Lab 3 - Wyatt Madden & Dan Crowley

February 5, 2020

1 3.1

```
In [1]: import scipy.io as scipy
        import random as random
        import numpy as np
        import seaborn as sns
        import matplotlib.pyplot as plt
        import pandas as pand
```

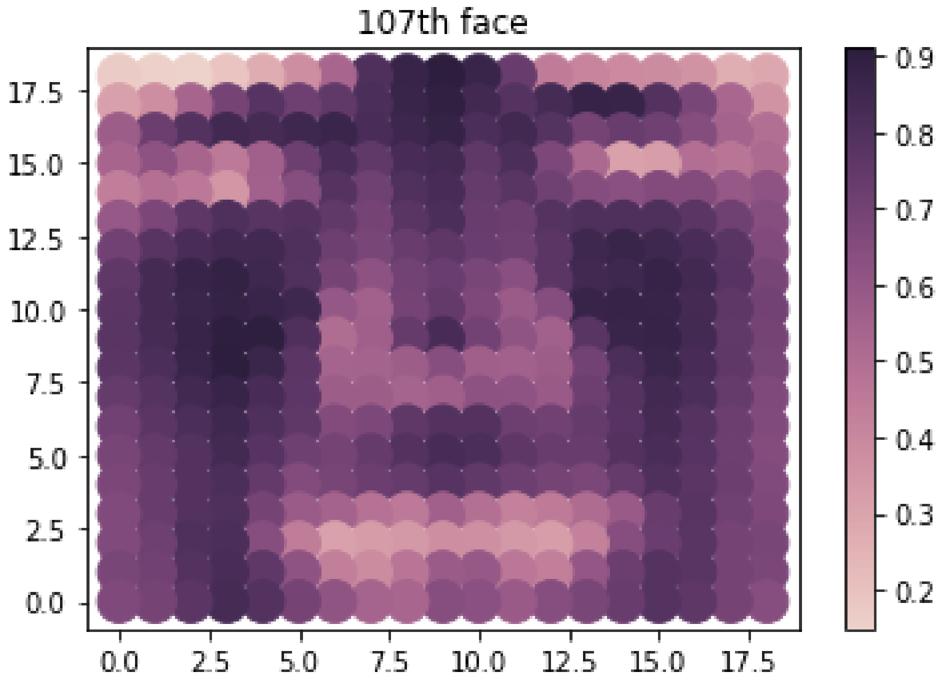
```
In [67]: # PRINcipal COMPonent calculator
         #   Calculates the principal components of a collection of points.
         # Input:
         #   X - D-by-N data matrix of N points in D dimensions.
         # Output:
         #   W - A D-by-M matrix containing the M principal components of the 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 = 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))

             return W, Z, mu, lambdas
```

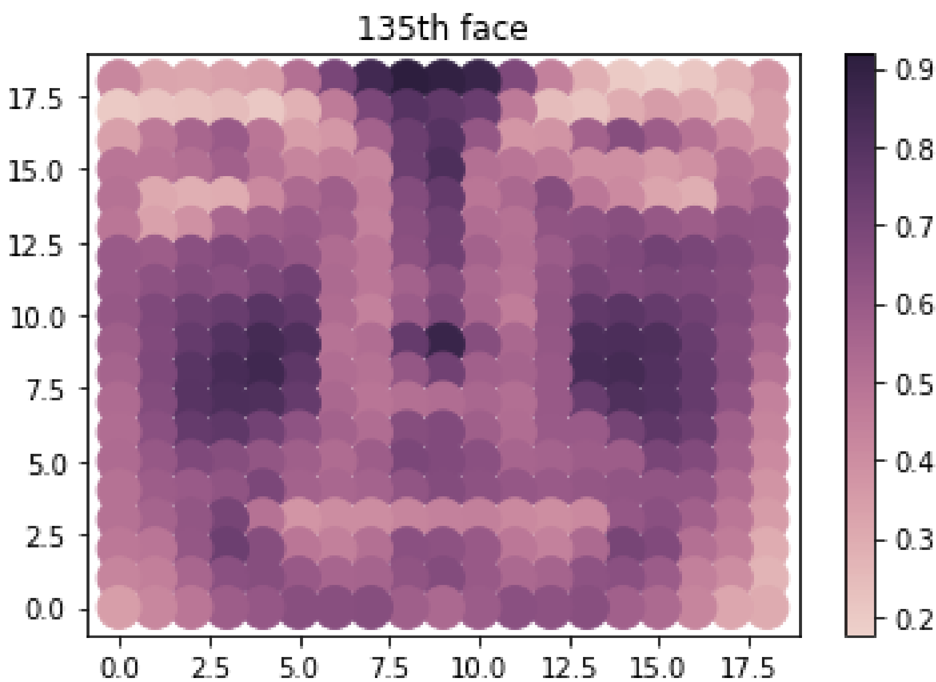
2 3.2

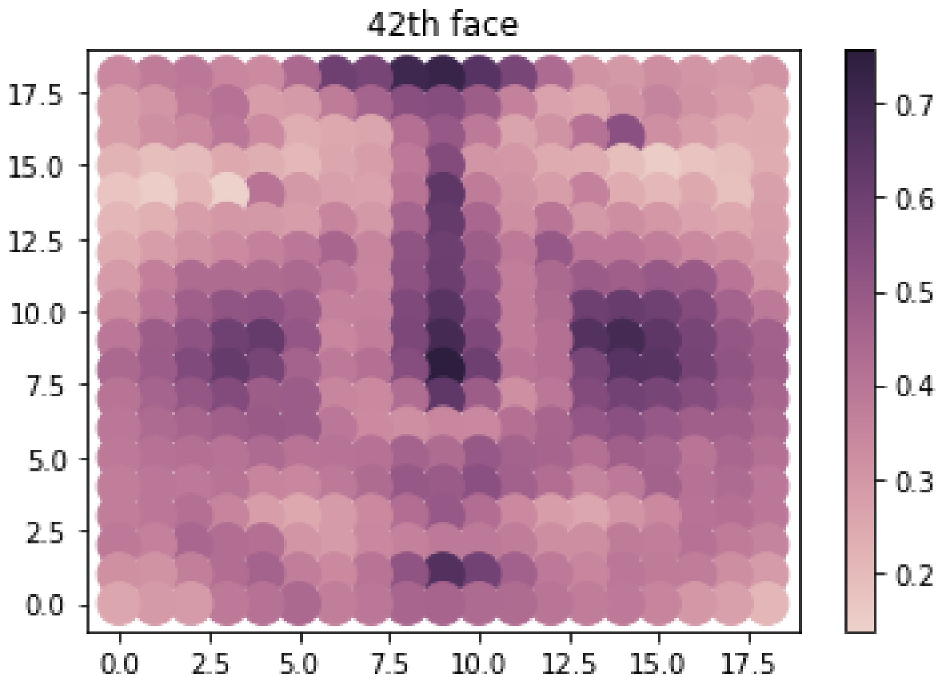
```
In [3]: cbcl = scipy.loadmat('/Users/wyattmadden/Documents/school/' +
                             'MSU/2020/spring/m508/lab_info/lab_3/cbcl.mat',
                            squeeze_me = True)
```

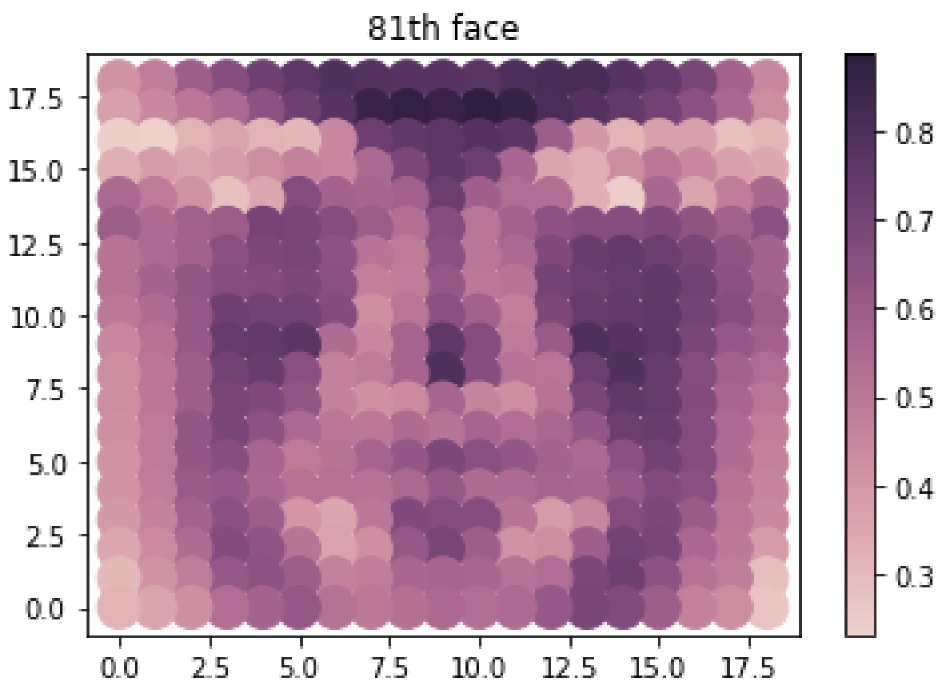
```python
X = cbcl['X']

X_shaped = np.reshape(X, (int(np.sqrt(X.shape[0])),
                              int(np.sqrt(X.shape[0])),
                              X.shape[1]))

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)

rand_ints_25 = random.sample(range(0, 139), 25)

for i in rand_ints_25:
    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")
```

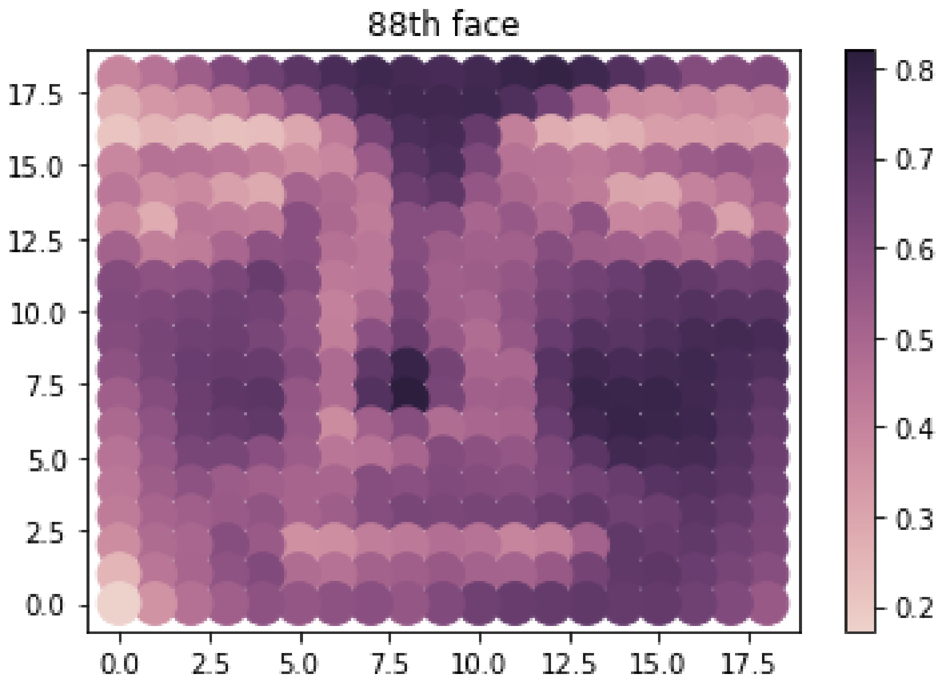
```
/Users/wyattmadden/anaconda3/lib/python3.6/site-packages/matplotlib/pyplot.py:537: RuntimeWarnin
  max_open_warning, RuntimeWarning)
```

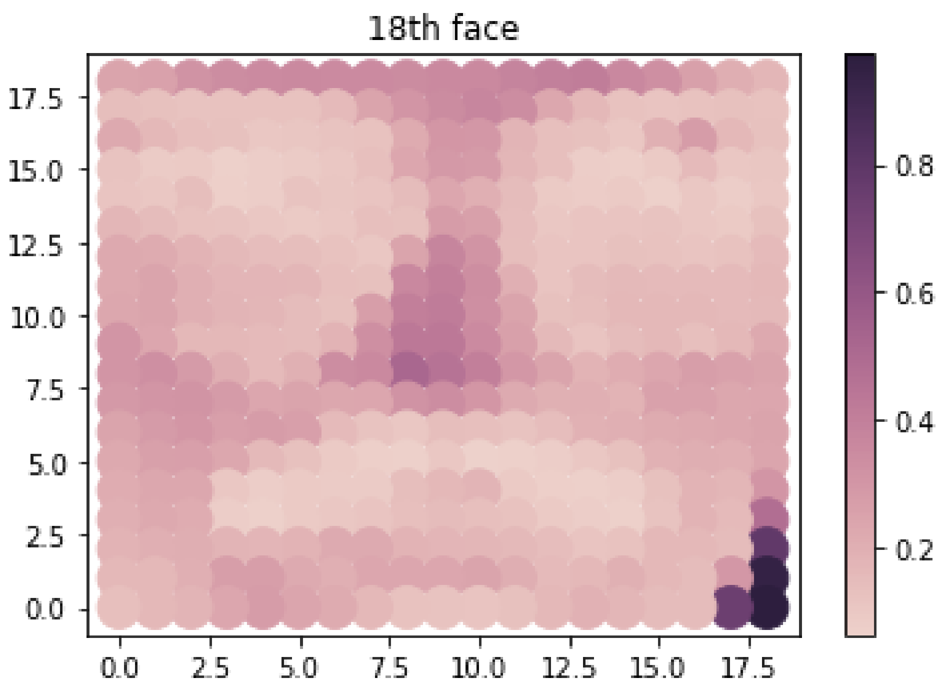

107th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2

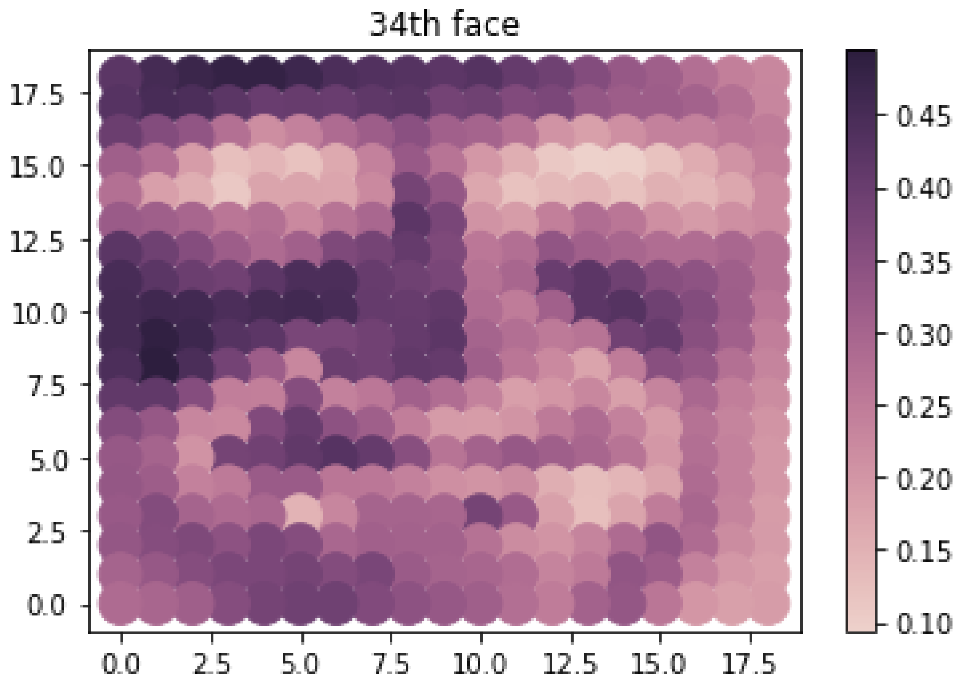

135th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2

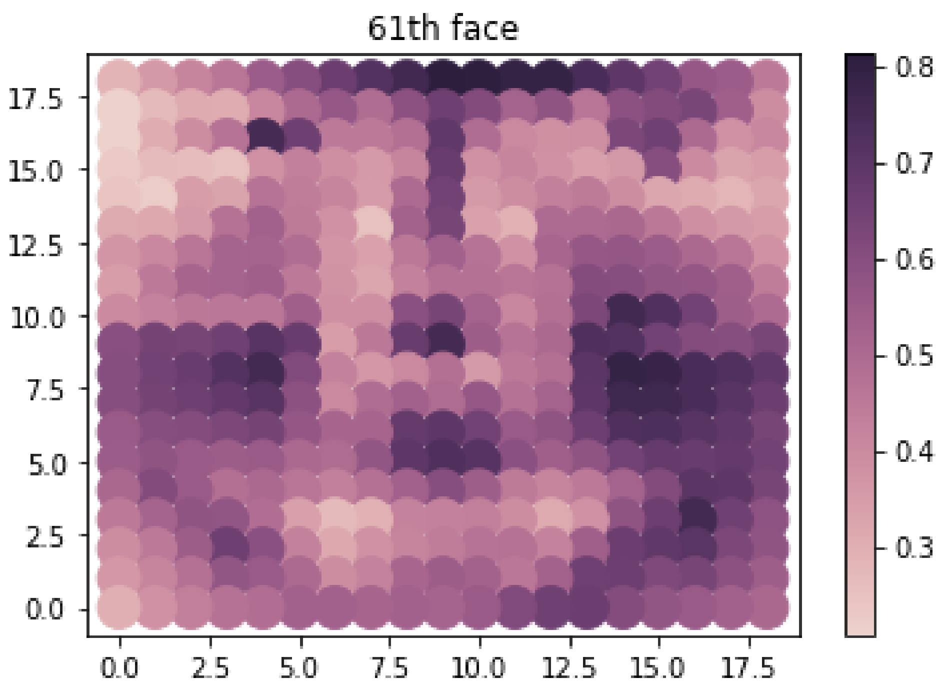

42th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.7
0.6
0.5
0.4
0.3
0.2

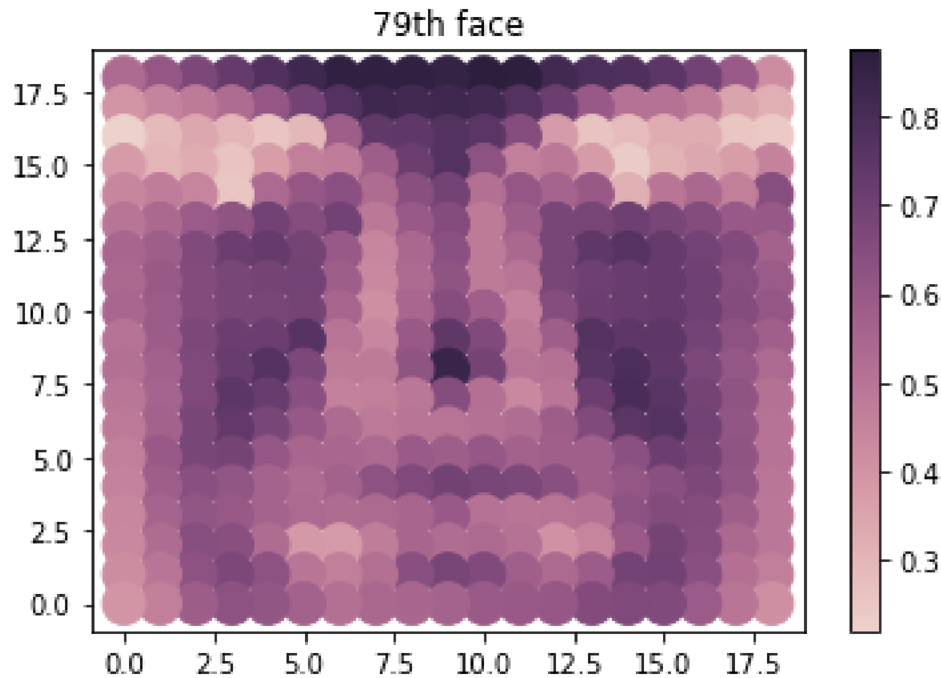

81th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.8
0.7
0.6
0.5
0.4
0.3

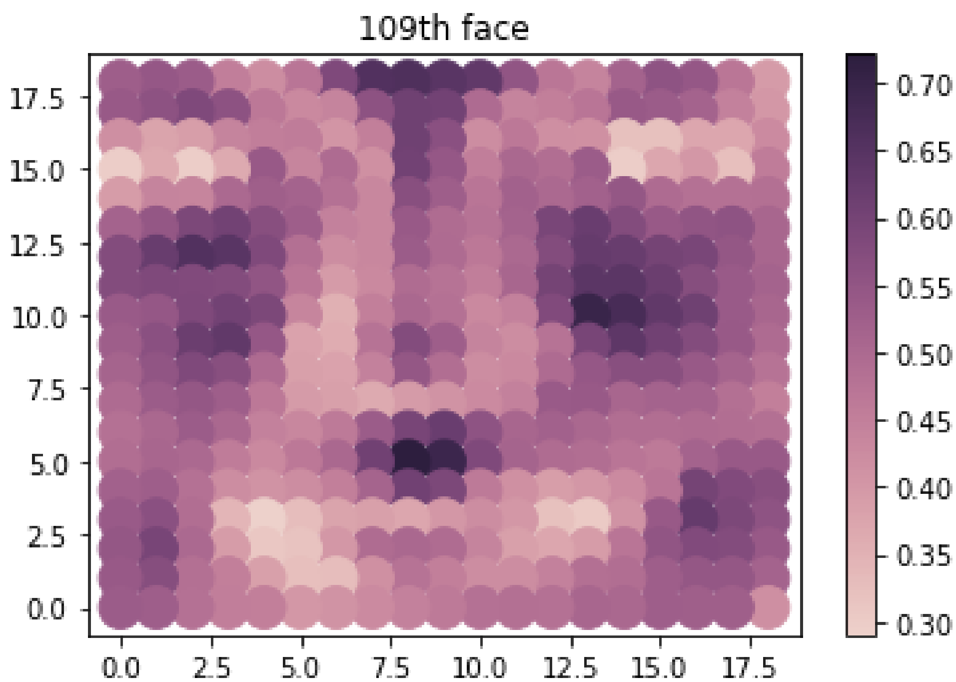

88th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.8
0.7
0.6
0.5
0.4
0.3
0.2

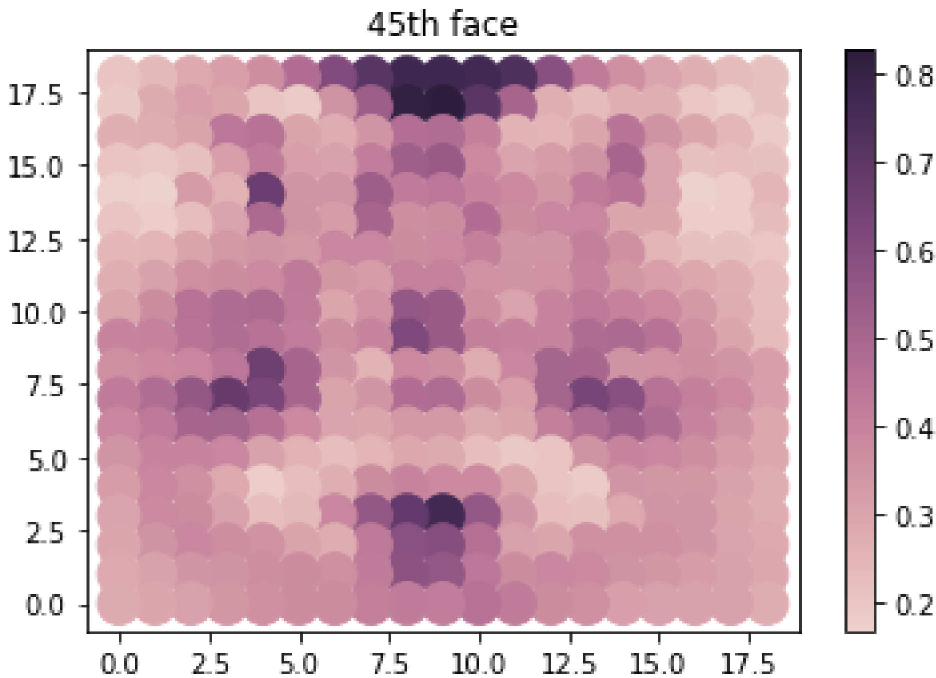

18th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.8
0.6
0.4
0.2

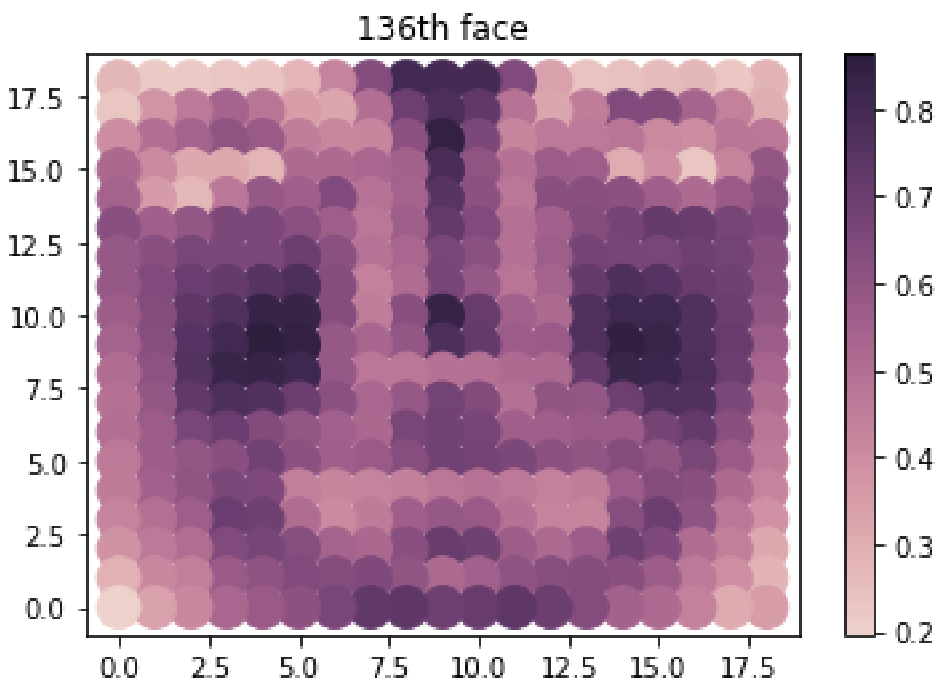

34th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10

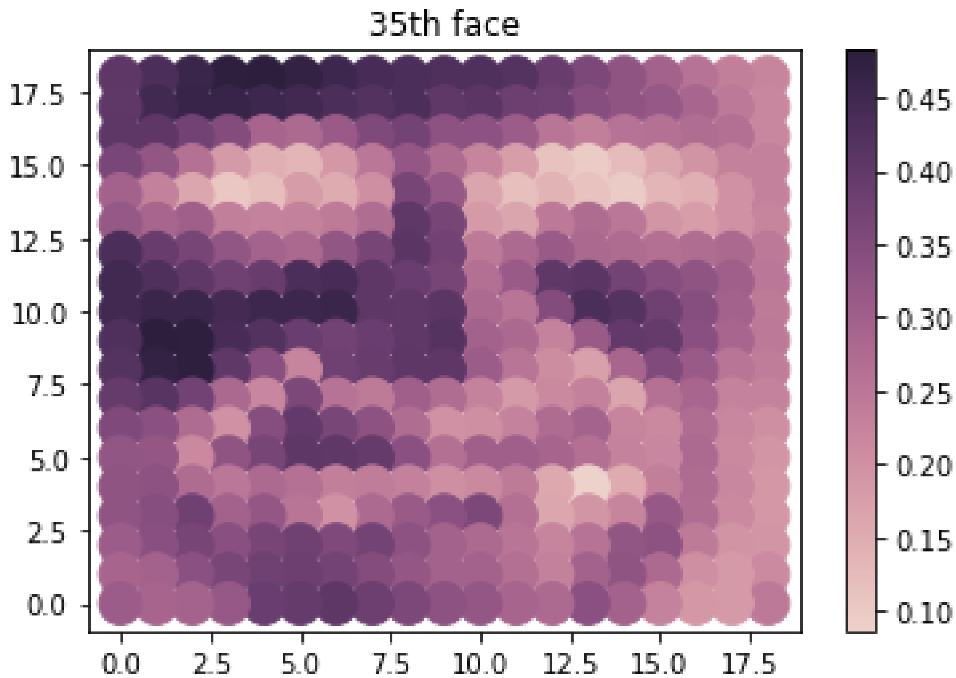

61th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.8
0.7
0.6
0.5
0.4
0.3

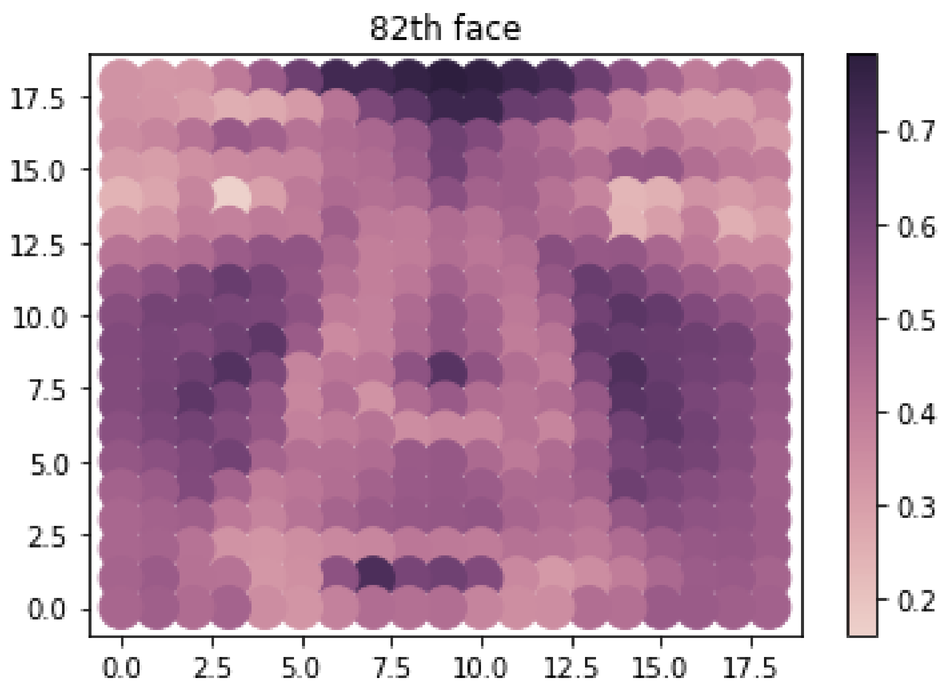

79th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.8
0.7
0.6
0.5
0.4
0.3

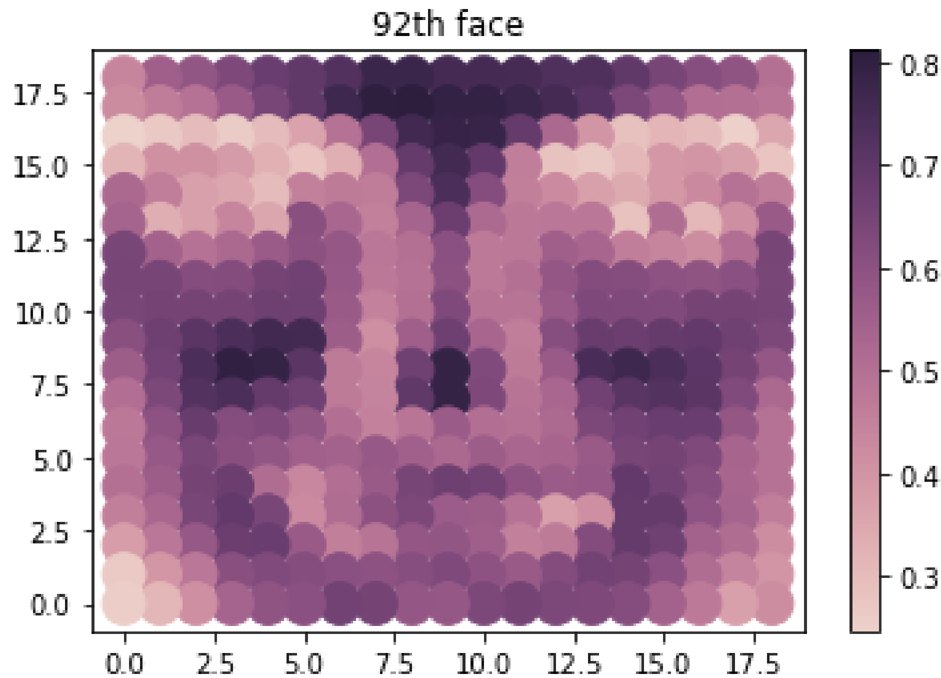

109th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30

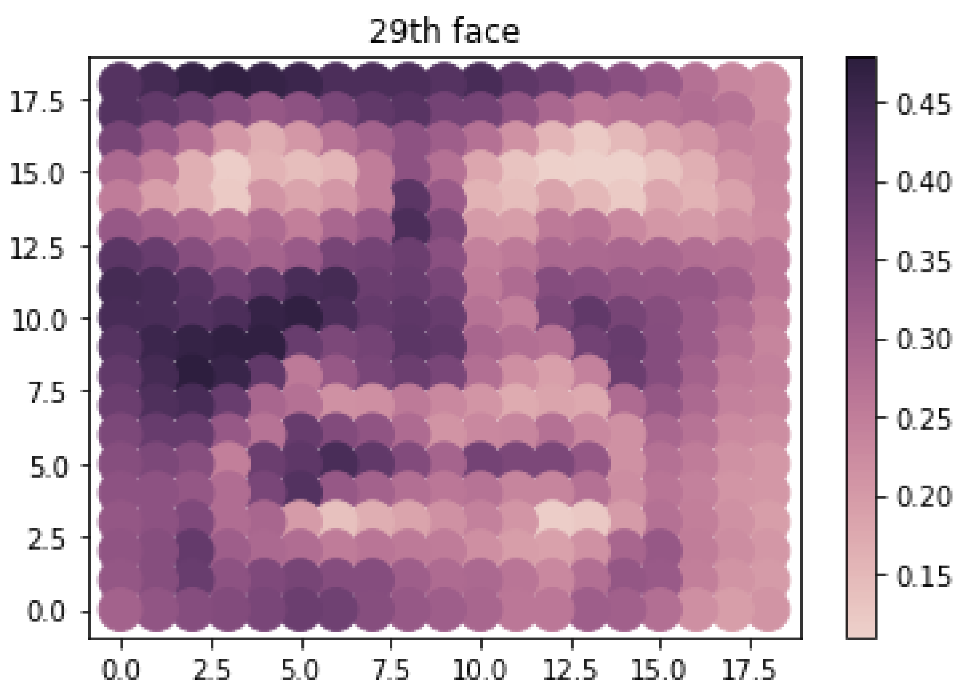

45th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.8
0.7
0.6
0.5
0.4
0.3
0.2

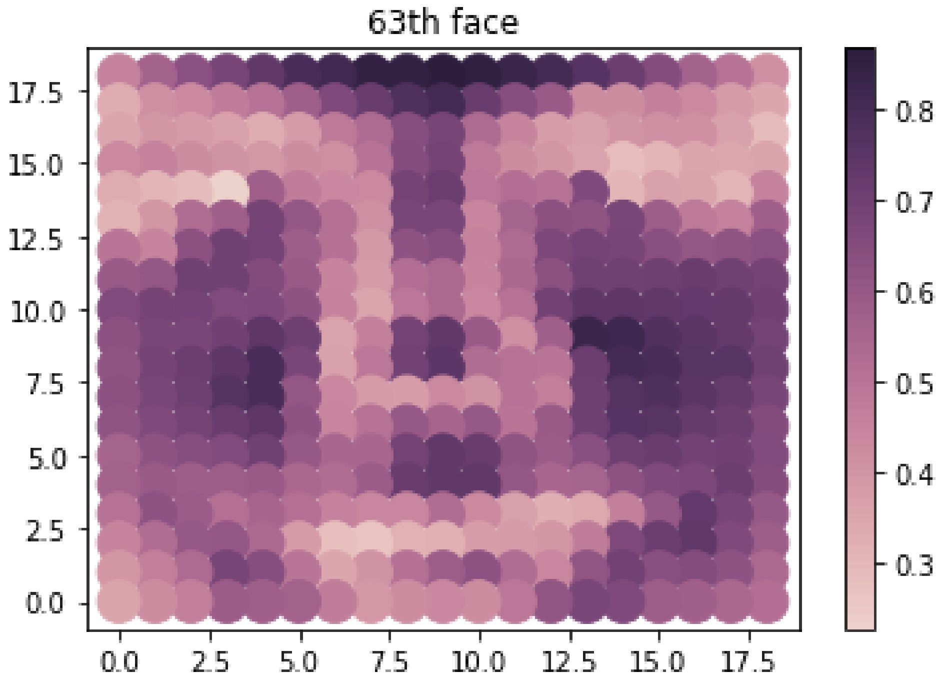

136th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.8
0.7
0.6
0.5
0.4
0.3
0.2

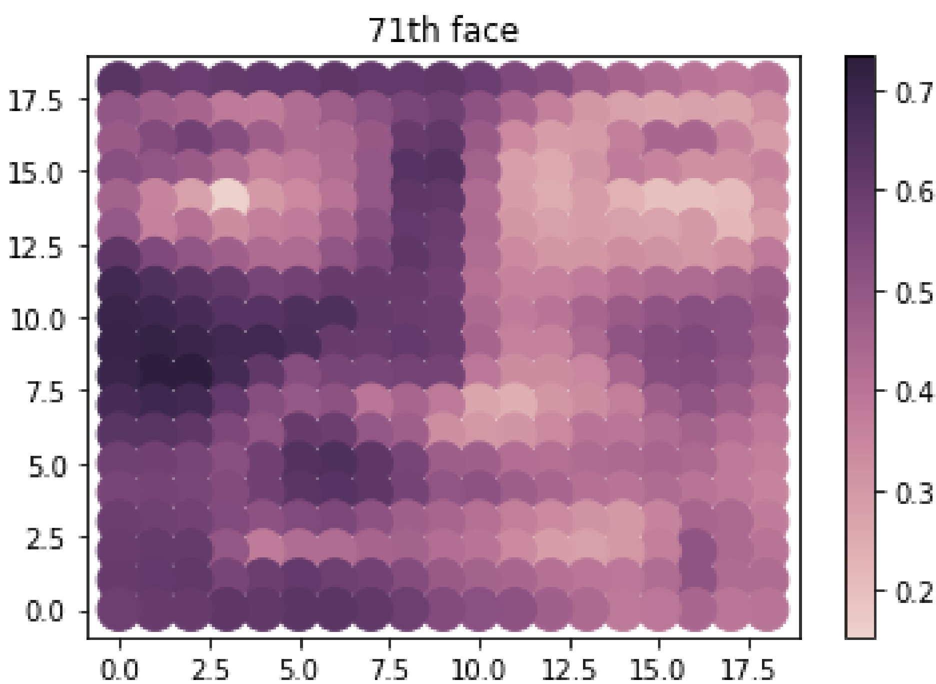

35th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10

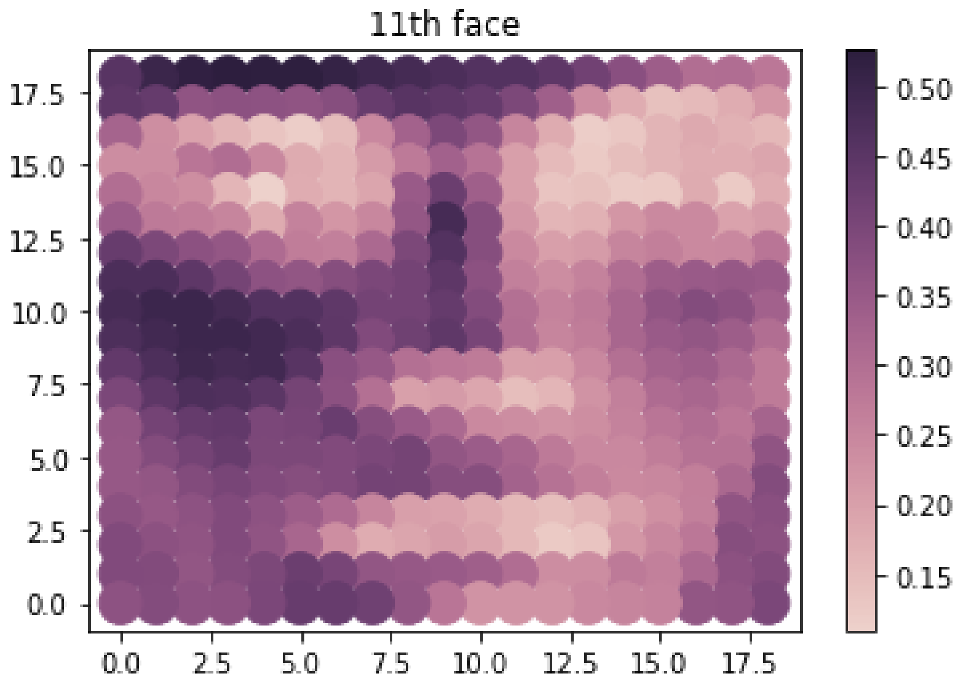

82th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.7
0.6
0.5
0.4
0.3
0.2

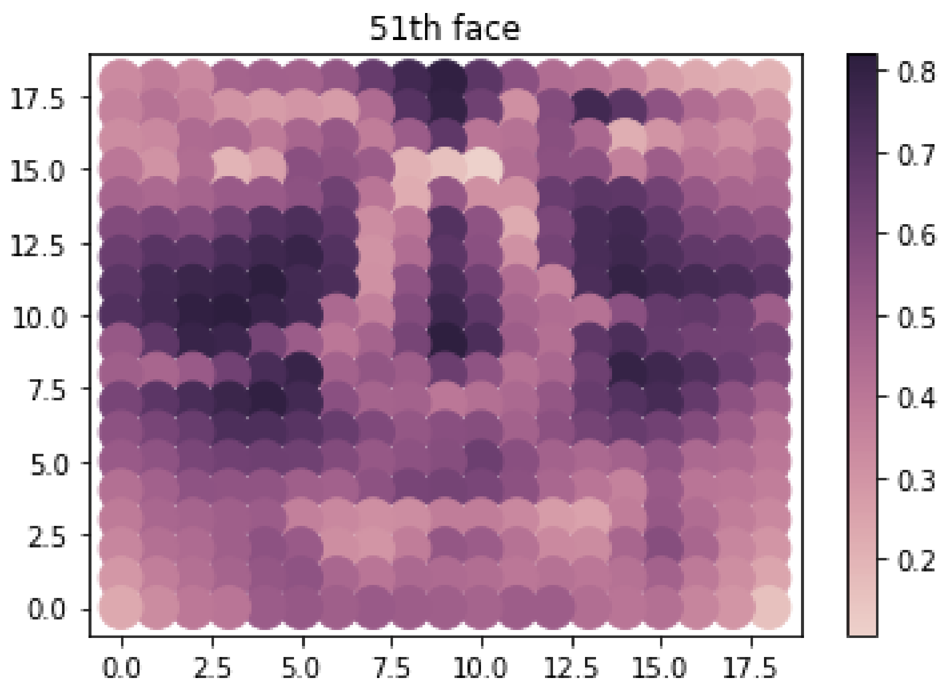

92th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.8
0.7
0.6
0.5
0.4
0.3

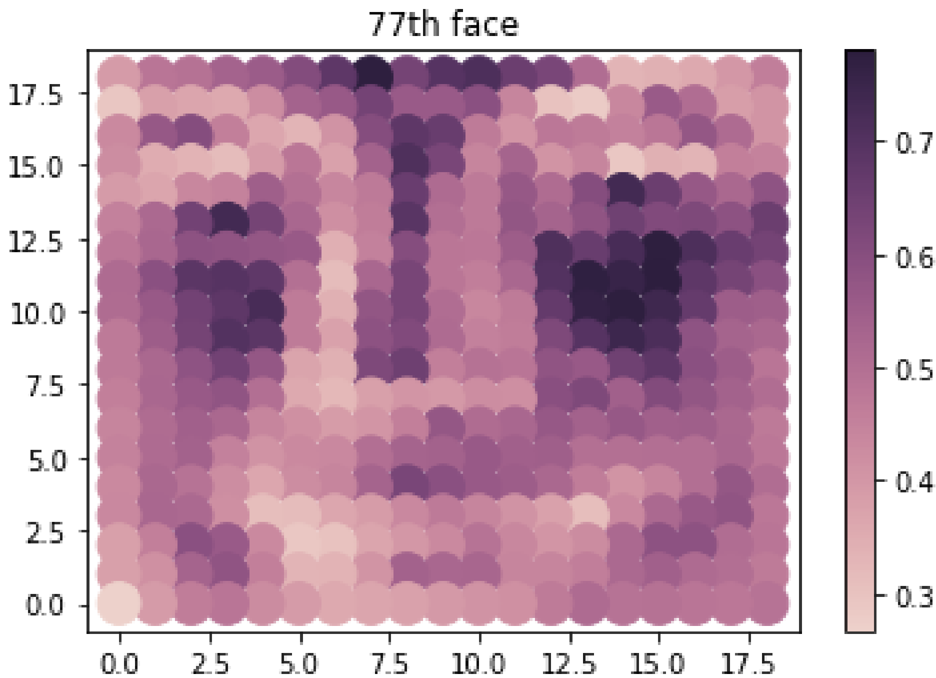

29th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.45
0.40
0.35
0.30
0.25
0.20
0.15

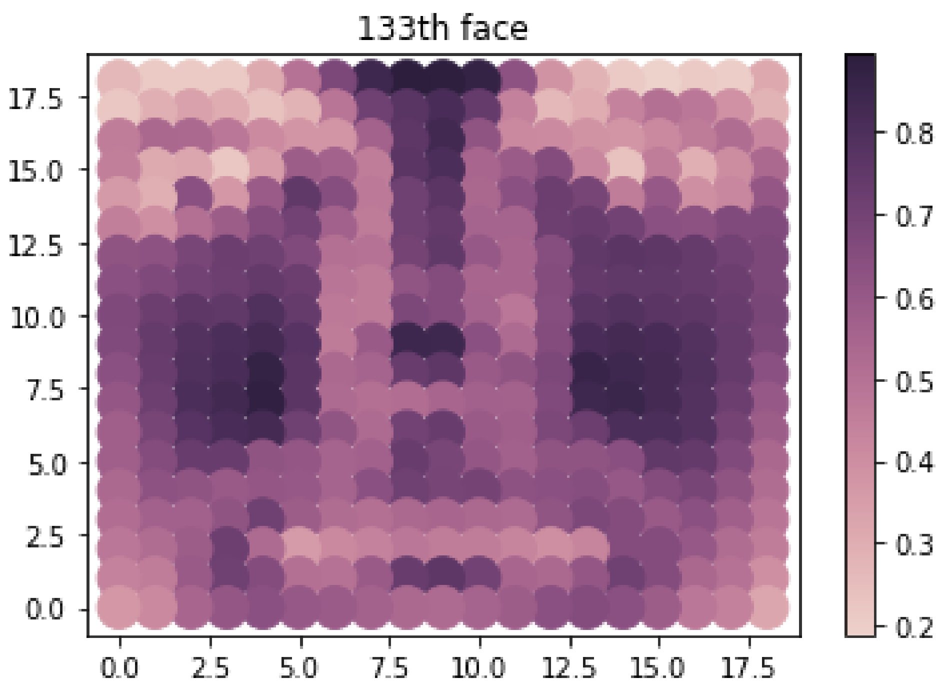

63th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.8
0.7
0.6
0.5
0.4
0.3

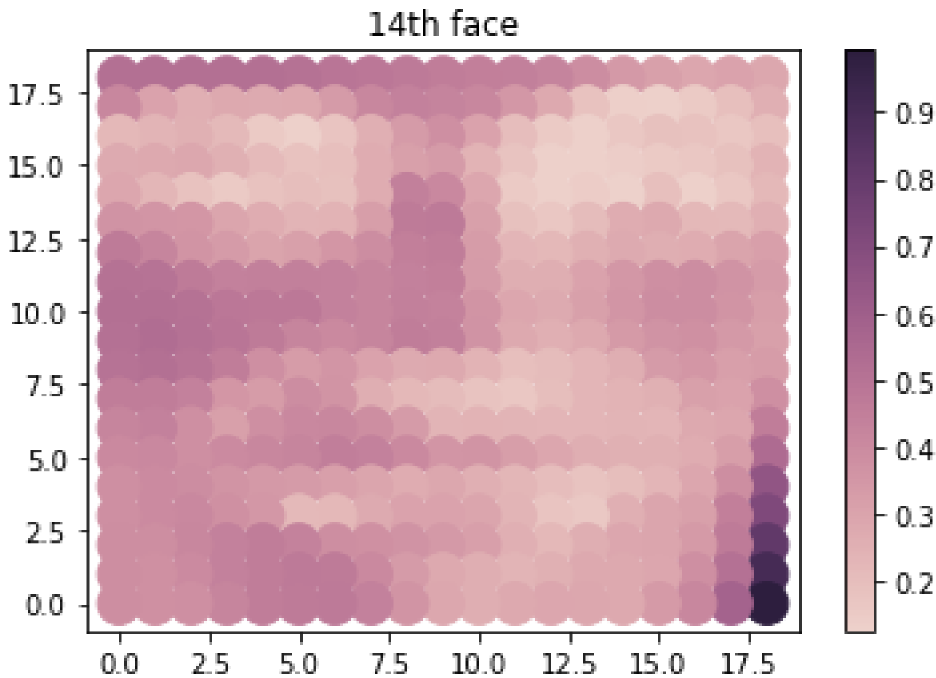

71th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.7
0.6
0.5
0.4
0.3
0.2

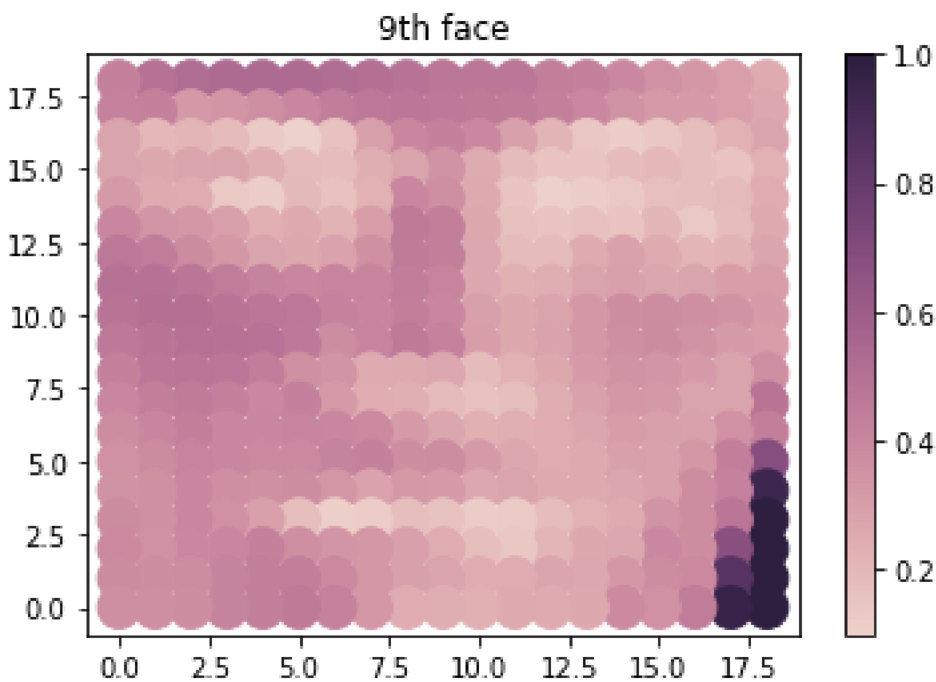

11th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15

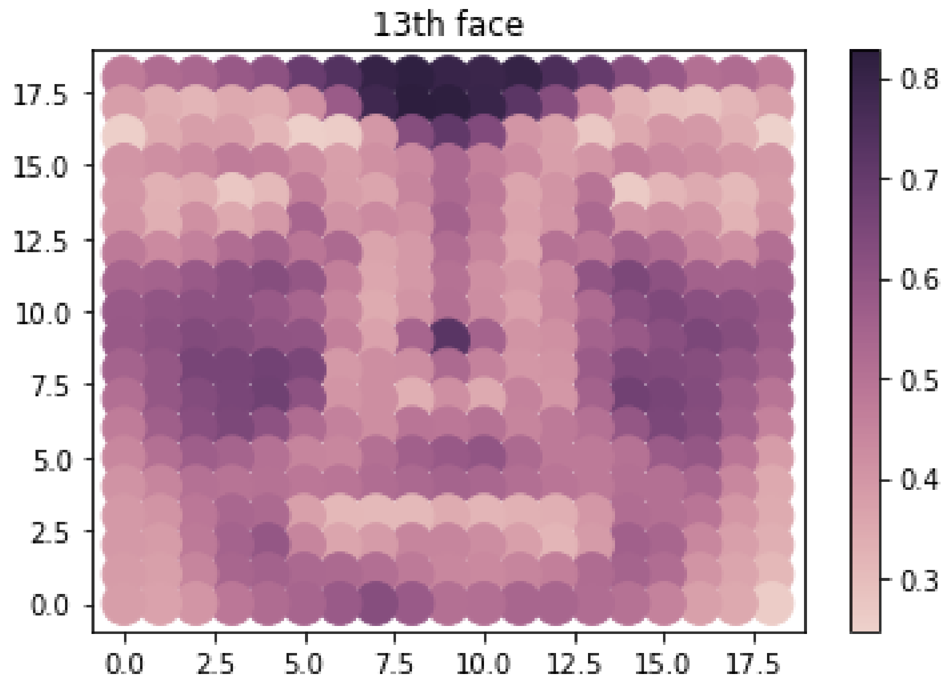

51th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.8
0.7
0.6
0.5
0.4
0.3
0.2


77th 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
0.7
0.6
0.5
0.4
0.3


133th 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
0.8
0.7
0.6
0.5
0.4
0.3
0.2

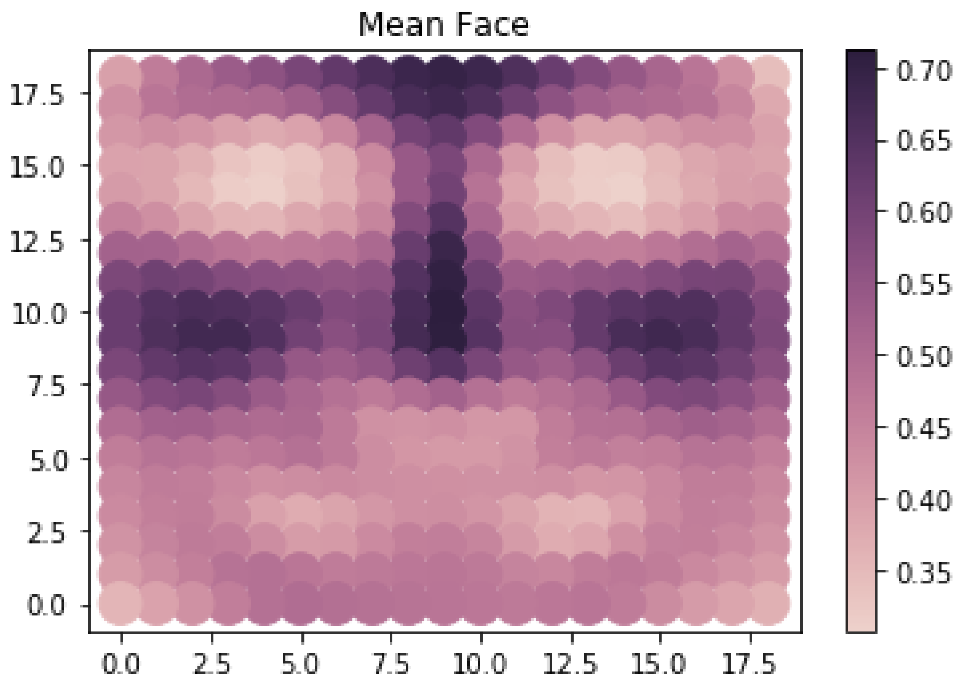

14th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2


9th 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
1.0
0.8
0.6
0.4
0.2

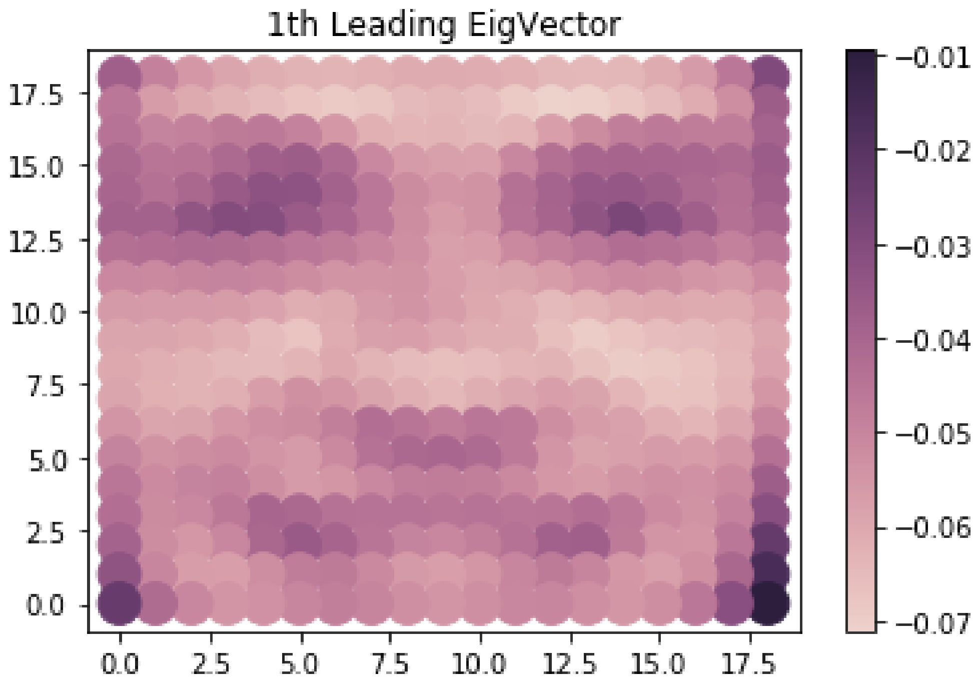


3 3.3

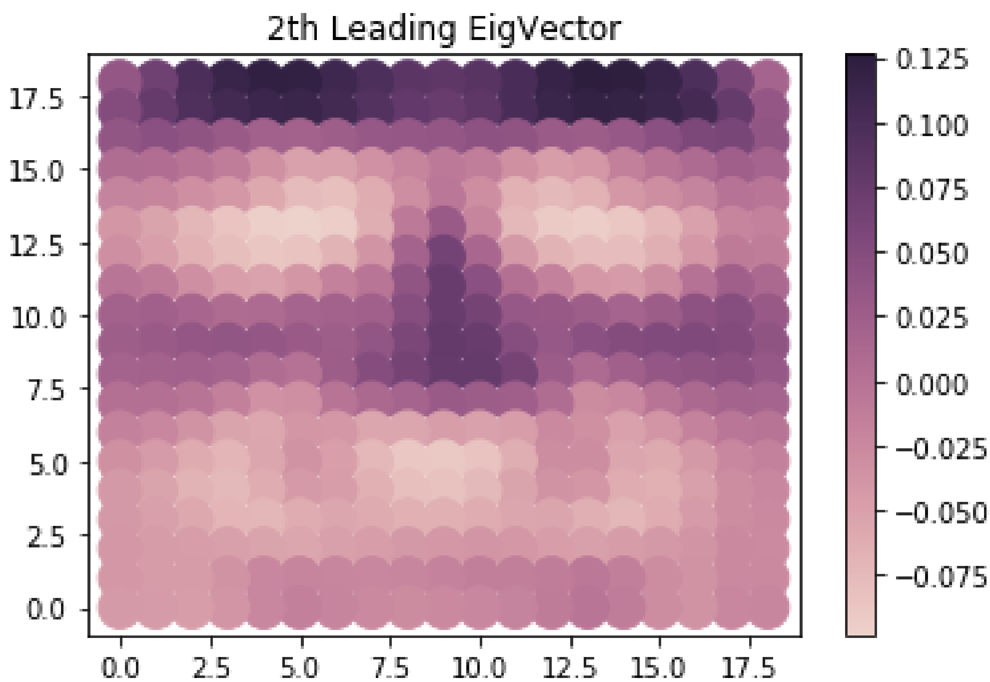
In [4]:
```
W, Z, mu, lambdas = pca(X, 5)

mean_face = mu


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



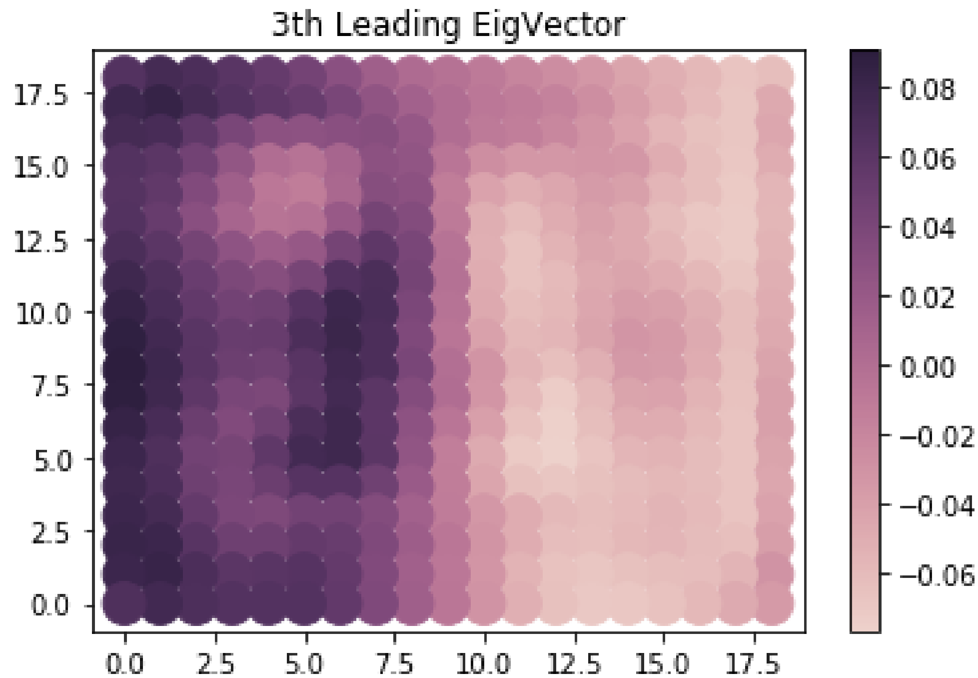

```
In [5]: lead_5_eig_vecs = W
        lead_5_eig_vals = lambdas

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

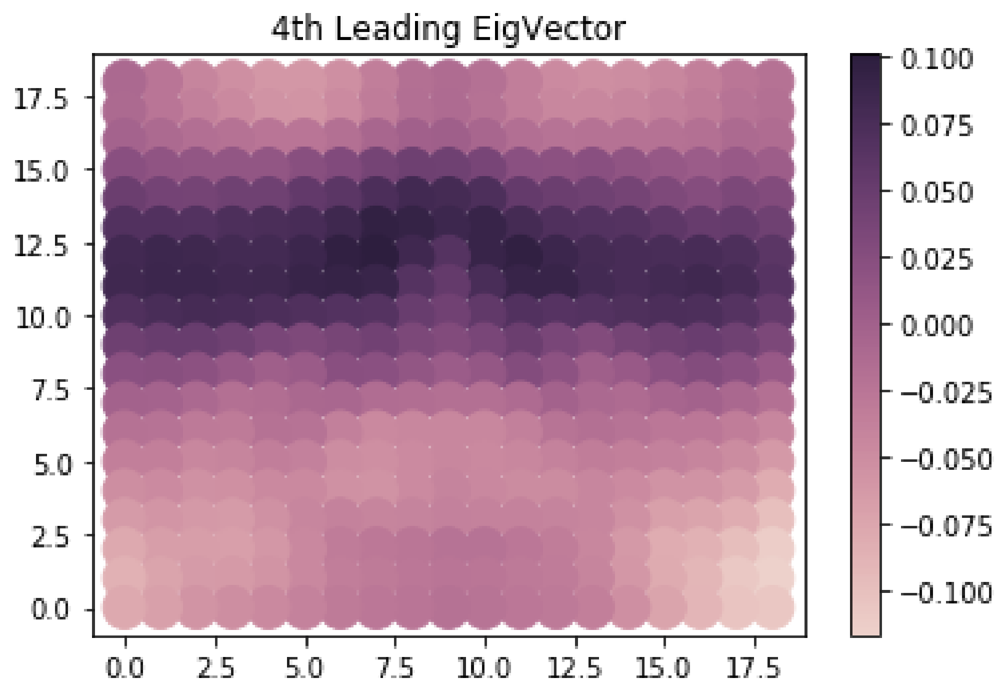


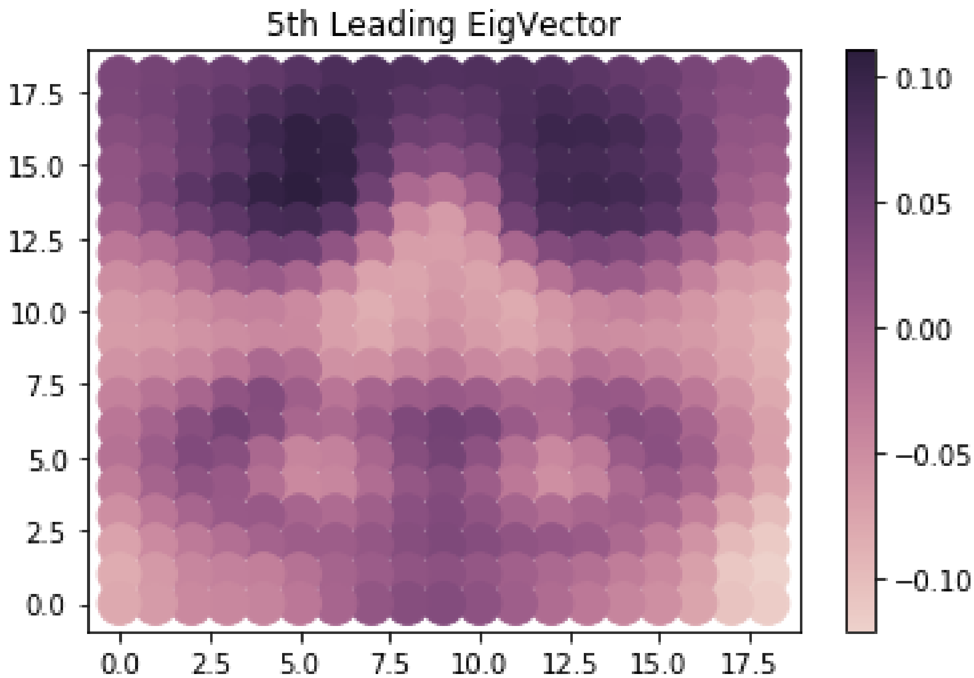
1th Leading EigVector
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−0.07



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
0.125
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075


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
0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06


4th 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



It looks like the eigenvectors corrspond with the follow information:
First: The general construct of a face (shading).
Second: Nose and mouth contours and differences in shading.
Third: Differences in lighting, from left to right.
Fourth: Depth of brow.
Fifth: Location of eyes and mouth expression.

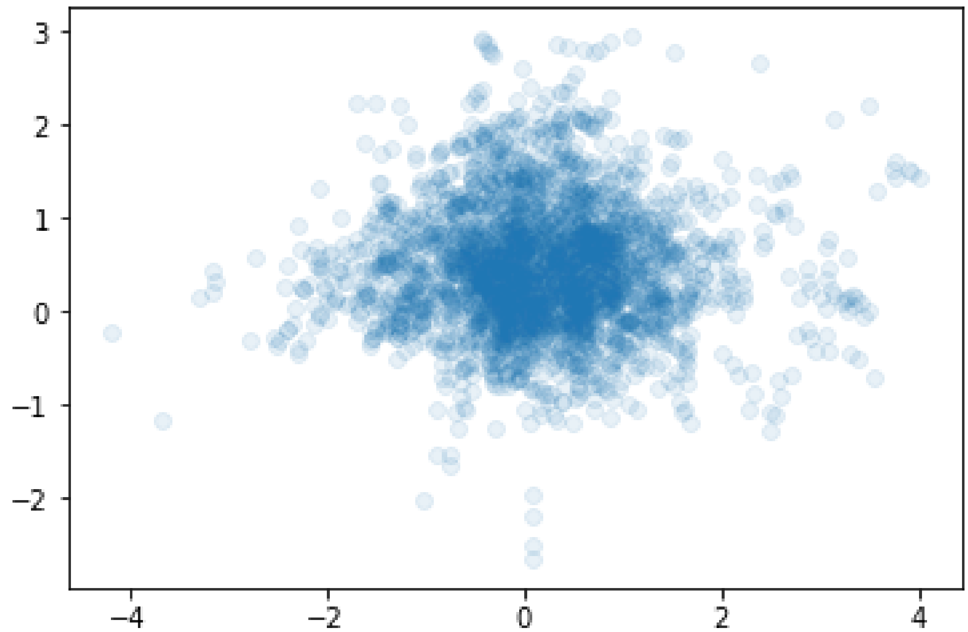
4 3.4

```
In [6]: 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)
```

Looks like there are two possible clusters of data within their respective two leading eigenvectors. One has a positive relationship between eigenvector one and two. One of these clusters are a group of faces that with generally darker overall shading also have generally darker mouth and eye shadding, while the other does not have a very apparent relationship between overall shading and mouth/eye shading.

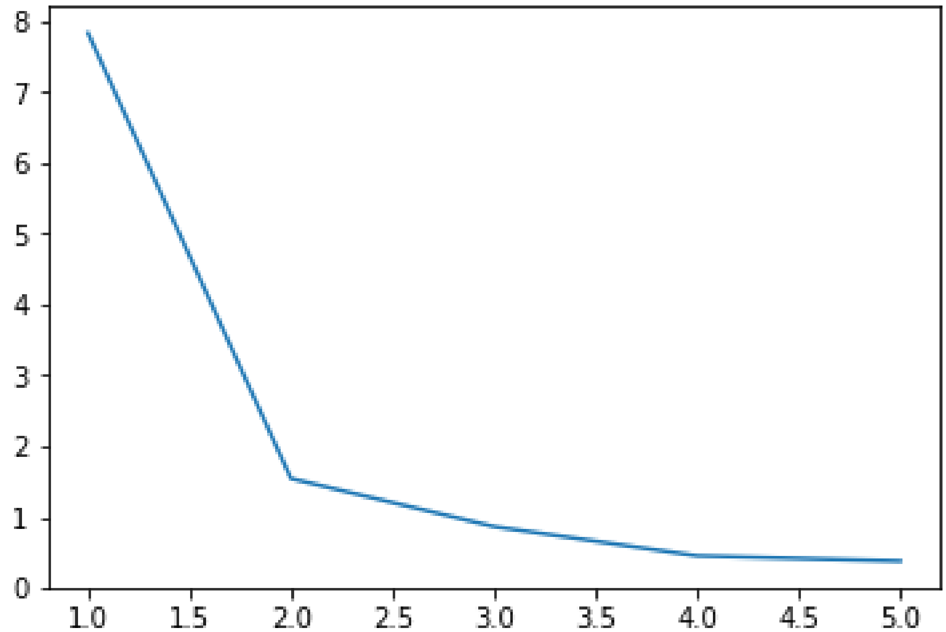
5 3.5

```
In [7]: 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

```
In [8]: plt.plot(list(range(1 ,6)), lead_5_eig_vals)

Out[8]: [<matplotlib.lines.Line2D at 0x1a1fefbe10>]
```



It appears that the first eigenvector captures the majority of variation, and after second, third, and fourth eigenvectors are accounted for, only marginal gains are encountered with further eigenvectors. This suggests to me that for "optimal" dimensionality reduction, M should be at least four.

7 3.7

```
In [72]: rand_ints_25 = np.array(random.sample(range(0, 139), 4))
         m_enumerated = np.array([1, 2, 5, 10, 25])
         x_approx = np.empty(19*19*rand_ints_25.size*m_enumerated.size)


         k = 0
         for i in m_enumerated:
             for j in rand_ints_25:
                 W, Z, mu, lambdas = pca(X, i)
                 x_approx[(k * 19*19):(19*19*(1 + k))] = np.matmul(W, Z)[:, j] + mu
                 k += 1

         faces = {'M': np.repeat(m_enumerated, rand_ints_25.size*19*19),
                 'random_faces': np.tile(np.repeat(rand_ints_25, 19*19), m_enumerated.size),
                 'x_coord': np.tile(x_axis_points, rand_ints_25.size*m_enumerated.size),
                 'y_coord': np.tile(y_axis_points, rand_ints_25.size*m_enumerated.size),
                 'faces_approx': np.asarray(x_approx)}

         faces = pand.DataFrame(faces)
```

Out[72]:

	M	random_faces	x_coord	y_coord	faces_approx
0	1	25	18	18	0.363026
1	1	25	18	17	0.404401
2	1	25	18	16	0.418836
3	1	25	18	15	0.414741
4	1	25	18	14	0.431306
5	1	25	18	13	0.469787
6	1	25	18	12	0.527512
7	1	25	18	11	0.584590
8	1	25	18	10	0.617562
9	1	25	18	9	0.626386
10	1	25	18	8	0.607084
11	1	25	18	7	0.570492
12	1	25	18	6	0.527732
13	1	25	18	5	0.491100
14	1	25	18	4	0.467950
15	1	25	18	3	0.455584
16	1	25	18	2	0.437411
17	1	25	18	1	0.414428
18	1	25	18	0	0.373515
19	1	25	17	18	0.454016
20	1	25	17	17	0.483078
21	1	25	17	16	0.459860
22	1	25	17	15	0.426264
23	1	25	17	14	0.430649
24	1	25	17	13	0.468876
25	1	25	17	12	0.552107
26	1	25	17	11	0.632521
27	1	25	17	10	0.668545
28	1	25	17	9	0.672978
29	1	25	17	8	0.649516
...	..	...	...	...	...
7190	25	43	1	10	0.558612
7191	25	43	1	9	0.573366
7192	25	43	1	8	0.569104
7193	25	43	1	7	0.546560
7194	25	43	1	6	0.498241
7195	25	43	1	5	0.439064
7196	25	43	1	4	0.405226
7197	25	43	1	3	0.405198
7198	25	43	1	2	0.410108
7199	25	43	1	1	0.395521
7200	25	43	1	0	0.357467
7201	25	43	0	18	0.282639
7202	25	43	0	17	0.291208
7203	25	43	0	16	0.300475
7204	25	43	0	15	0.270345
7205	25	43	0	14	0.229803

```
7206  25            43        0       13      0.287186
7207  25            43        0       12      0.384215
7208  25            43        0       11      0.469835
7209  25            43        0       10      0.515413
7210  25            43        0        9      0.527880
7211  25            43        0        8      0.520276
7212  25            43        0        7      0.493546
7213  25            43        0        6      0.450504
7214  25            43        0        5      0.405428
7215  25            43        0        4      0.381773
7216  25            43        0        3      0.374712
7217  25            43        0        2      0.362616
7218  25            43        0        1      0.337710
7219  25            43        0        0      0.289022

[7220 rows x 5 columns]
```

In [73]:
```
g = sns.FacetGrid(faces, col = "M", row = "random_faces", hue = "faces_approx", palette
g.map(plt.scatter, "x_coord", "y_coord"
```

```
0, s = 250)
```

Out[73]: <seaborn.axisgrid.FacetGrid at 0x1a2dcfb898>