Lab 3 - Wyatt Madden & Dan Crowley

February 2, 2020

1 3.1

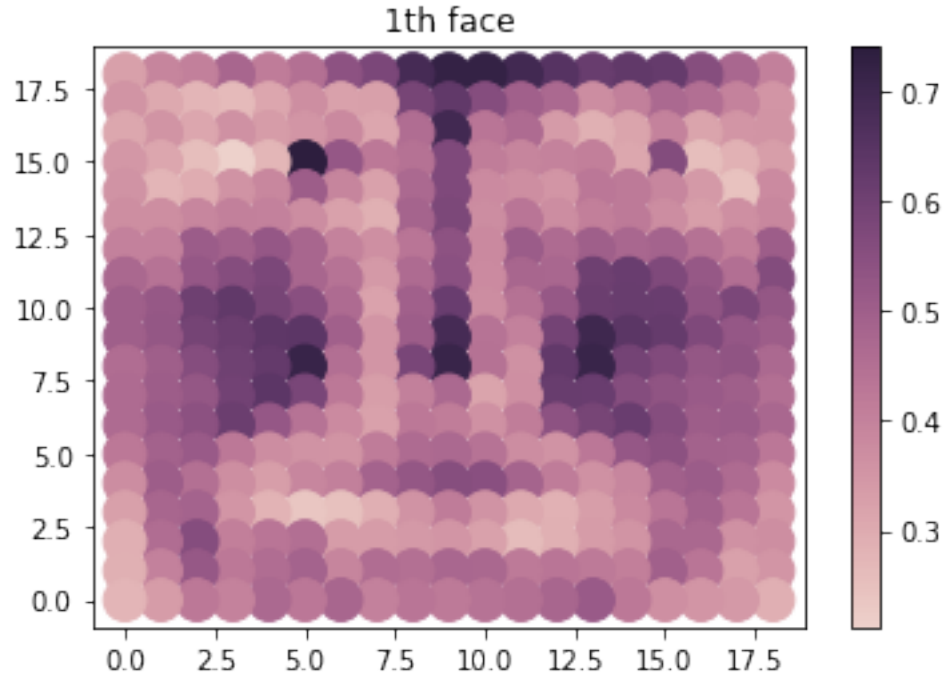
In [235]:
```
import scipy.io as scipy
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

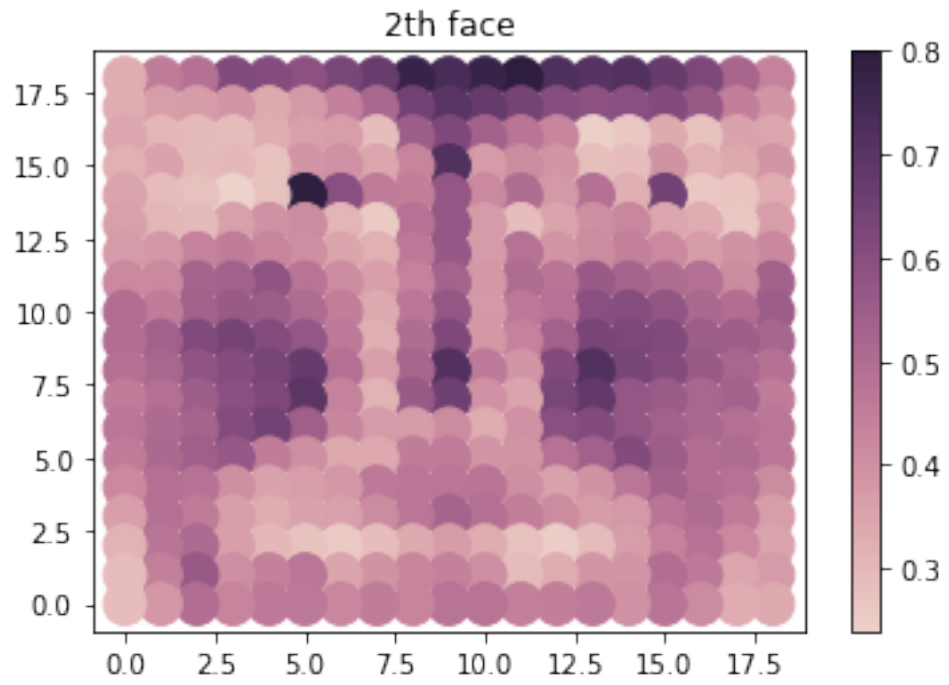
In [329]:
```
# PRINcipal COMPonent calculator
#   Calculates the principal components of a collection of points.
# Input:
#   X - D-by-N data matrix of N points in D dimensions.
# Output:
#   W - A D-by-M matrix containing the M principal components of the data.
#   Z - A M-by-N matrix containing the 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 c

def pca(X, M):
    mu = X.mean(axis = 1)
    X_centered = np.transpose(X) - mu
    S = np.cov(np.transpose(X_centered))
    eig_vals, eig_vecs = np.linalg.eigh(S)
    top_M_eigs_inds = np.argpartition(eig_vals, -M)[-M:][::-1]
    lambdas = eig_vals[top_M_eigs_inds]
    W = eig_vecs[:, top_M_eigs_inds]
    Z = np.matmul(np.transpose(W), np.transpose(X_centered)) / M

    return W, Z, mu, lambdas
```

2 3.2

In [330]:
```
cbcl = scipy.loadmat('/Users/wyattmadden/Documents/school/' +
                     'MSU/2020/spring/m508/lab_info/lab_3/cbcl.mat',
                     squeeze_me = True)

X = cbcl['X']
```

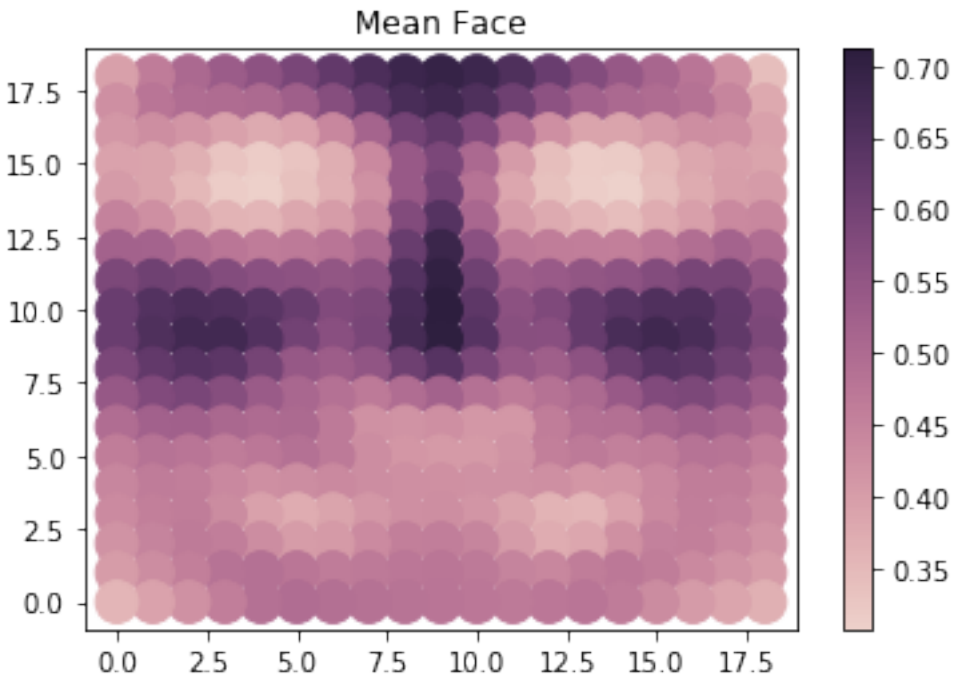
```
x_axis_points = np.repeat(list(range(X_shaped.shape[0] - 1, -1, -1)),
                          X_shaped.shape[0])
y_axis_points = np.tile(list(range(X_shaped.shape[0] - 1, -1, -1)),
                        X_shaped.shape[0])

def plot_19_grid(colour, title):
    cmap = sns.cubehelix_palette(as_cmap=True)
    f, ax = plt.subplots()
    points = ax.scatter(x_axis_points,
                        y_axis_points,
                        c = colour,
                        s = 250,
                        cmap = cmap)
    f.colorbar(points)
    ax.set_title(title)

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]))

    plot_19_grid(one_face, str(i + 1) + "th face")
```



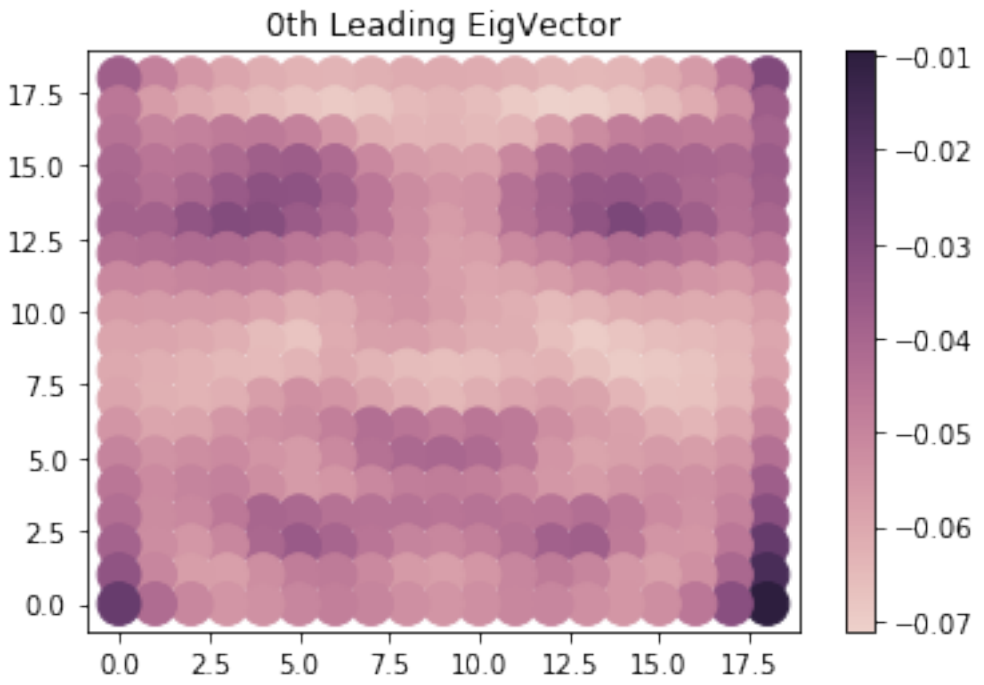


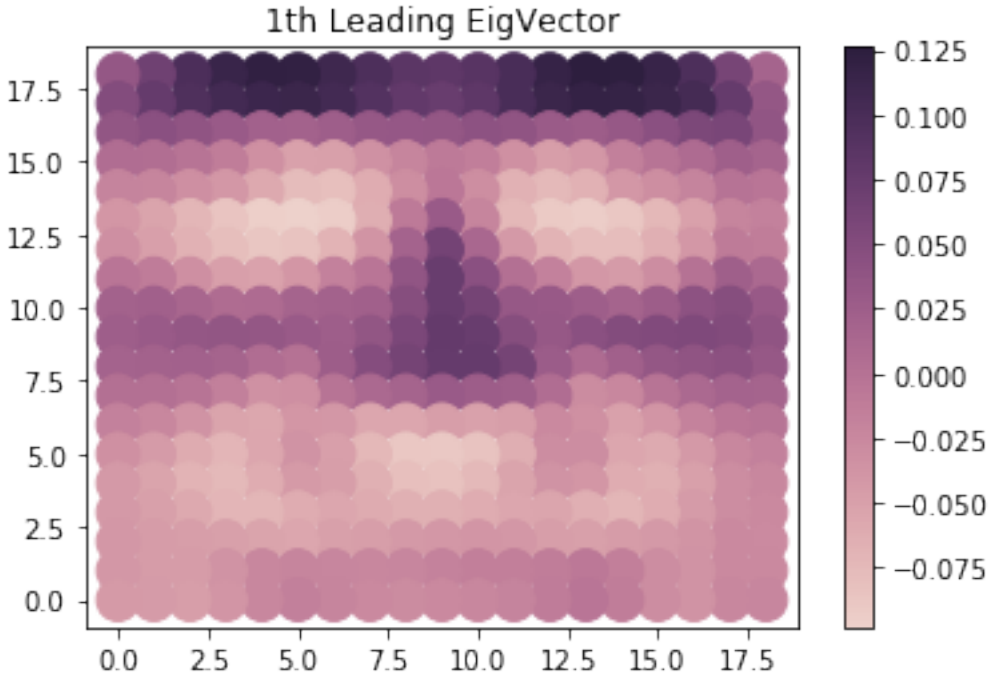
3 3.3

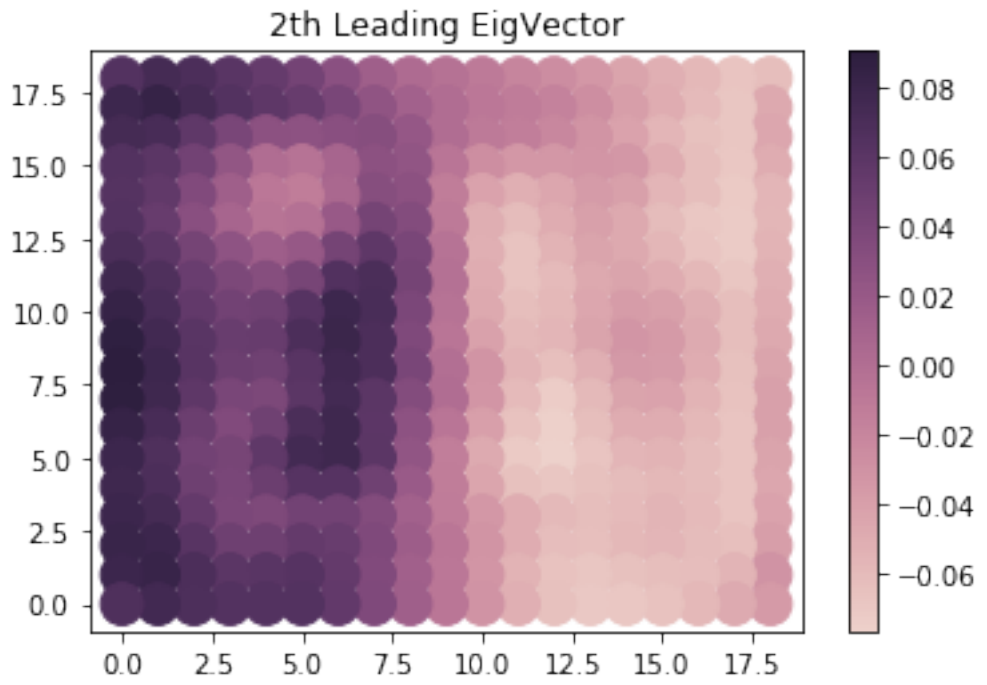
In [331]:
```
W, Z, mu, lambdas = pca(X, 5)

mean_face = mu

plot_19_grid(mean_face, "Mean Face")
```

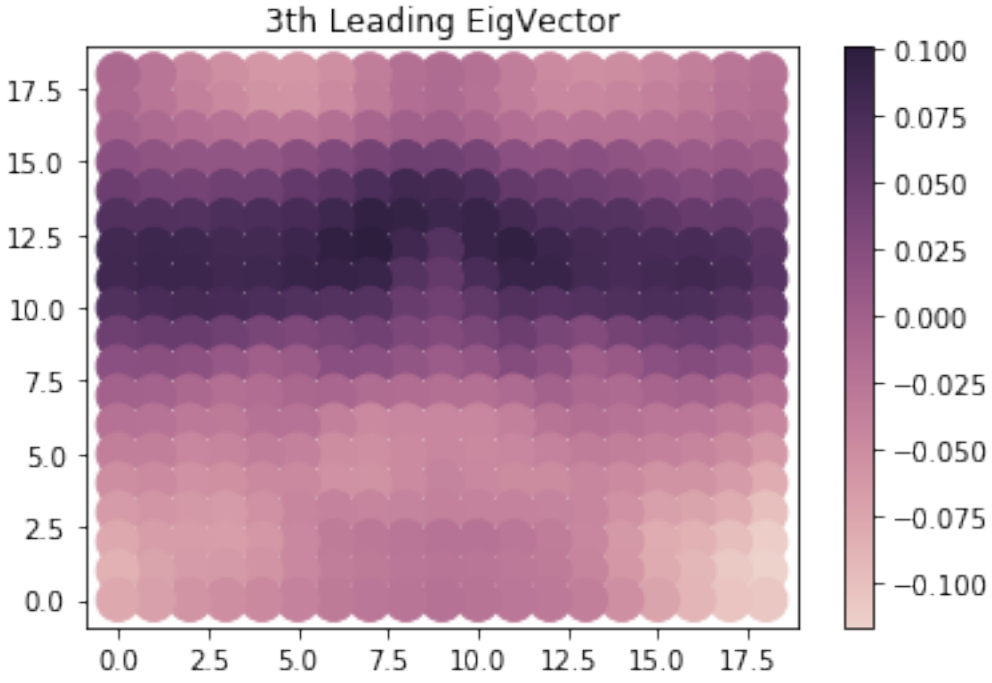


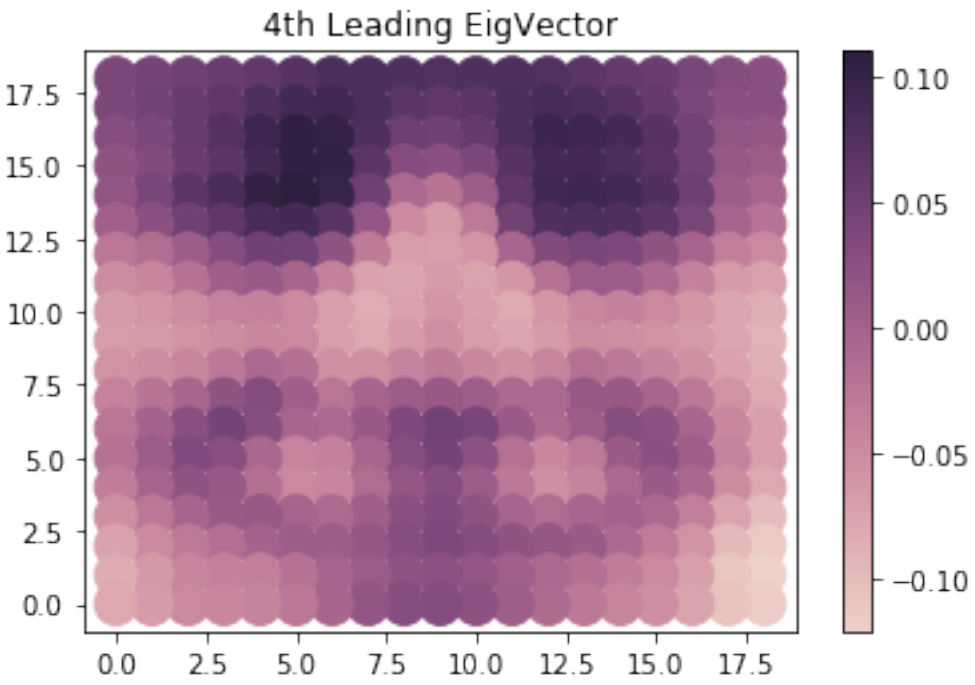
In [339]:
```
lead_5_eig_vecs = W
lead_5_eig_vals = lambdas

iter = 0
for i in np.transpose(lead_5_eig_vecs):
    plot_19_grid(i, str((iter)) + "th Leading EigVector")
    iter += 1
```


0th Leading EigVector
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
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−0.07


1th Leading EigVector
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
0.125
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075

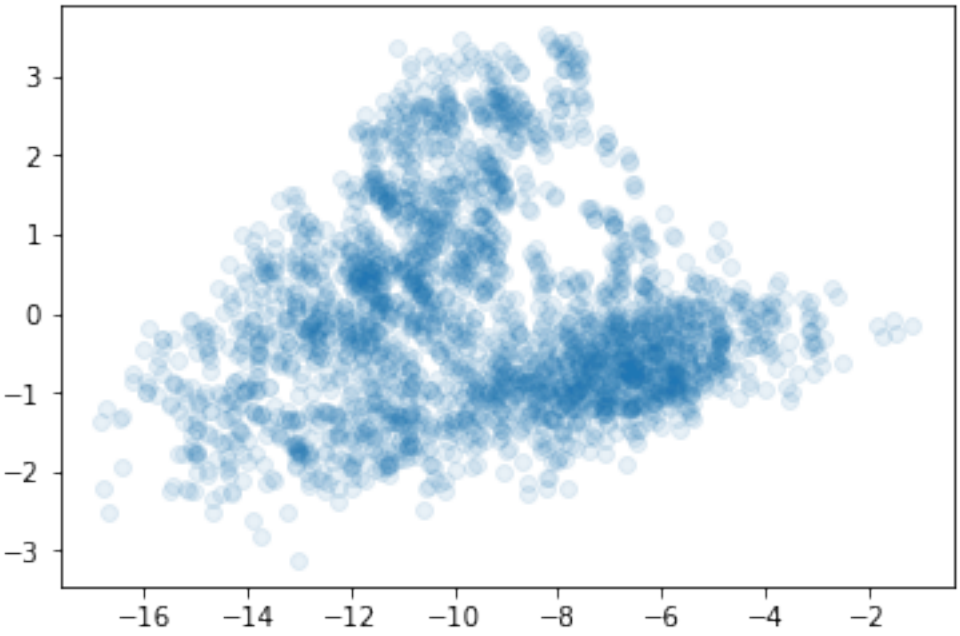

2th Leading EigVector
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06


3th Leading EigVector
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
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100



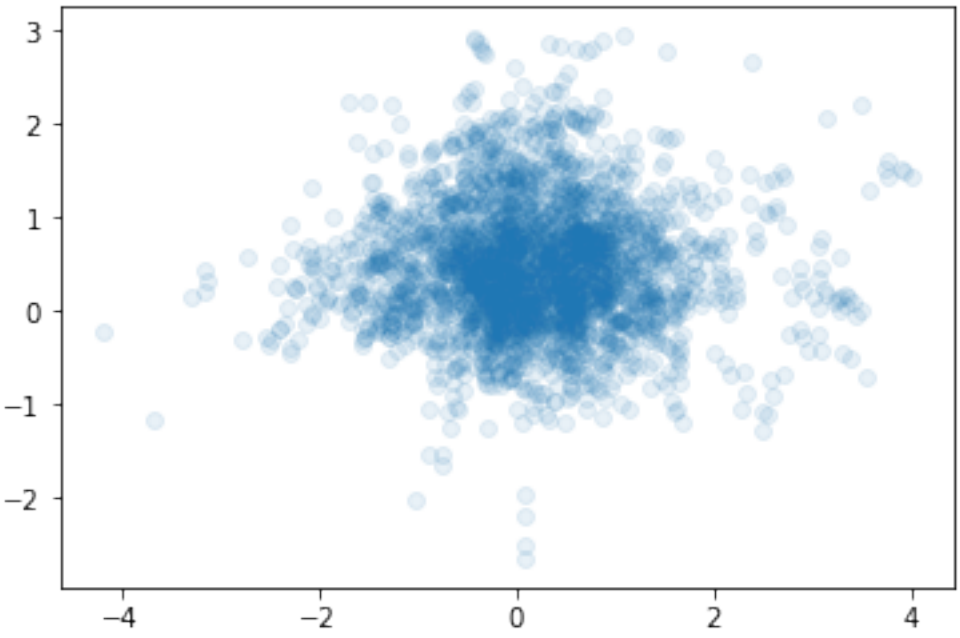
4 3.4

```
In [359]: def plot_two_prin_comps(eig_vecs, X):
              two_dim_red = np.matmul(eig_vecs, X)
              f, ax = plt.subplots()
              points = ax.scatter(two_dim_red[0],
                                  two_dim_red[1],
                                  alpha = 0.1)
```



5 3.5

In [365]:
```
eig_vecs_2_and_3 = np.transpose(lead_5_eig_vecs)[2:4]

plot_two_prin_comps(eig_vecs_2_and_3, X)
```



6 3.6