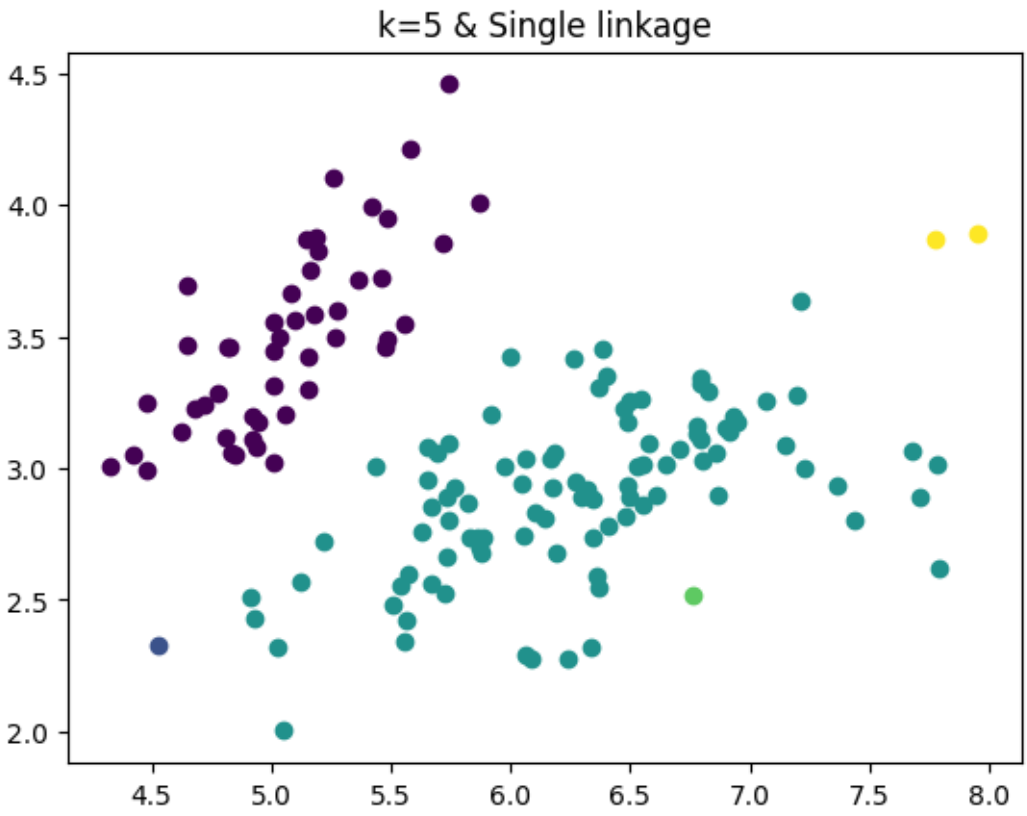
[65]: <matplotlib.collections.PathCollection at 0x7f7a33f9c2b0>

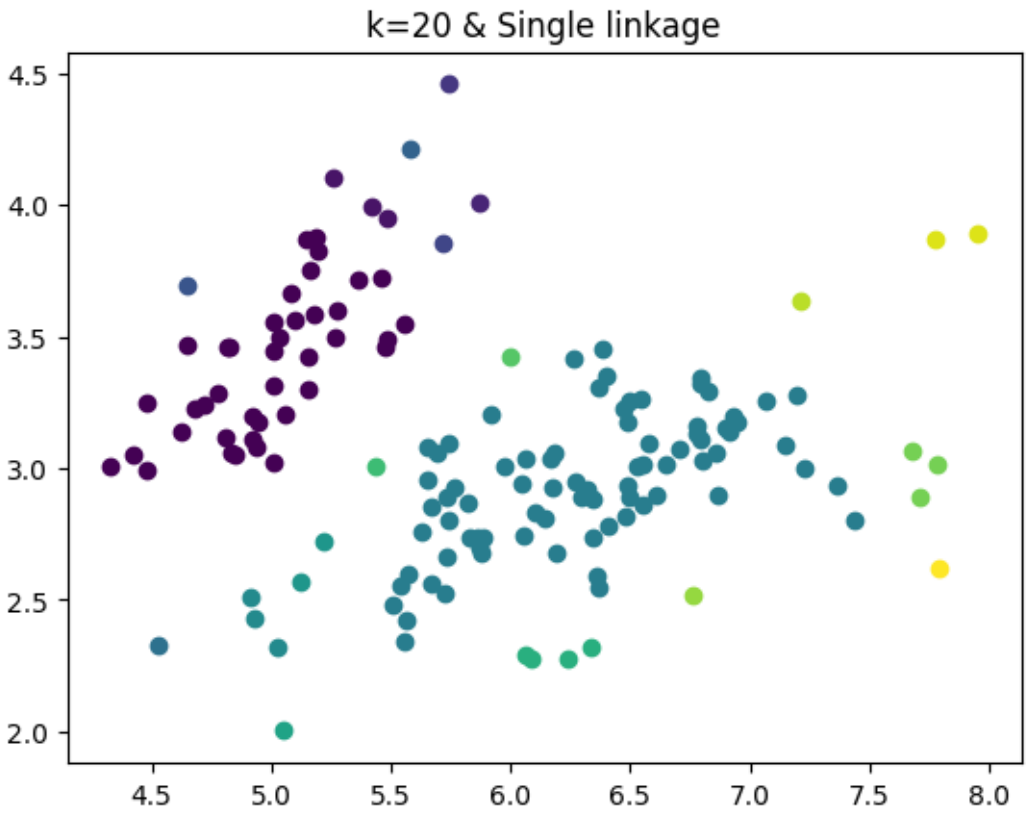


0.2 Problem 1.3

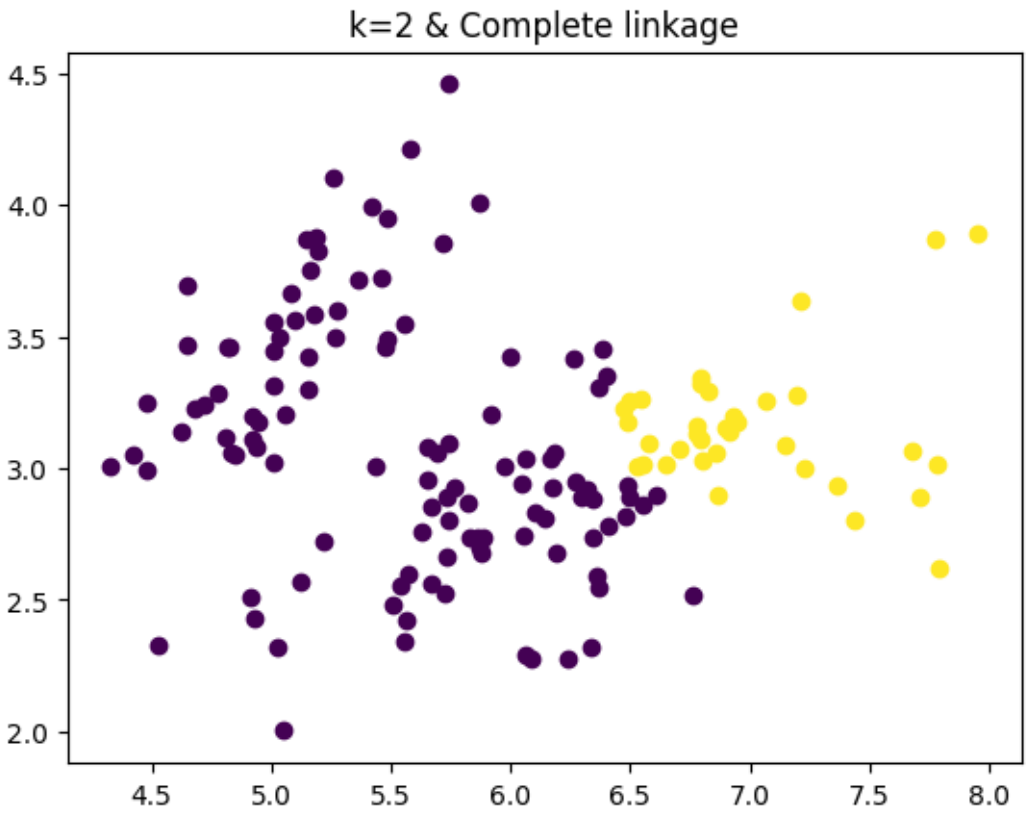
[66]:
```
k_arr = [2,5, 20]
for k in k_arr:
    z, _ = ml.cluster.agglomerative(X, k, method="min")
    ml.plotClassify2D(None, X, z)
    plt.title('k={} & Single linkage'.format(k))
    plt.show()
```

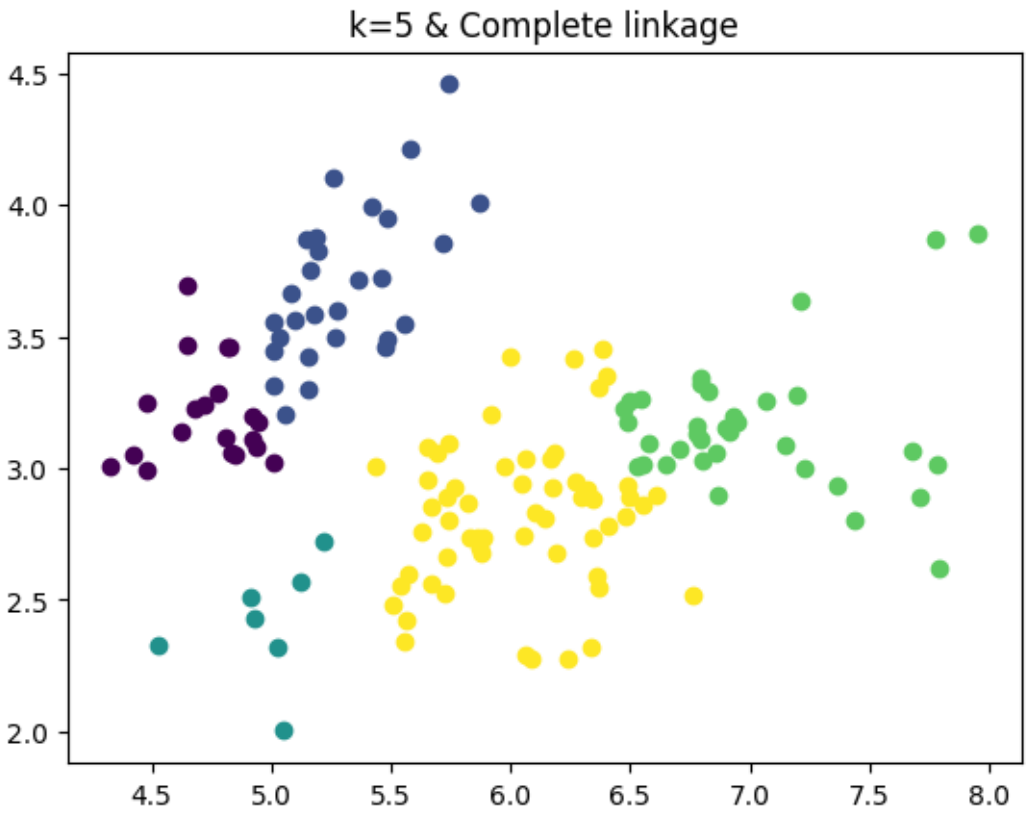

k=2 & Single linkage
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

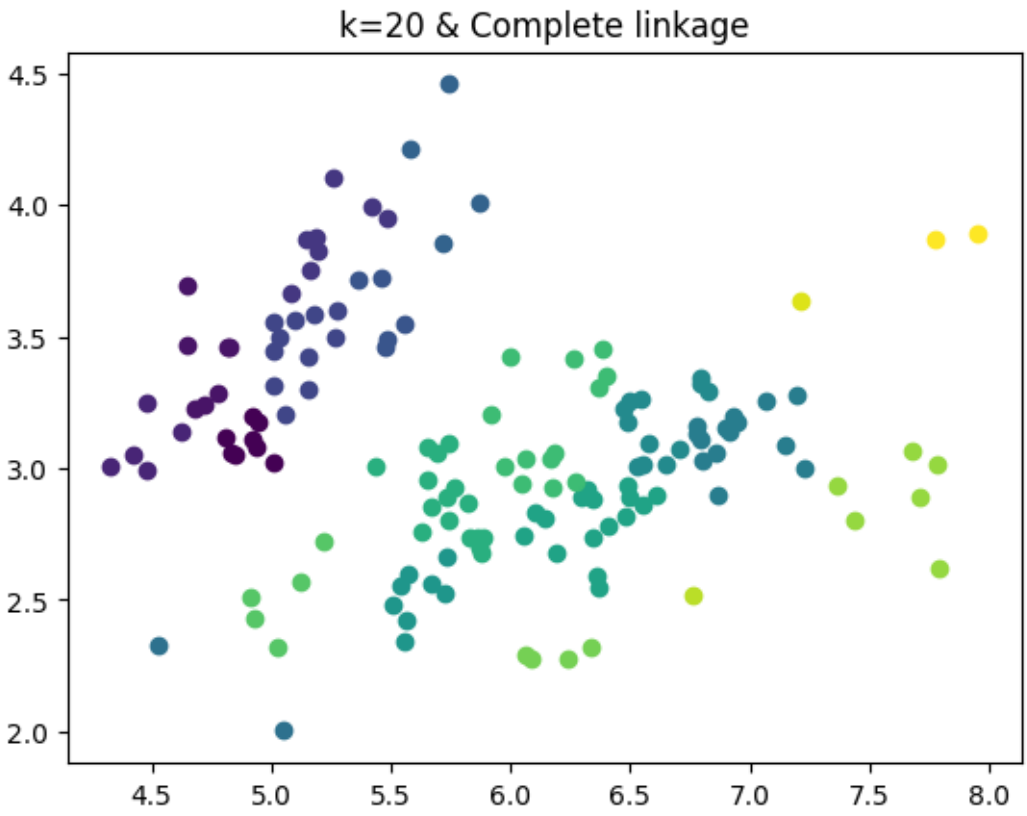

k=5 & Single linkage
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0



[67]:
```
for k in k_arr:
    z, _ = ml.cluster.agglomerative(X, k, method="max")
    ml.plotClassify2D(None, X, z)
    plt.title('k={} & Complete linkage'.format(k))
    plt.show()
```


k=2 & Complete linkage
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


k=5 & Complete linkage
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0



0.3 Problem 1.4

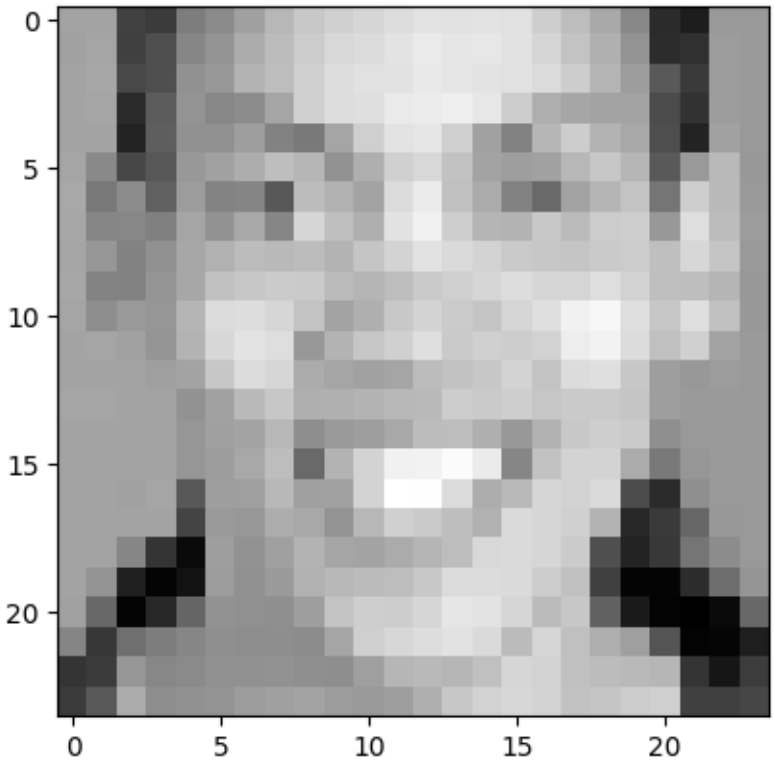
Similarities:
1. They aim to partition the data into clusters

Differences: 1. In k-means, the number of clusters needs to be specified in advance, but agglomerative doesn't. 2. k-means is sensitive to the initialization, but agglomerative doesn't.

0.4 Problem 2.1

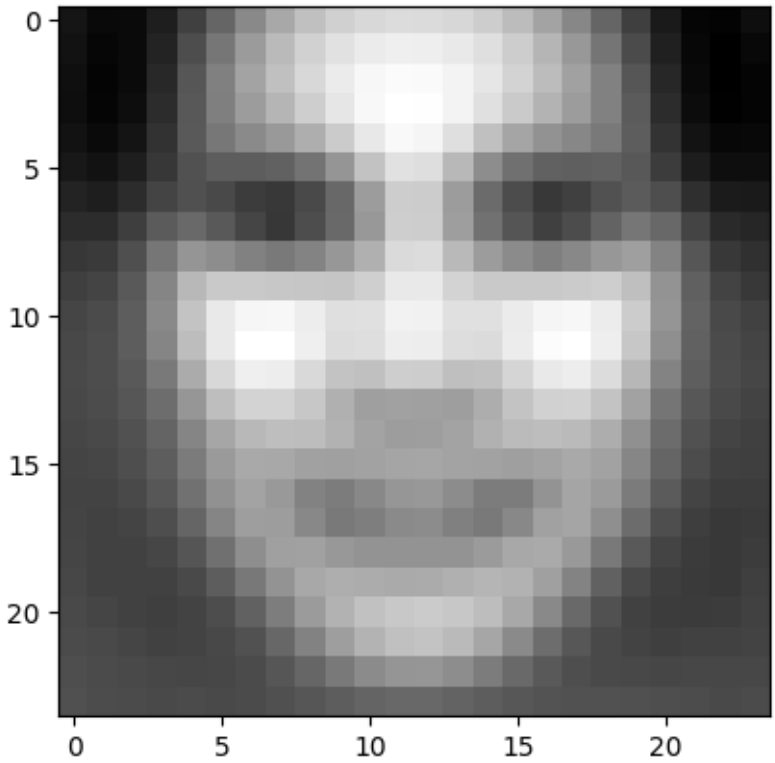
[68]:
```
X = np.genfromtxt("data/faces.txt", delimiter=None) # load face dataset
plt.figure()
# pick a data point i for display
img = np.reshape(X[i,:],(24,24)) # convert vectorized data to 24x24 image␣
 ↪patches
plt.imshow( img.T , cmap="gray") # display image patch; you may have to squint
```

[68]:
```
<matplotlib.image.AxesImage at 0x7f7a231767c0>
```



[69]:
```
mu=X.mean(axis=0,keepdims=True)
X0=X-mu
plt.figure()
mean_face=np.reshape(mu,(24,24))
plt.imshow(mean_face.T,cmap="gray")
```

[69]: <matplotlib.image.AxesImage at 0x7f7a33da96a0>



0.5 Problem 2.2

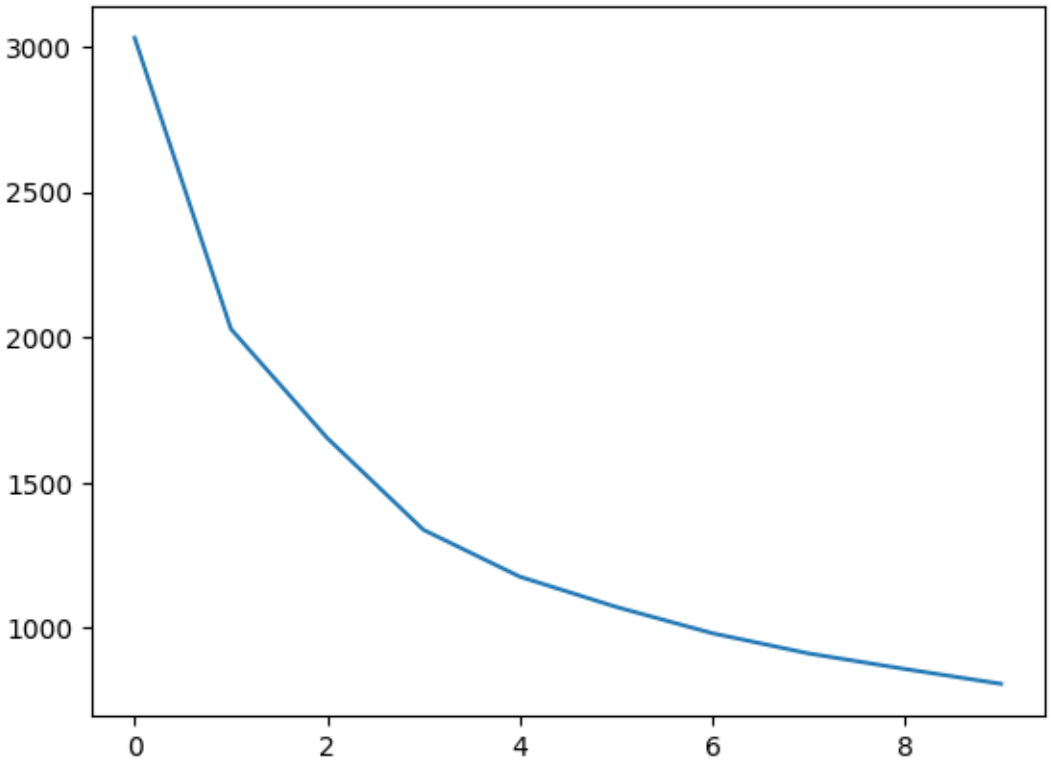
[70]:
```
import scipy.linalg
U,S,V=scipy.linalg.svd(X0,full_matrices=False)
W=U.dot(np.diag(S))
print(W.shape,V.shape)
```

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

0.6 Problem 2.3

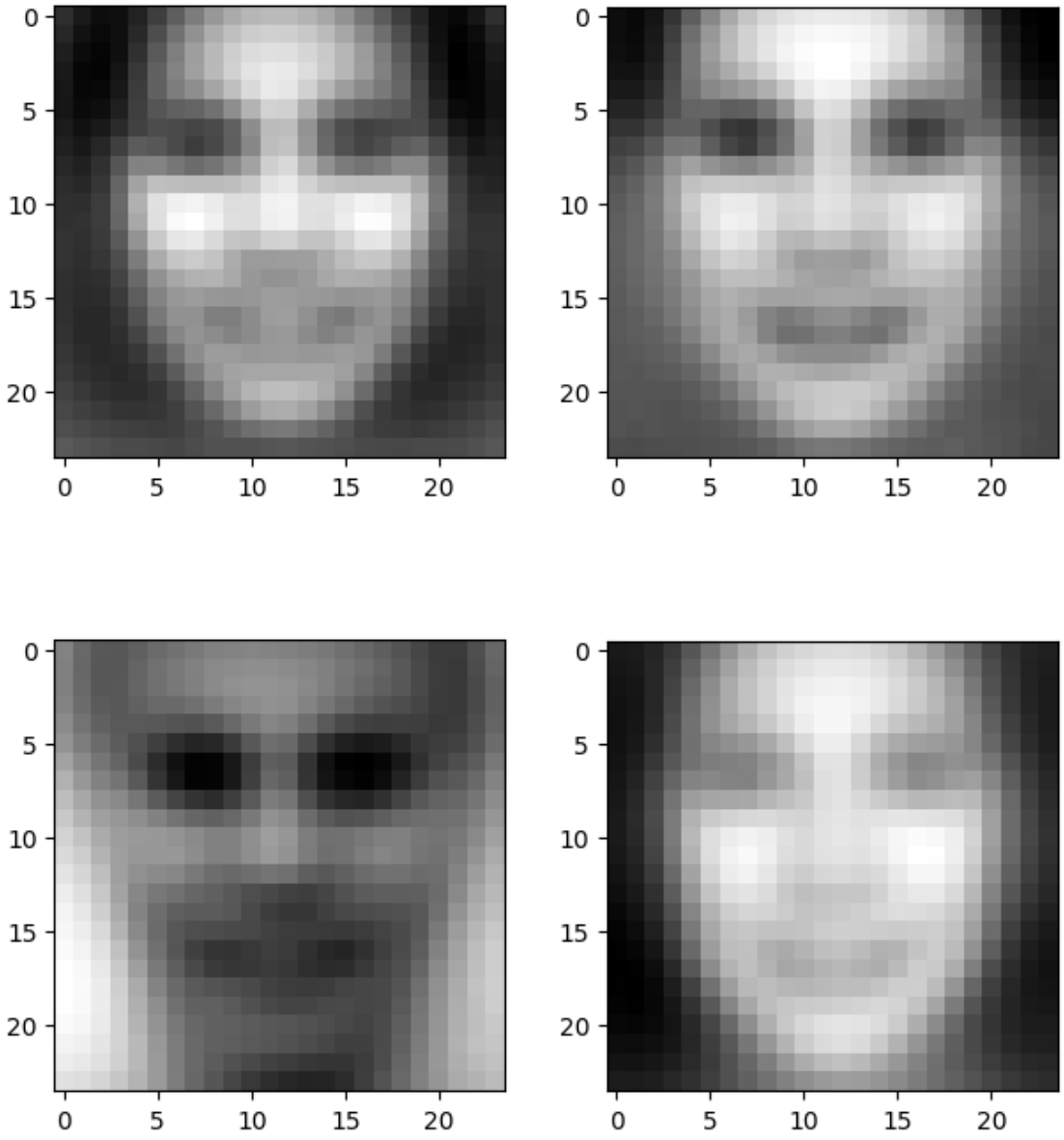
[71]:
```
mse_array=[None]*10
for i in range(10):
    X0_hat=W[:,:i].dot(V[:i,:])
    mse_array[i]=((X0-X0_hat)**2).mean()
plt.plot(mse_array)
```

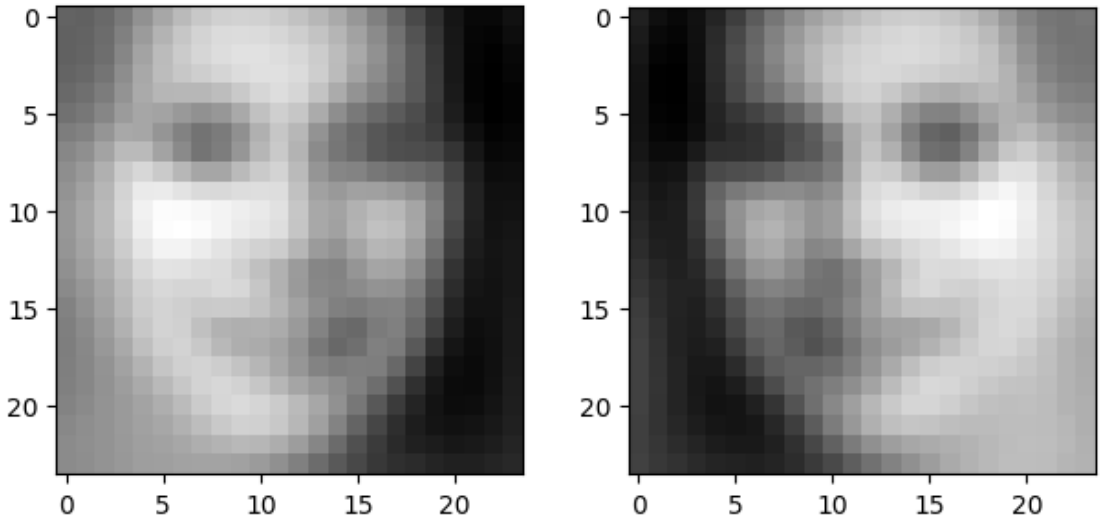
[71]:
```
[<matplotlib.lines.Line2D at 0x7f7a34a1df40>]
```



0.7 Problem 2.4

[72]:
```
for i in range(3):
    alpha = 2 * np.mean(np.abs(W[:, i]))
    vector1 = mu + alpha * V[i, :]
    vector2 = mu - alpha * V[i, :]

    img1 = np.reshape(vector1, (24, 24))
    img2 = np.reshape(vector2, (24, 24))

    fig, axs = plt.subplots(1, 2)
    plt.tight_layout()
    axs[0].imshow(img1.T, cmap='gray')
    axs[1].imshow(img2.T, cmap='gray')
    plt.show()
```

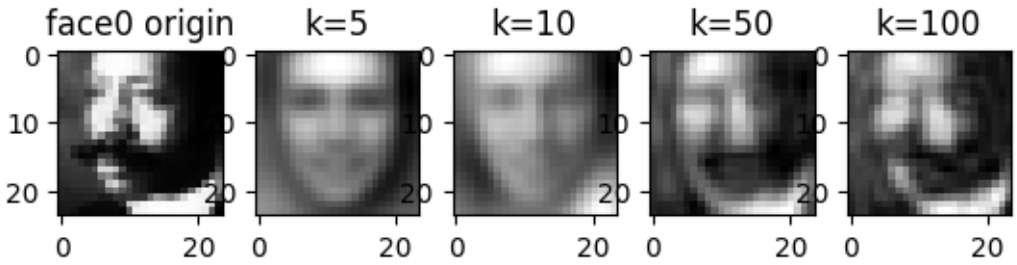

0
5
10
15
20
0
5
10
15
20
0
5
10
15
20
0
5
10
15
20
0
5
10
15
20
0
5
10
15
20
0
5
10
15
20
0
5
10
15
20
0
5
10
15
20



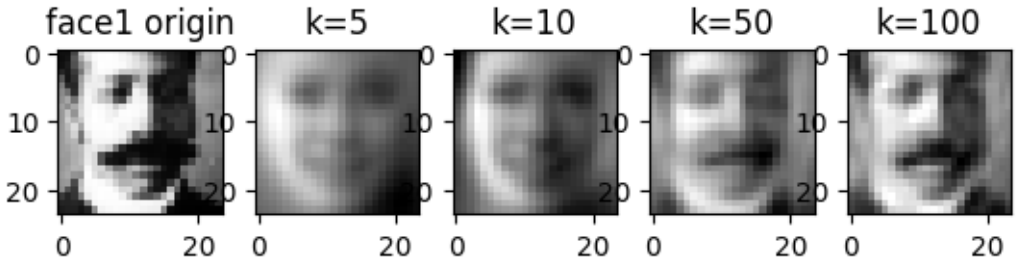
0.8 Problem 2.5

[73]:
```
for i in range(2):
    origin_img = np.reshape(X[i,:], (24, 24))
    plt.figure()
    fig,axs=plt.subplots(1,5)
    axs[0].imshow(origin_img.T,cmap="gray")
    axs[0].set_title(f"face{i} origin")
    index=1
    for k in [5,10,50,100]:
        img=mu + W[i, :k].dot(V[:k, :])
        img=np.reshape(img,(24,24))
        axs[index].imshow(img.T,cmap="gray")
        axs[index].set_title(f'k={k}')
        index +=1
plt.show()
```

```
<Figure size 640x480 with 0 Axes>
```



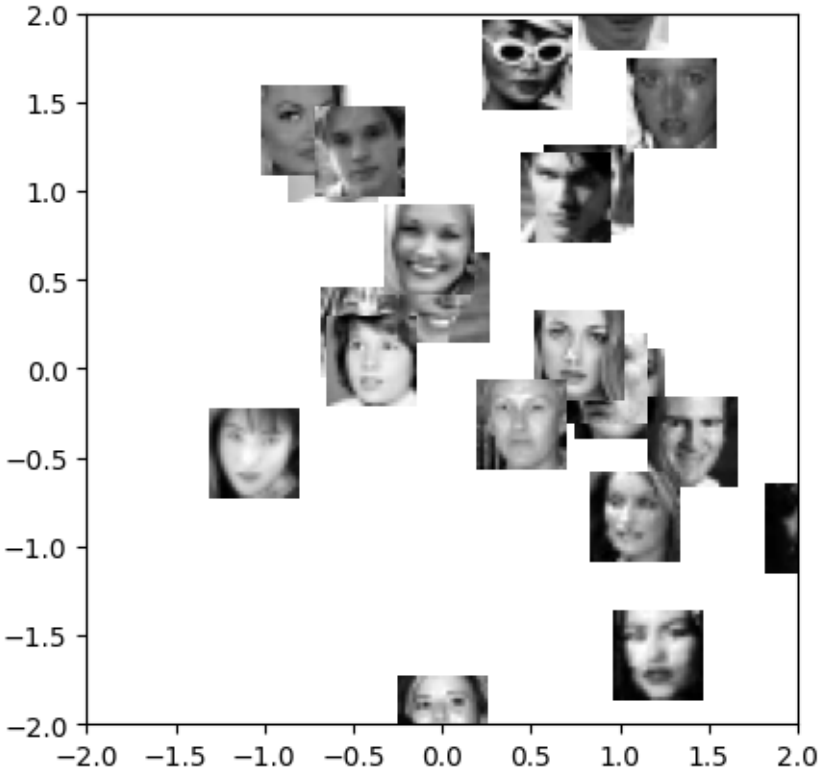
```
<Figure size 640x480 with 0 Axes>
```



0.9 Problem 2.6

[74]:
```
import random
import mltools.transforms

idx = random.sample(range(W.shape[0]), 25)
print(idx)

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
```

```
[1472, 4894, 2326, 4209, 1263, 1621, 2884, 1811, 996, 3309, 3708, 2007, 4332,
702, 1909, 3788, 213, 2958, 653, 225, 4045, 362, 1366, 1431, 143]
```



0.10 Statement of Collaboration

I do it by myself