Lab 3 - Wyatt Madden & Dan Crowley

January 31, 2020

1 3.1

```
In [3]: import scipy.io as scipy
        import numpy as np
        import seaborn as sns
        import matplotlib.pyplot as plt
```

```
In [65]: # PRINcipal COMPonent calculator
         #   Calculates the principal components of a collection of points.
         # Input:
         #   X - D-by-N data matrix of N points in D dimensions.
         # Output:
         #   W - A D-by-M matrix containing the M principal components of the data.
         #   Z - A M-by-N matrix containing the latent variables of the data.
         #   mu - A D-by-1 vector containing the mean of the data.
         #   lambda - A vector containing the eigenvalues associated with the above principal co

         def pca(X, M):
             mu = np.sum(X, axis = 1)/X.shape[1]
             X_centered = np.transpose(np.subtract(mu, np.transpose(X)))
             S = np.matmul(np.transpose(X_centered), X_centered) / len(X)
             eig_vals_and_vecs = np.linalg.eigh(S)
             eig_vals = eig_vals_and_vecs[0]
             eig_vecs = eig_vals_and_vecs[1]
             top_M_eigs_inds = np.argpartition(eig_vals, -M)[-M:]
             lambdas = eig_vals[top_M_eigs_inds]
             W = eig_vecs[top_M_eigs_inds]
             Z = np.matmul(W, np.subtract(mu, np.transpose(X))) / M

             d = dict()
             d['W'] = W
             d['Z'] = Z
             d['mu'] = mu
             d['lambda'] = lambdas

             return d
```

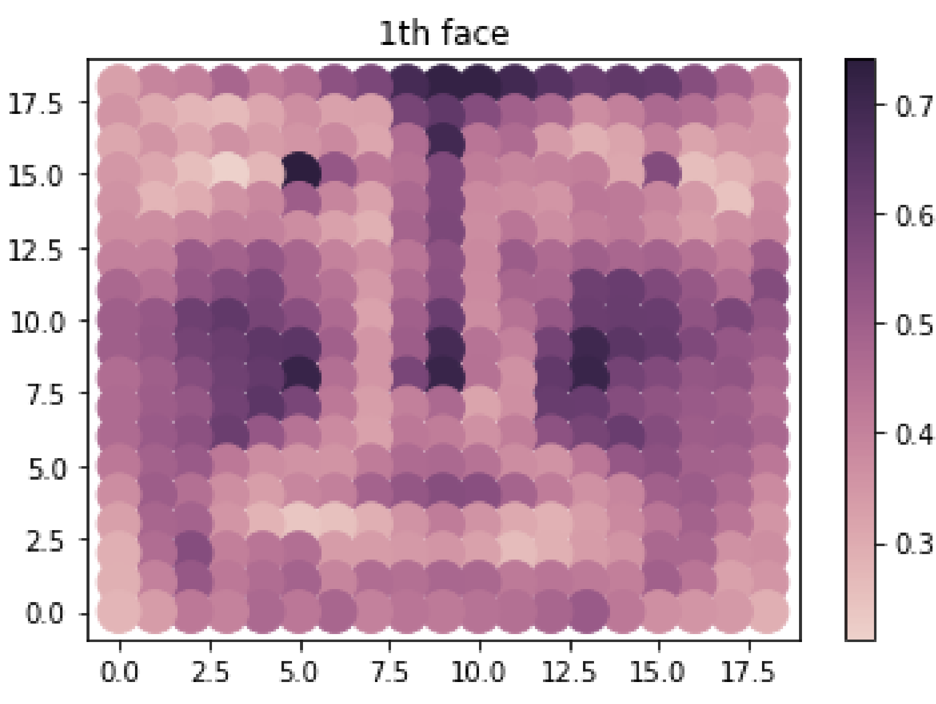
2 3.2

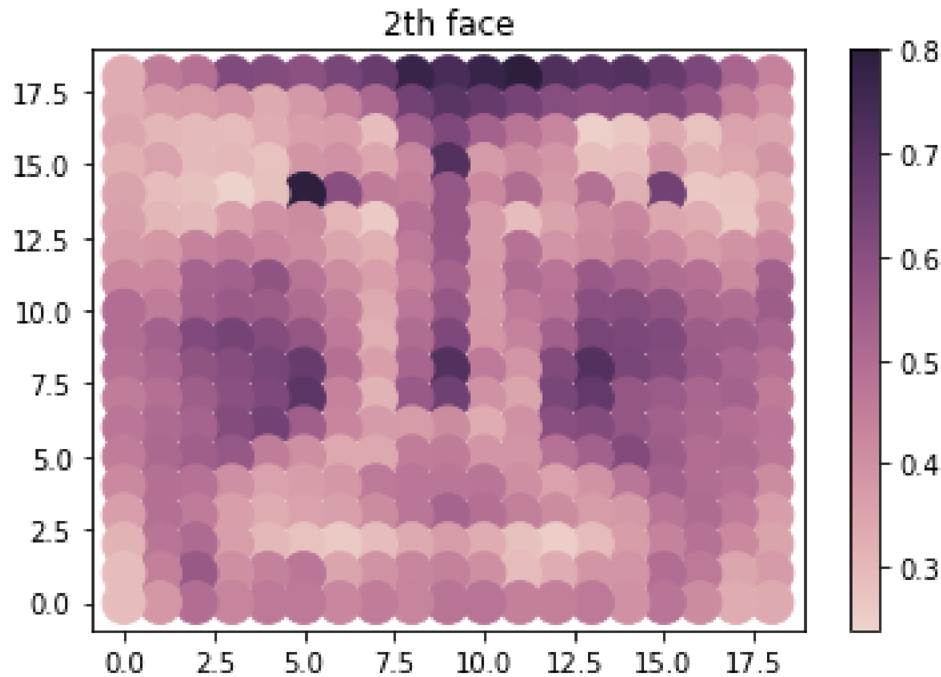
```
In [63]: cbcl = scipy.loadmat('/Users/wyattmadden/Documents/school/' +
                              'MSU/2020/spring/m508/lab_info/lab_3/cbcl.mat',
                             squeeze_me = True)

         X = cbcl['X']
         x_axis_points = np.repeat(list(range(X_shaped.shape[0] - 1, -1, -1)),
                                   X_shaped.shape[0])
         y_axis_points = np.tile(list(range(X_shaped.shape[0] - 1, -1, -1)),
                                 X_shaped.shape[0])

         for i in range(0, 2):
             one_face = X[:, i]
             X_shaped = np.reshape(X, (int(np.sqrt(X.shape[0])),
                                      int(np.sqrt(X.shape[0])),
                                      X.shape[1]))

             cmap = sns.cubehelix_palette(as_cmap=True)
             f, ax = plt.subplots()
             points = ax.scatter(x_axis_points,
                                 y_axis_points,
                                 c = one_face,
                                 s = 250,
                                 cmap = cmap)
             f.colorbar(points)
             ax.set_title(str(i + 1) + "th face")
```





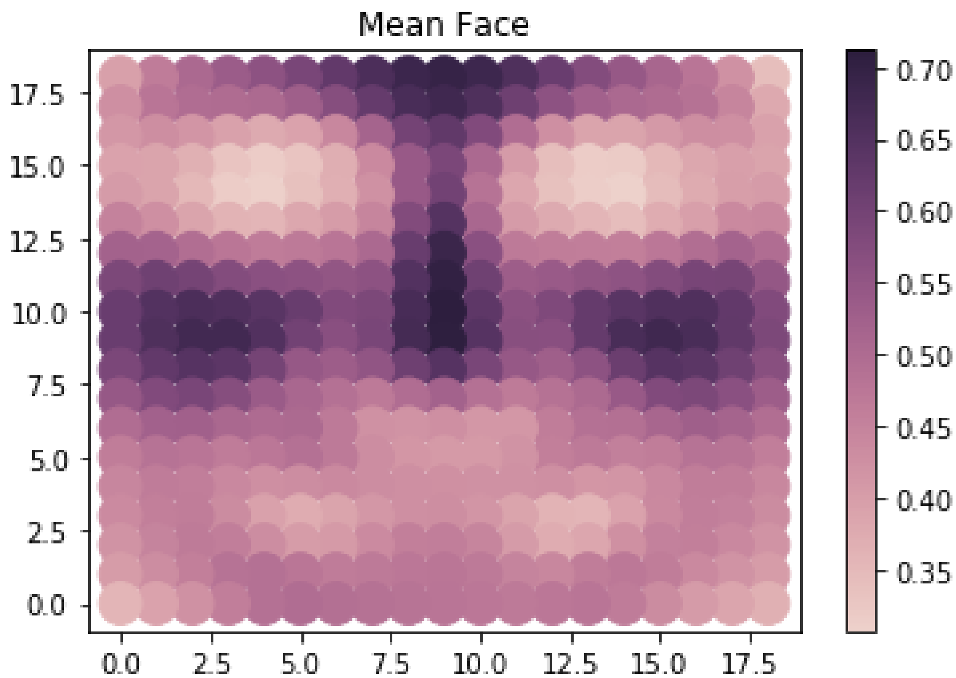
3 3.3

In [67]:
```
five_face = pca(X, 5)

mean_face = five_face['mu']


cmap = sns.cubehelix_palette(as_cmap=True)
f, ax = plt.subplots()
points = ax.scatter(x_axis_points,
                    y_axis_points,
                    c = mean_face,
                    s = 250,
                    cmap = cmap)
f.colorbar(points)
ax.set_title("Mean Face")
```

Out[67]: Text(0.5,1,'Mean Face')



Mean Face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35