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
```

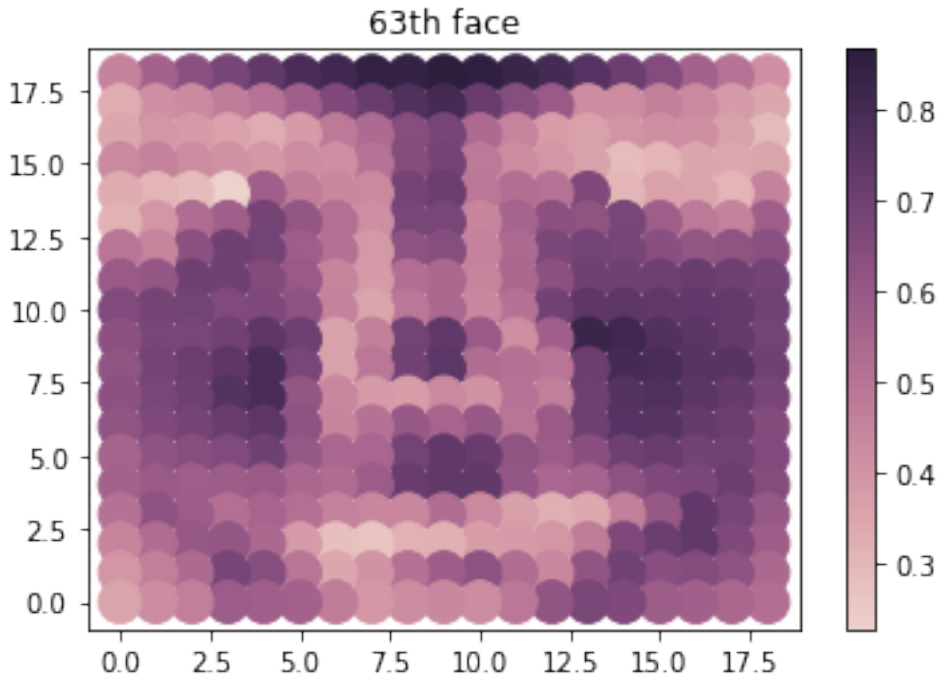
In [125]:
```
# PRINcipal COMPonent calculator
#   Calculates the principal components of a collection of points.
# Input:
#   X - D-by-N data matrix of N points in D dimensions.
# Output:
#   W - A D-by-M matrix containing the M principal components of the data.
#   Z - A M-by-N matrix containing the 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))

    return W, Z, mu, lambdas
```

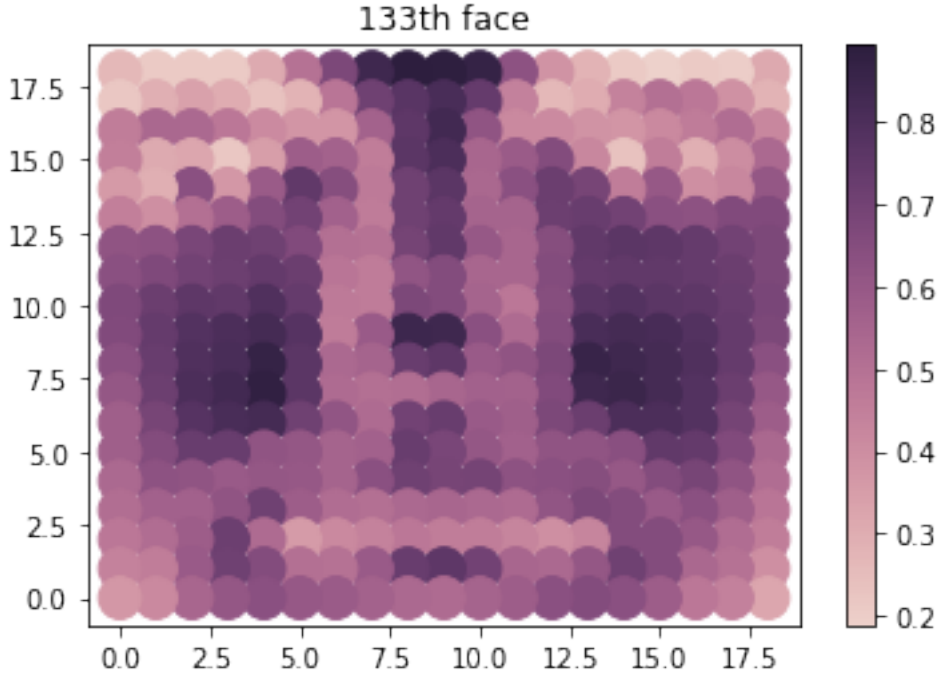
2 3.2

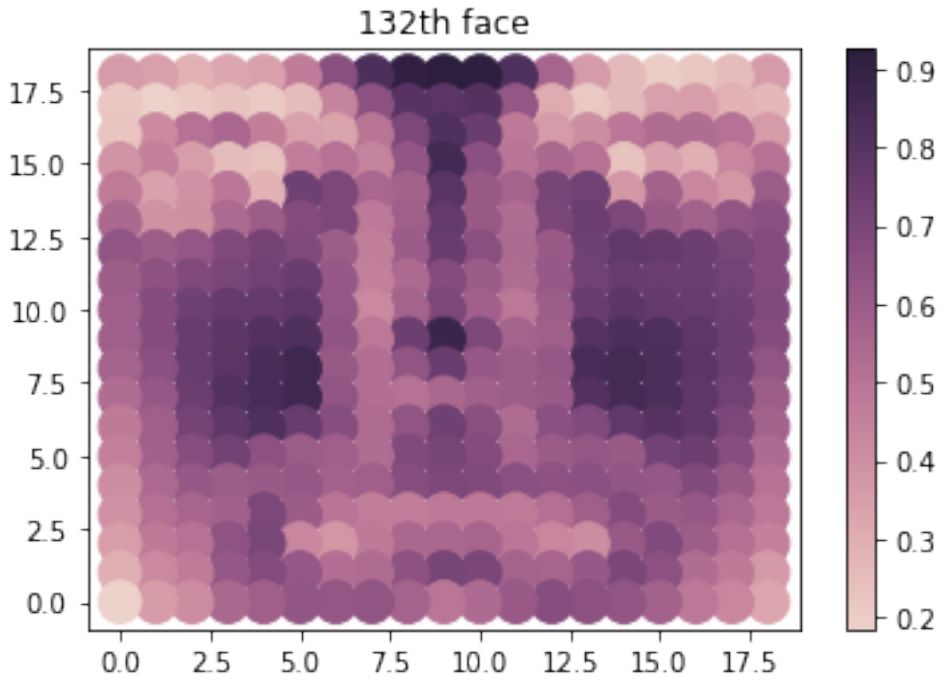
In [126]:
```
cbcl = scipy.loadmat('/Users/wyattmadden/Documents/school/' +
                     'MSU/2020/spring/m508/lab_info/lab_3/cbcl.mat',
                    squeeze_me = True)
```

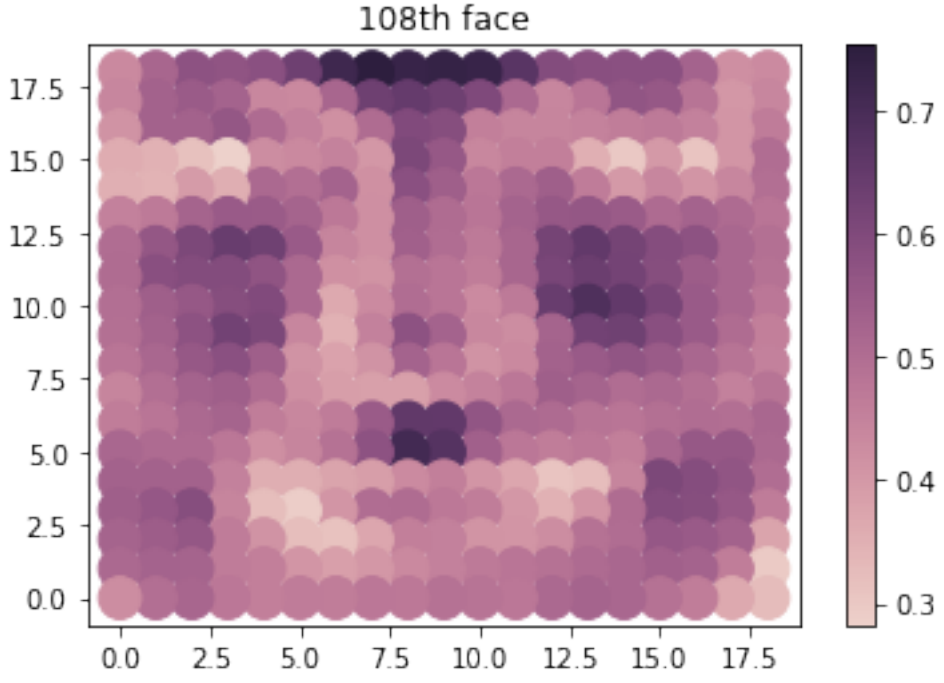
```
X = cbcl['X']

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")
```

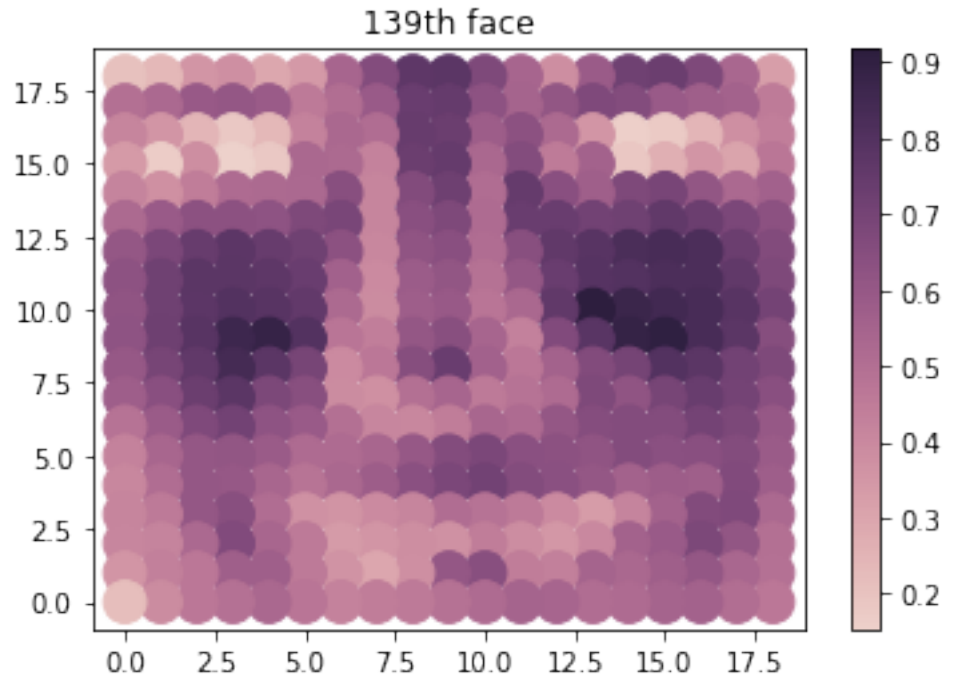
```
/Users/wyattmadden/anaconda3/lib/python3.6/site-packages/matplotlib/pyplot.py:537: RuntimeWarnin
  max_open_warning, RuntimeWarning)
```

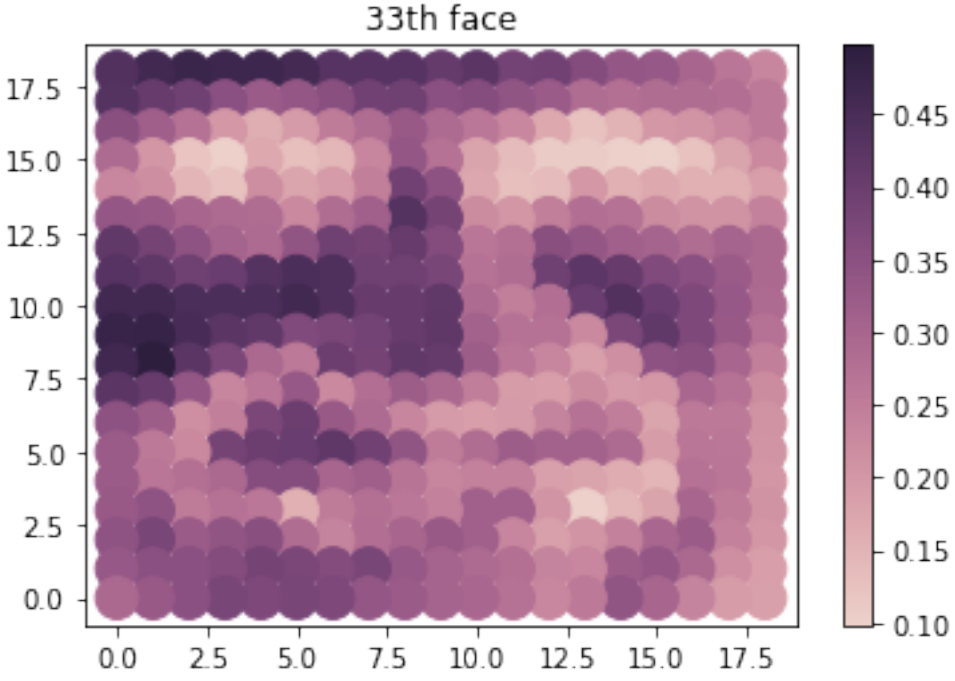

63th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.8
0.7
0.6
0.5
0.4
0.3

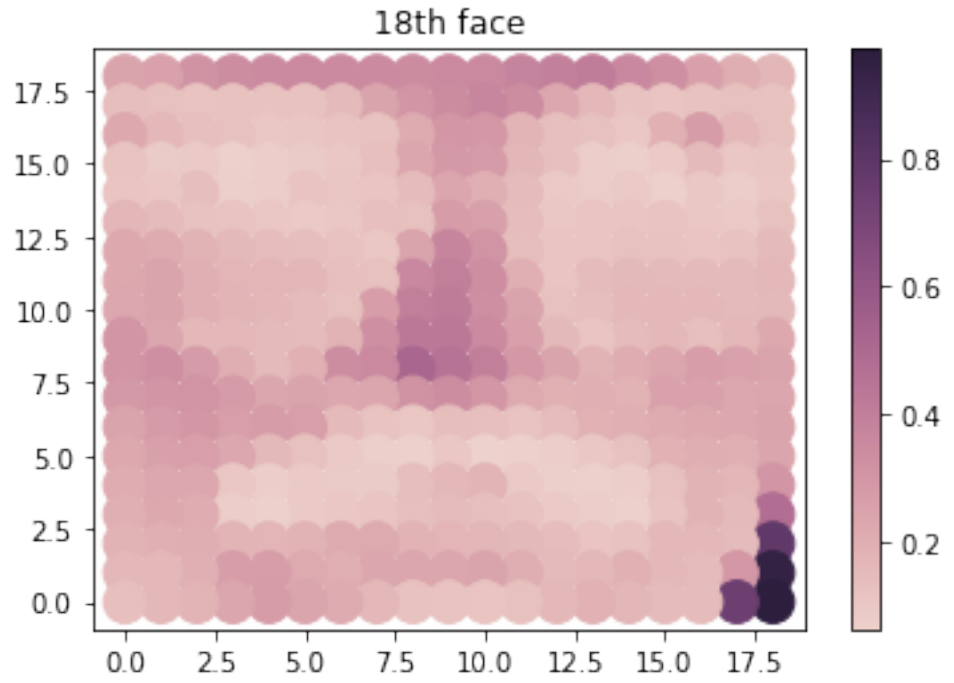

133th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.8
0.7
0.6
0.5
0.4
0.3
0.2

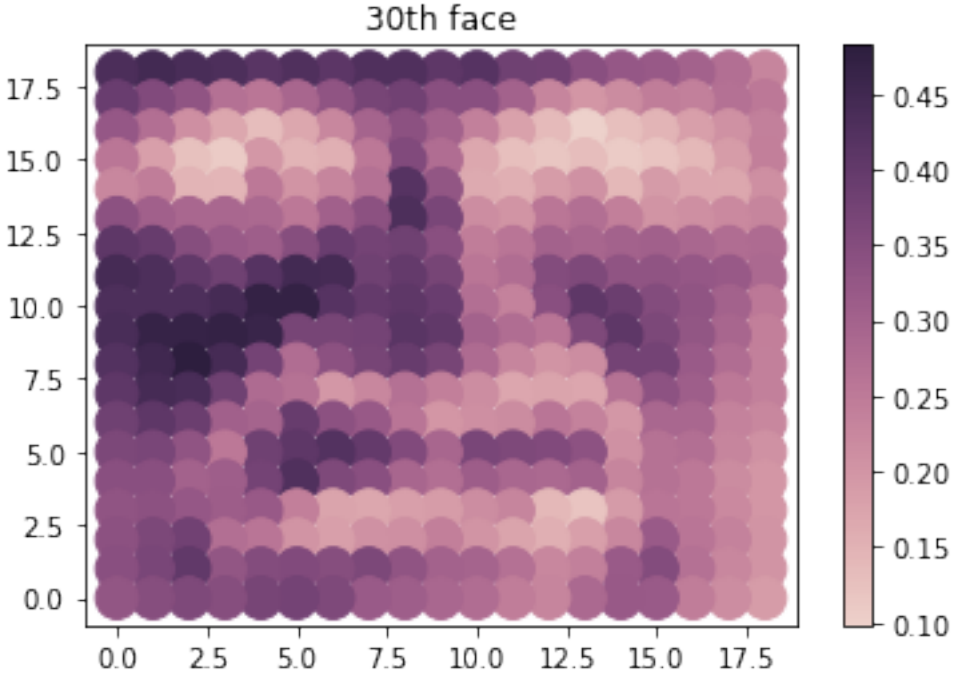

132th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2

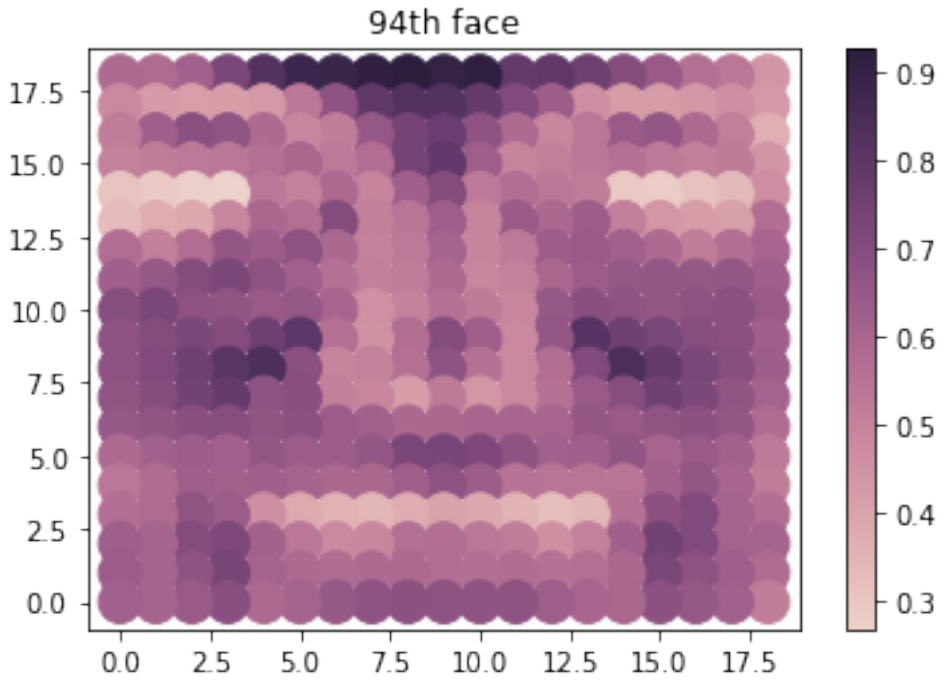

108th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.7
0.6
0.5
0.4
0.3

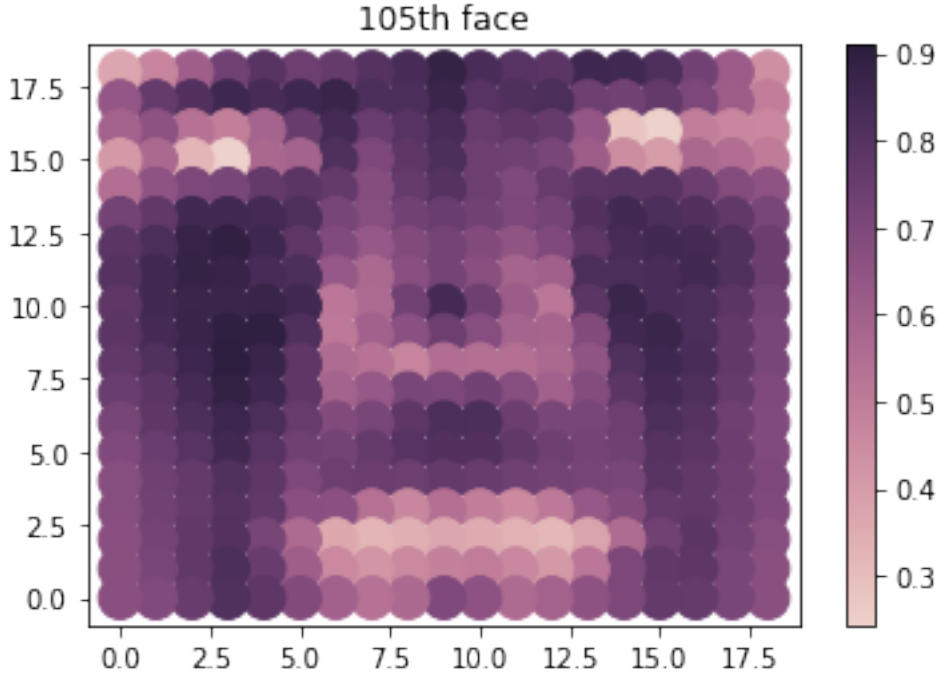

139th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2

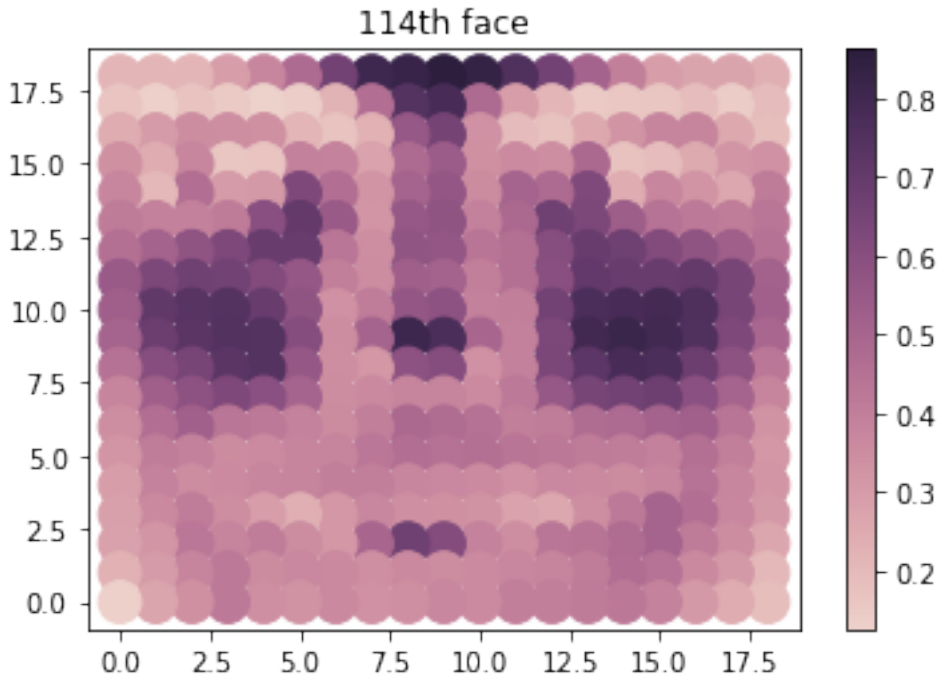

33th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10

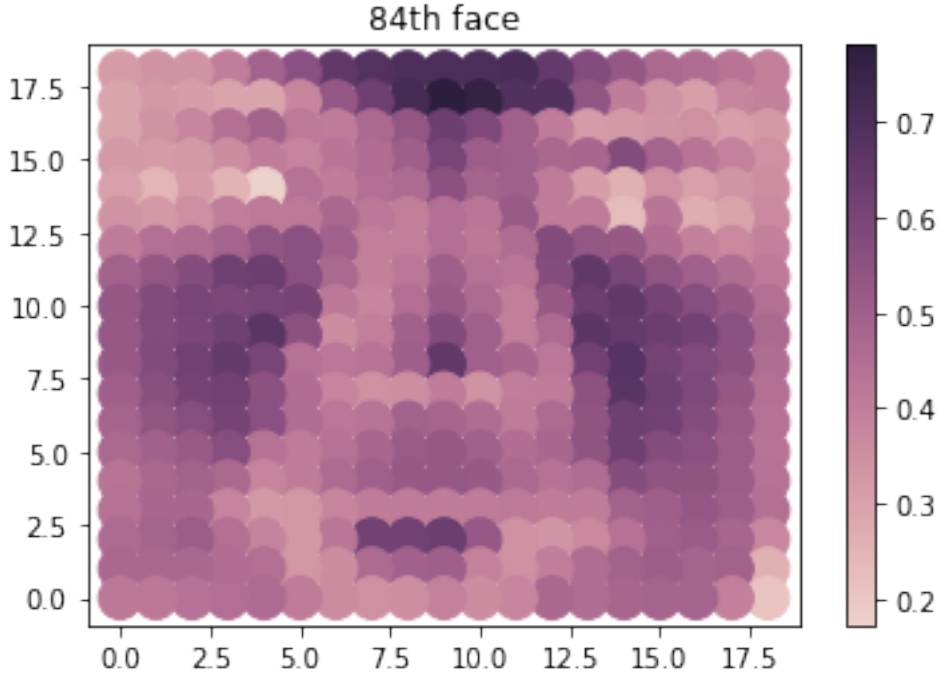

18th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.8
0.6
0.4
0.2

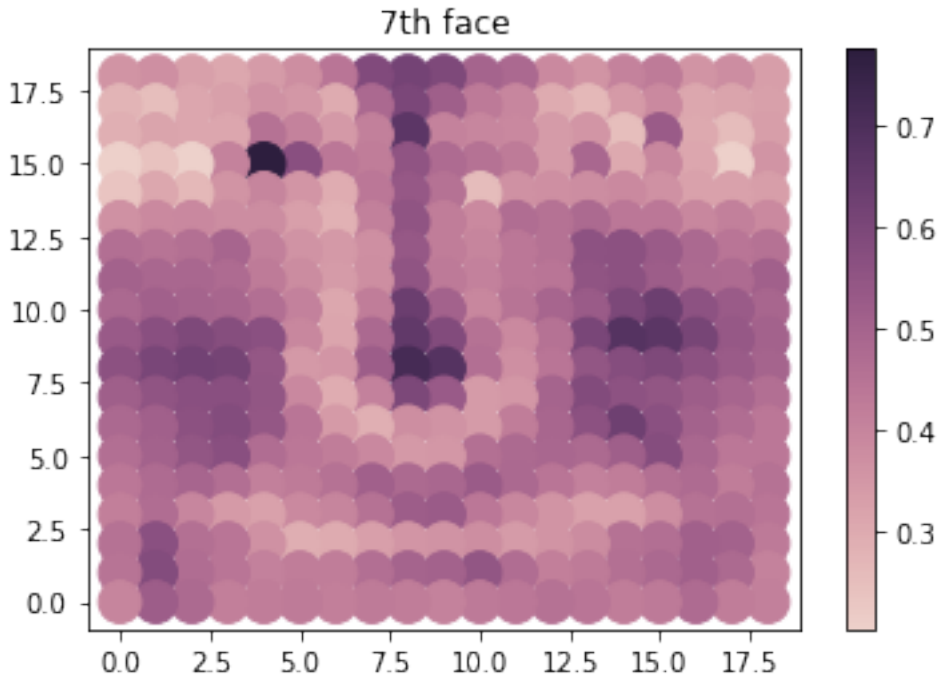

30th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10

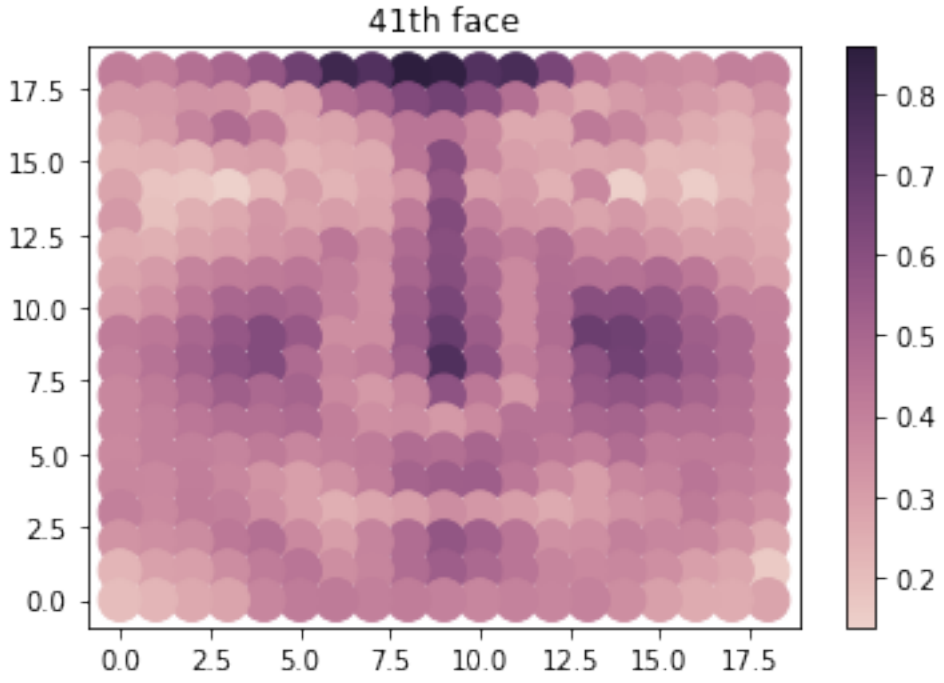

94th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.9
0.8
0.7
0.6
0.5
0.4
0.3

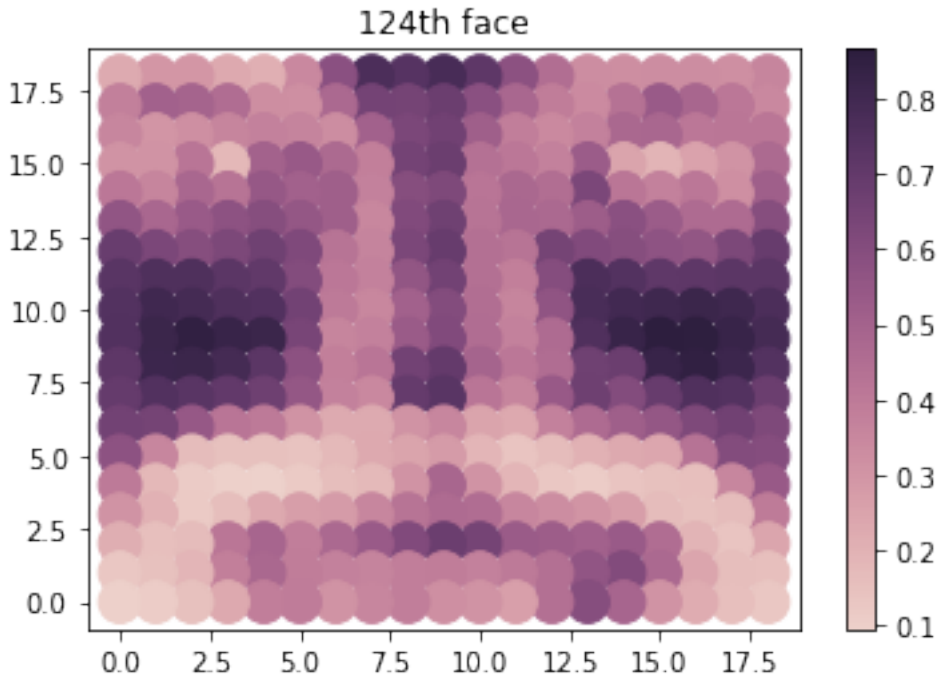

105th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.9
0.8
0.7
0.6
0.5
0.4
0.3

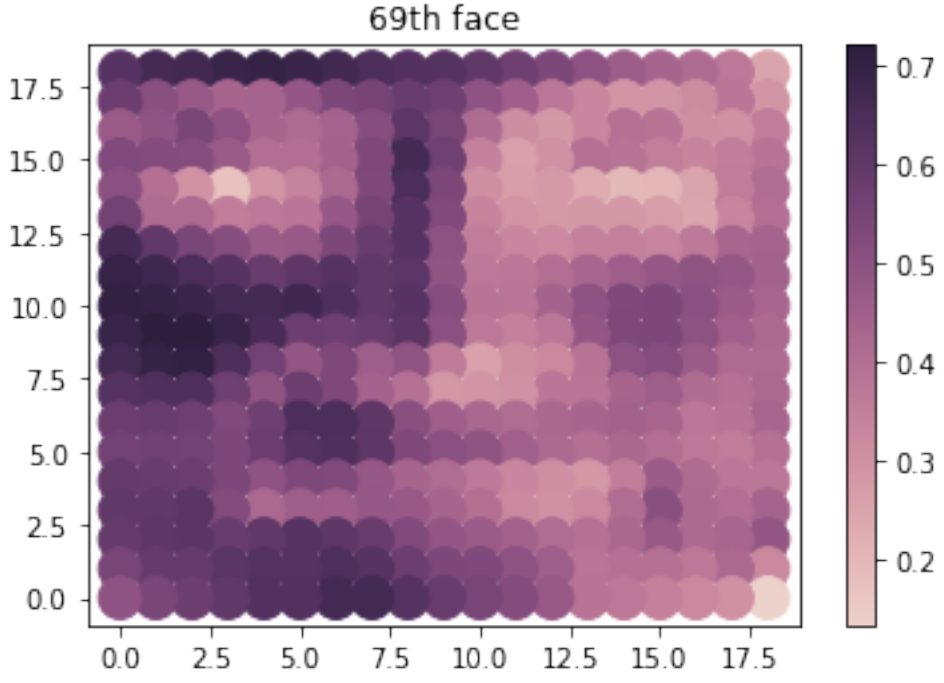

114th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.8
0.7
0.6
0.5
0.4
0.3
0.2

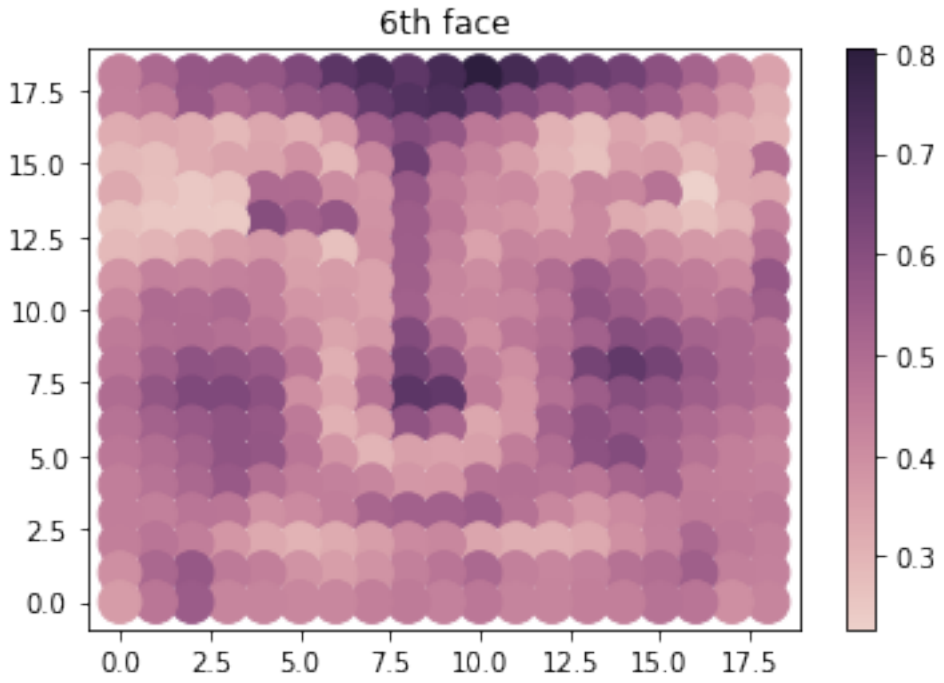

84th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.7
0.6
0.5
0.4
0.3
0.2

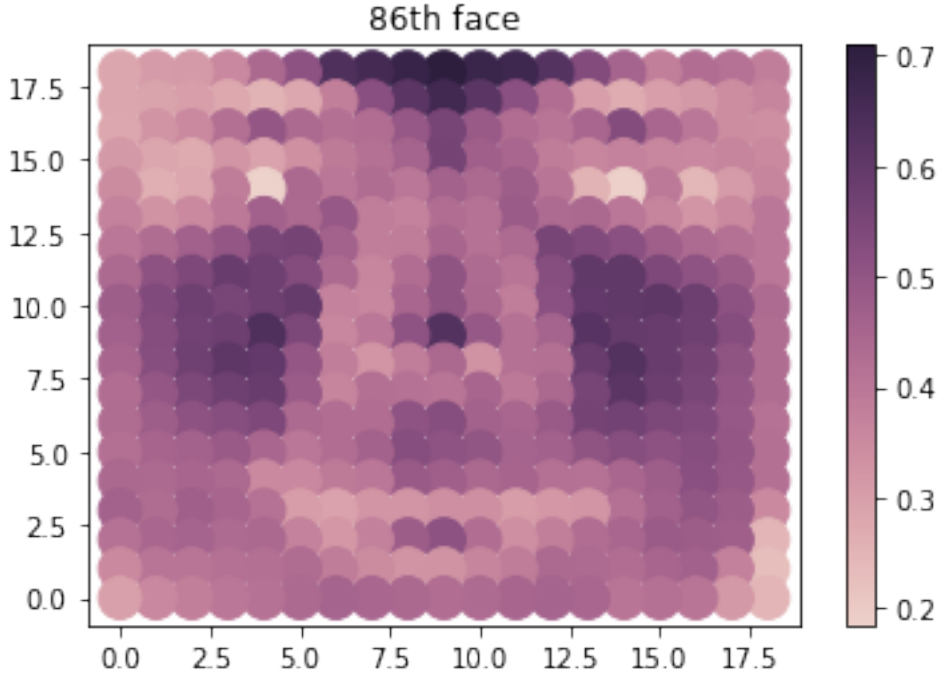

7th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.7
0.6
0.5
0.4
0.3

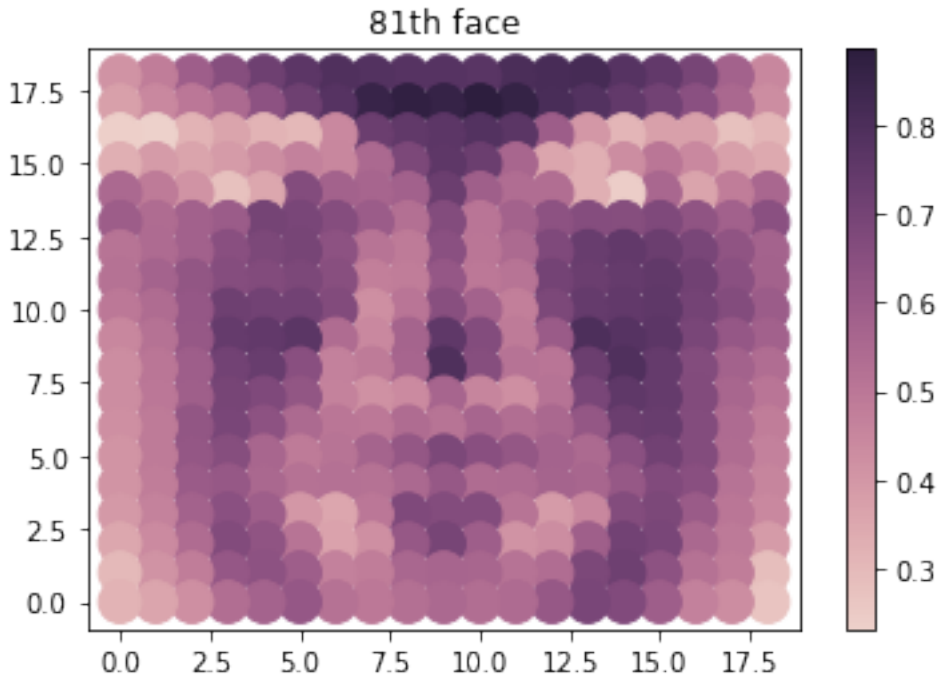

41th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.8
0.7
0.6
0.5
0.4
0.3
0.2

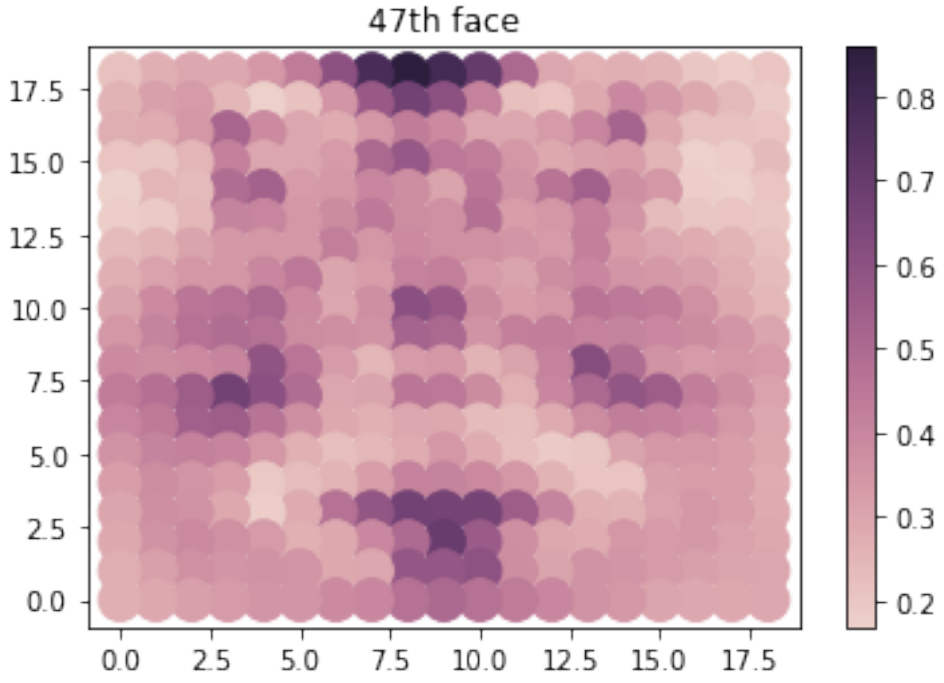

124th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1

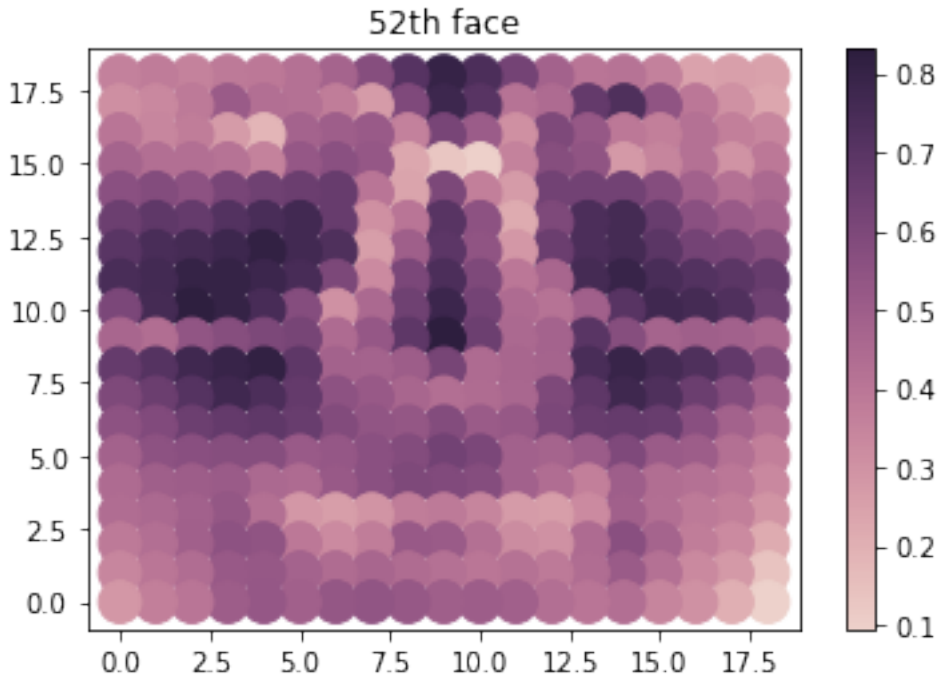

69th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.7
0.6
0.5
0.4
0.3
0.2

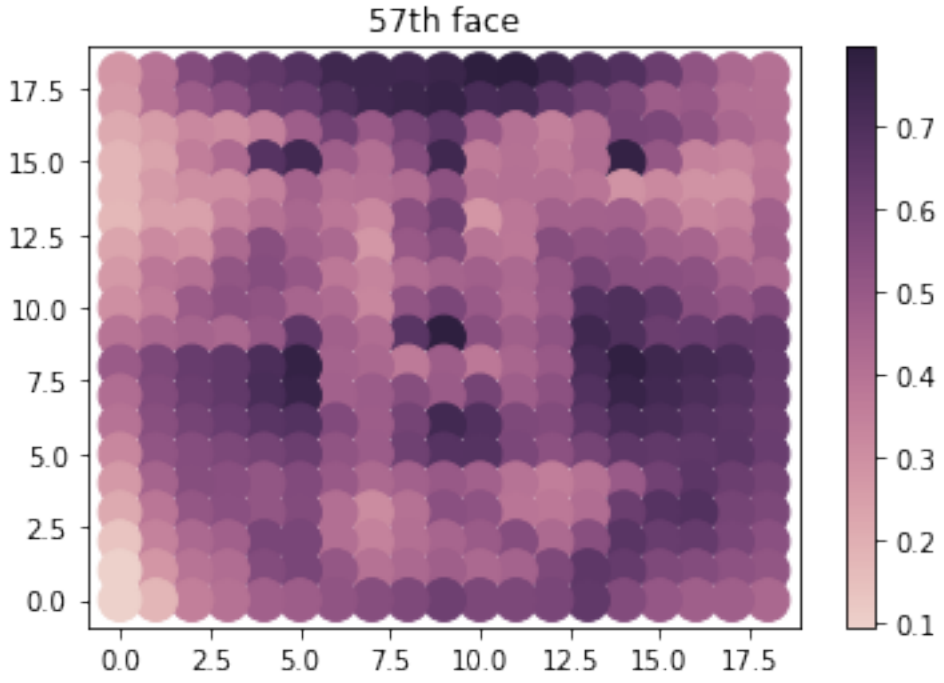

6th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.8
0.7
0.6
0.5
0.4
0.3

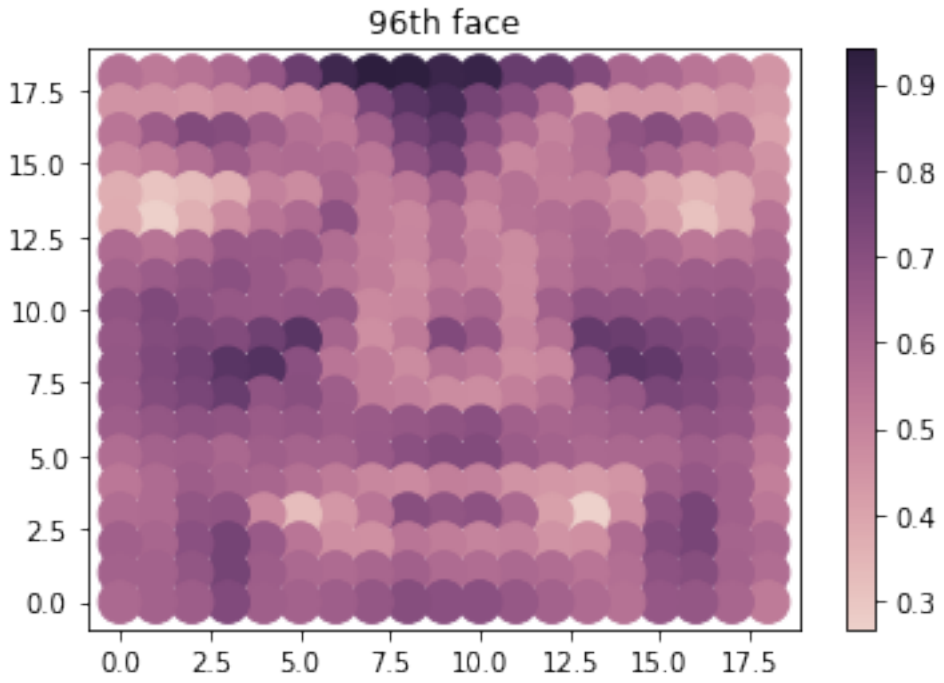

86th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.7
0.6
0.5
0.4
0.3
0.2

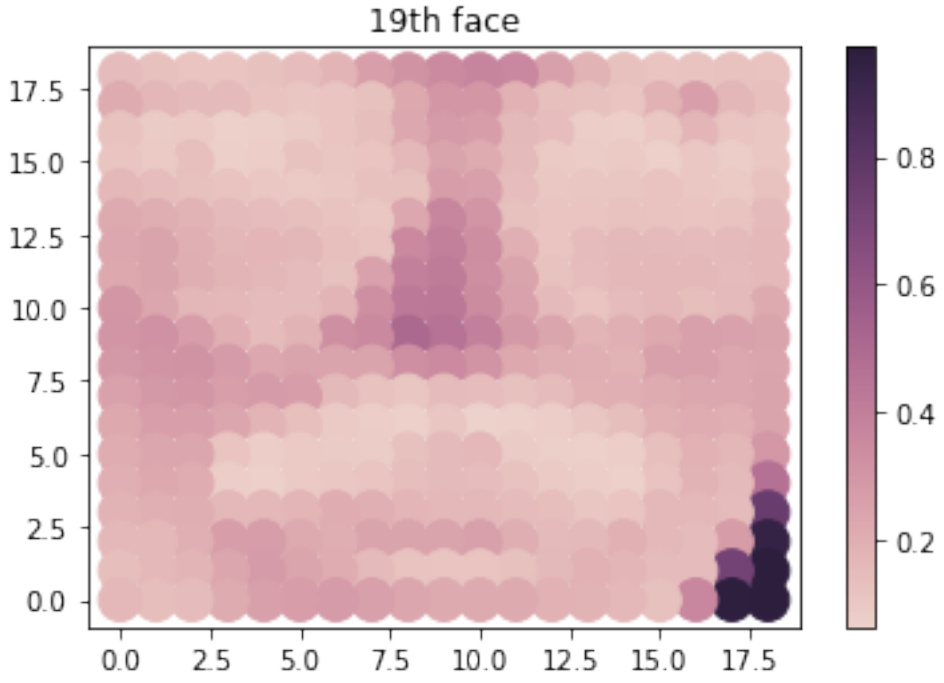

81th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.8
0.7
0.6
0.5
0.4
0.3

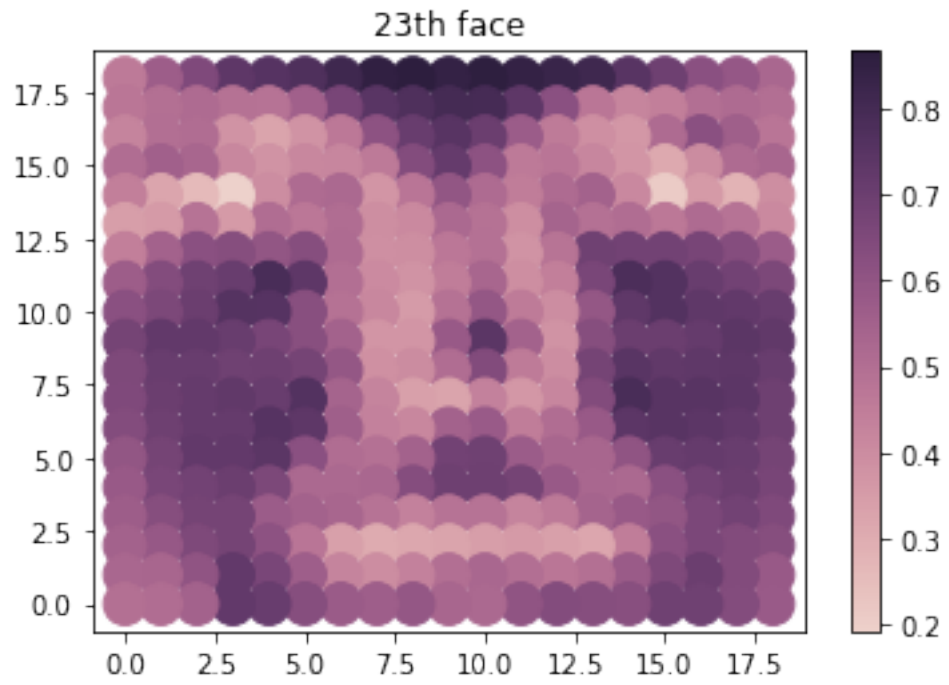

47th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.8
0.7
0.6
0.5
0.4
0.3
0.2


52th 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
0.1


57th 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
0.1

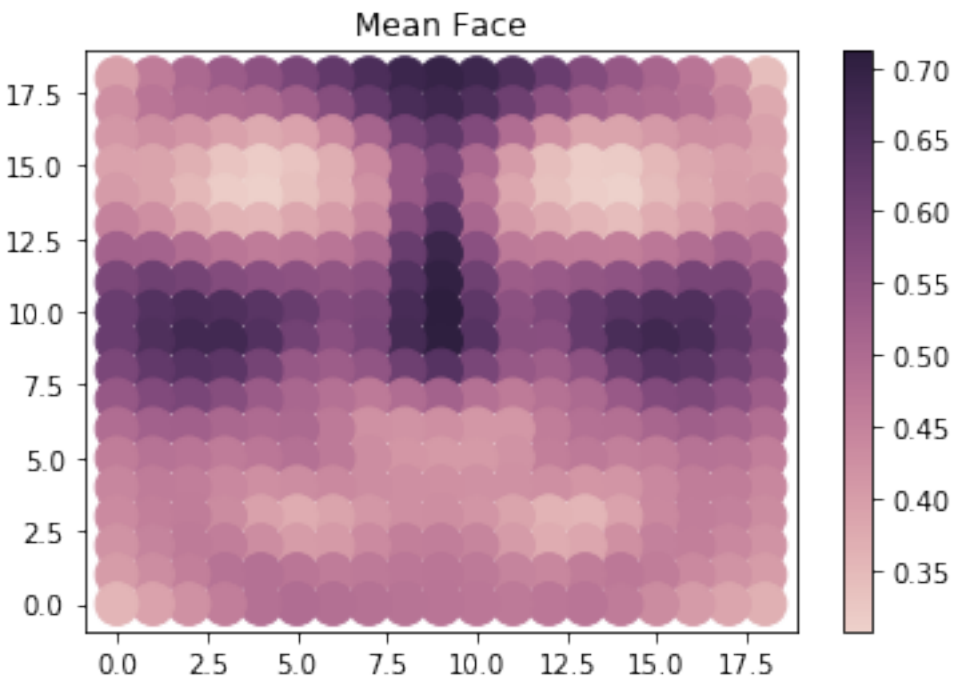

96th face
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.9
0.8
0.7
0.6
0.5
0.4
0.3


19th 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

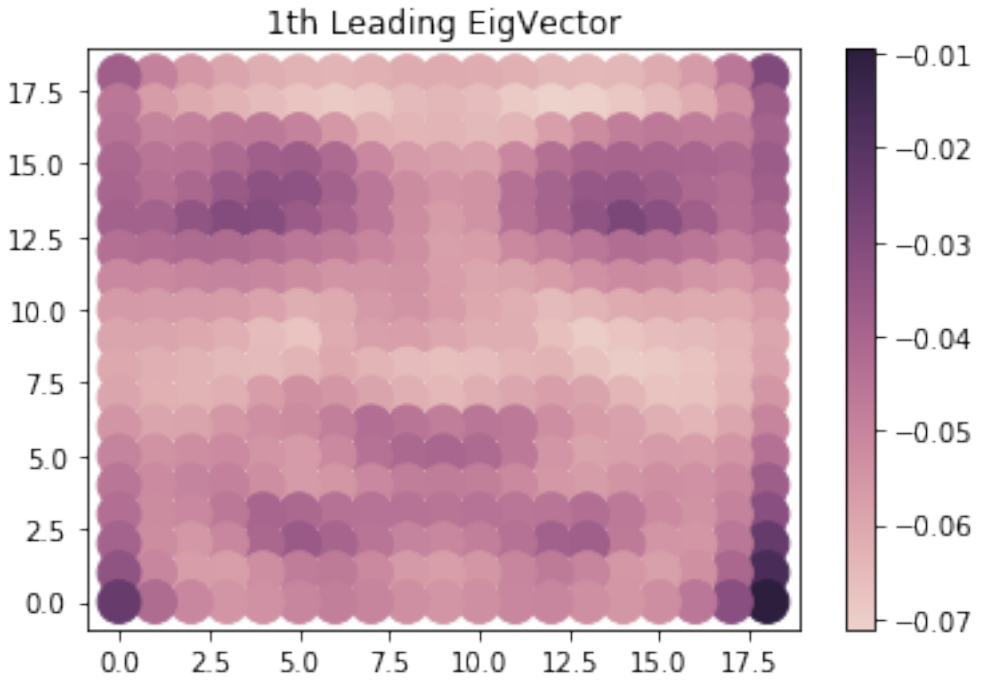


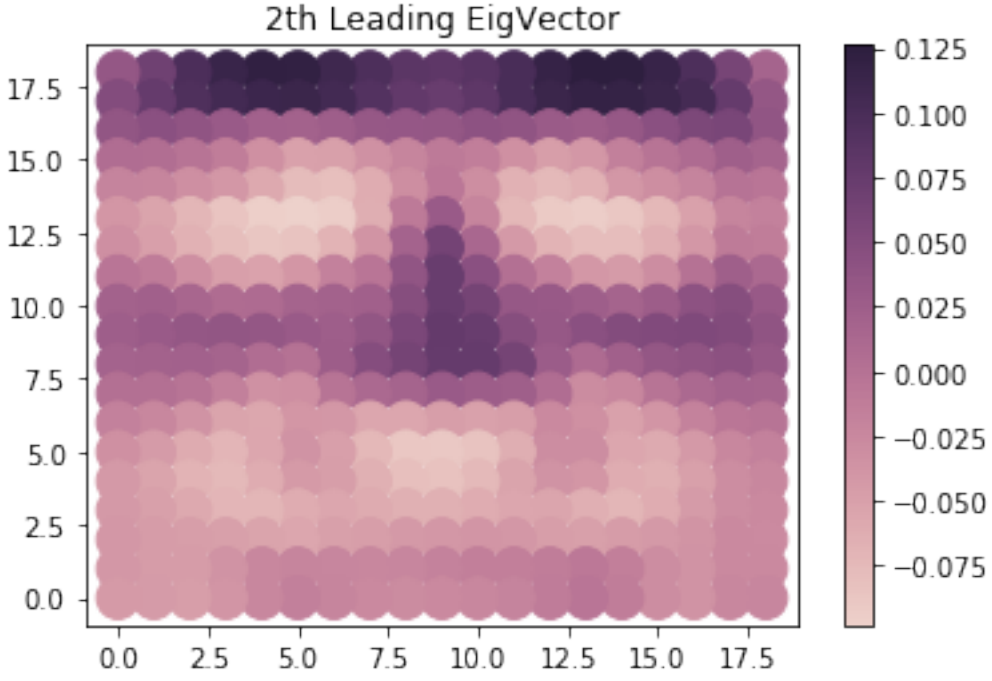
3 3.3

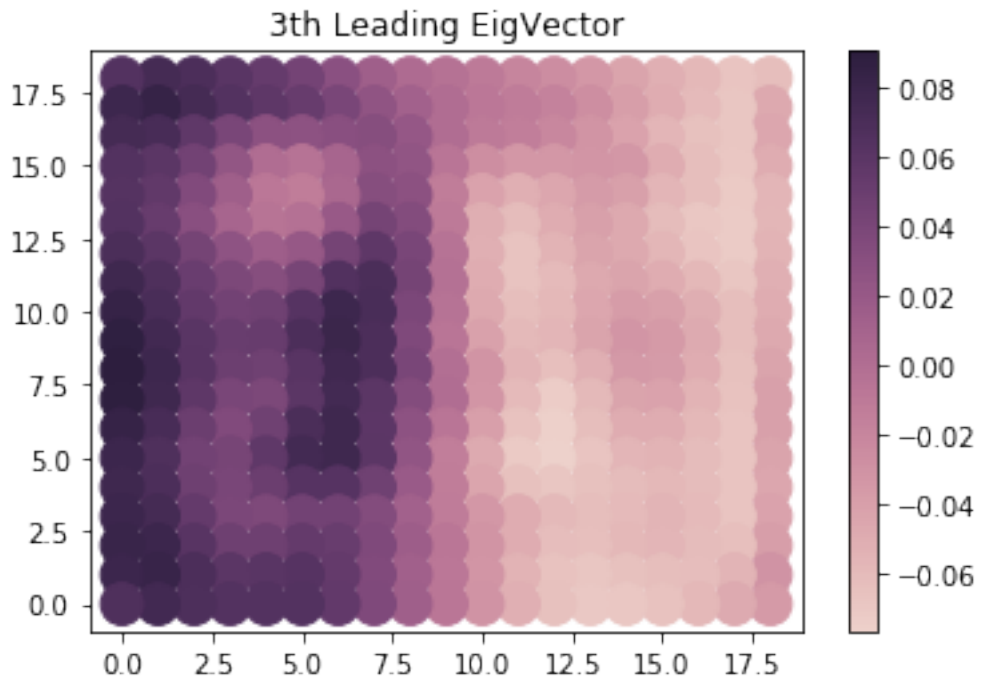
In [127]:
```
W, Z, mu, lambdas = pca(X, 5)

mean_face = mu


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

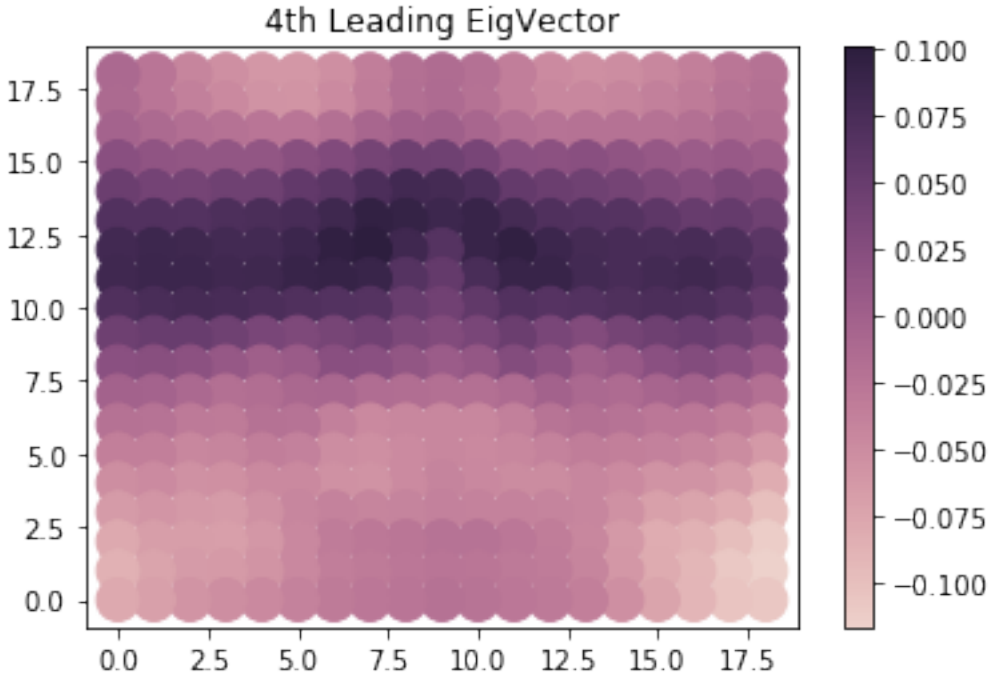


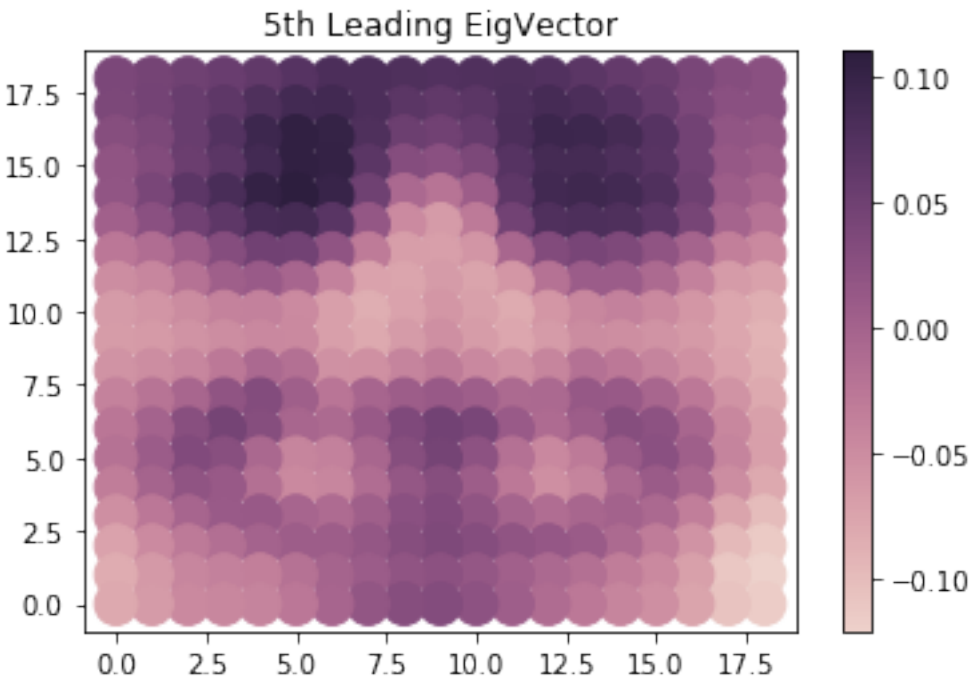
In [128]:
```
lead_5_eig_vecs = W
lead_5_eig_vals = lambdas

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.

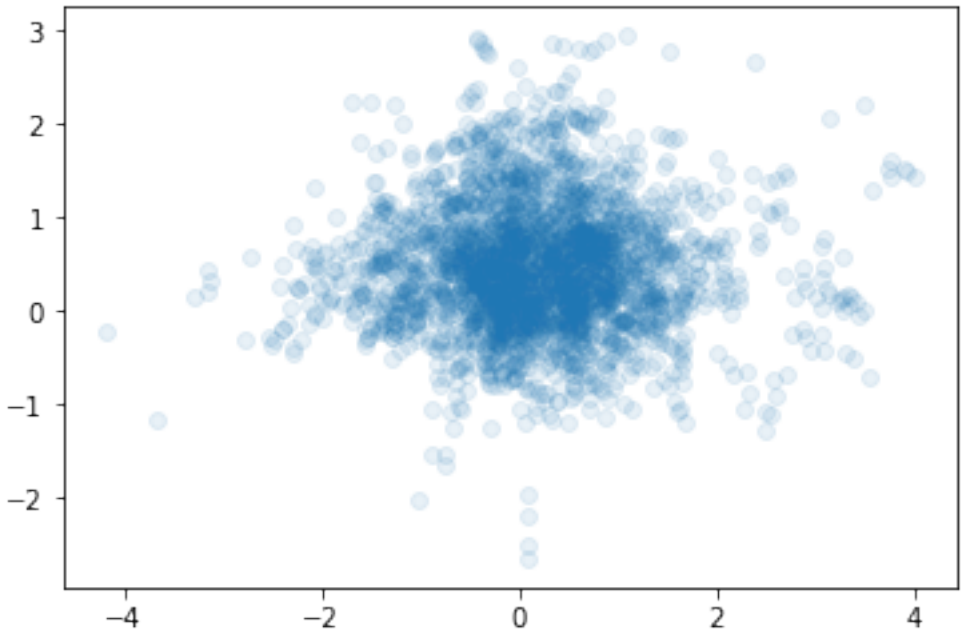
4 3.4

```
In [129]: 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.

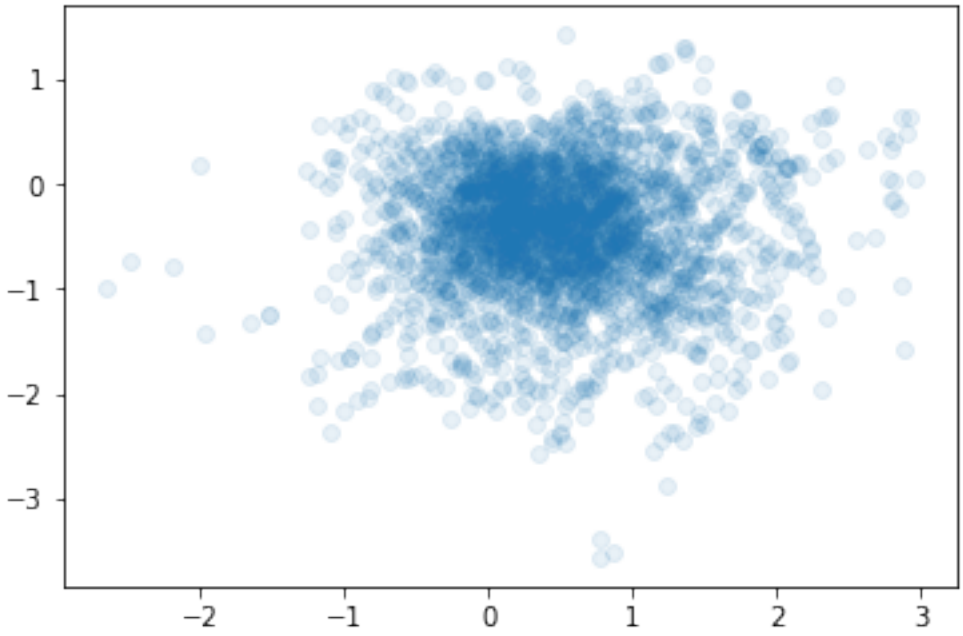
5 3.5

```
In [130]: eig_vecs_2_and_3 = np.transpose(lead_5_eig_vecs)[2:4]

          plot_two_prin_comps(eig_vecs_2_and_3, X)
```



It appears that the second and third principal components have minimal correlation. This is not too surprising, as I would not expect nose and mouth contours, and left to right shading, to have much relationship.

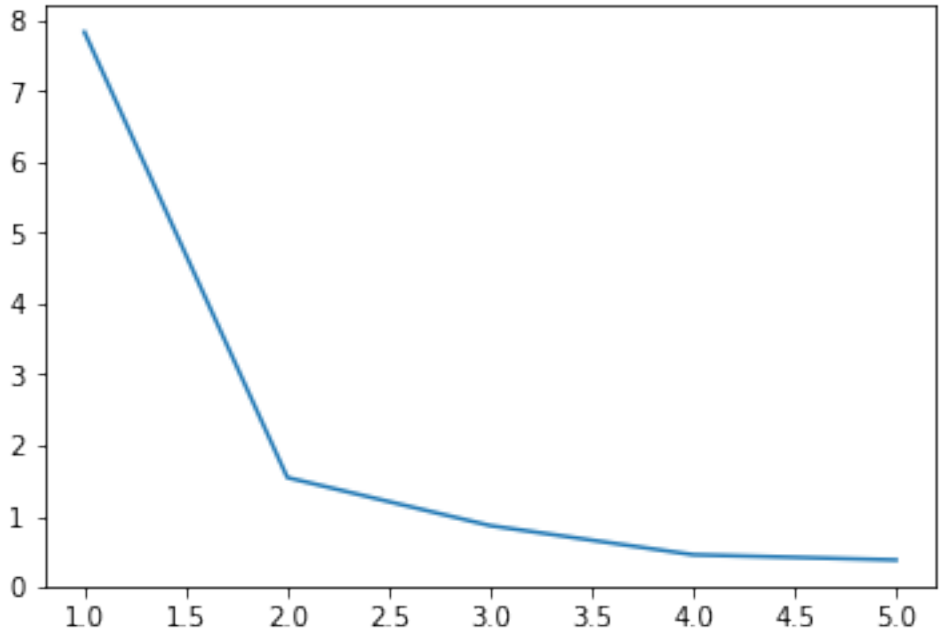
```
In [131]: eig_vecs_4_and_5 = np.transpose(lead_5_eig_vecs)[3:5]


          plot_two_prin_comps(eig_vecs_4_and_5, X)
```



It appears that the principal components corresponding with brow and mouth expressions are not very correlated, interestingly.

6 3.6

```
In [132]: plt.plot(list(range(1 ,6)), lead_5_eig_vals)

Out[132]: [<matplotlib.lines.Line2D at 0x1a468bbb38>]
```



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 [140]: rand_ints_10 = np.array(rand_ints_25[0:9])
          m_enumerated = np.array([1, 2, 5, 10, 25])
          x_approx = np.empty(19*19*rand_ints_10.size*m_enumerated.size)


          k = 0
          for i in m_enumerated:
              for j in rand_ints_10:
                  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_10.size*19*19),
                   'random_faces': np.tile(np.repeat(rand_ints_10, 19*19), m_enumerated.size),
                   'x_coord': np.tile(x_axis_points, rand_ints_10.size*m_enumerated.size),
                   'y_coord': np.tile(y_axis_points, rand_ints_10.size*m_enumerated.size),
                   'faces_approx': np.asarray(x_approx)}

          faces = pand.DataFrame(faces)
```

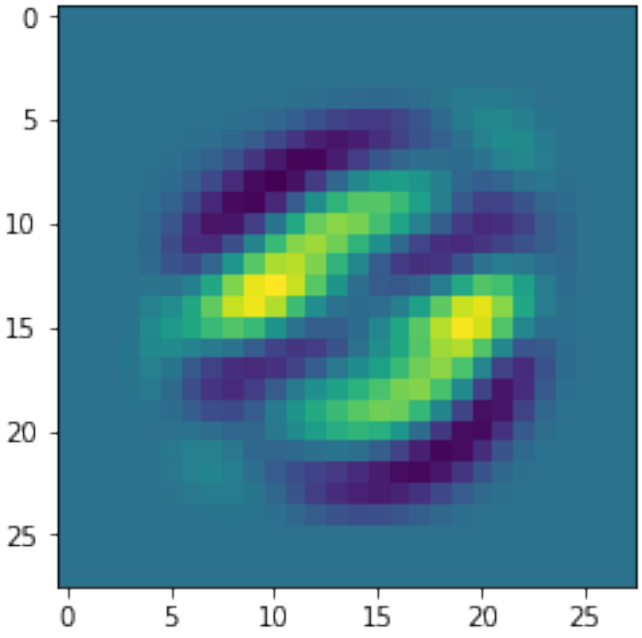
```
In [141]: g = sns.FacetGrid(faces, col = "M", row = "random_faces", hue = "faces_approx", palett
          g.map(plt.scatter, "x_coord", "y_coord", s = 250)

Out[141]: <seaborn.axisgrid.FacetGrid at 0x1a46df0400>
```


random_faces = 17 | M = 1
random_faces = 17 | M = 2
random_faces = 17 | M = 5
random_faces = 17 | M = 10
random_faces = 17 | M = 25
random_faces = 29 | M = 1
random_faces = 29 | M = 2
random_faces = 29 | M = 5
random_faces = 29 | M = 10
random_faces = 29 | M = 25
random_faces = 32 | M = 1
random_faces = 32 | M = 2
random_faces = 32 | M = 5
random_faces = 32 | M = 10
random_faces = 32 | M = 25
random_faces = 62 | M = 1
random_faces = 62 | M = 2
random_faces = 62 | M = 5
random_faces = 62 | M = 10
random_faces = 62 | M = 25
random_faces = 93 | M = 1
random_faces = 93 | M = 2
random_faces = 93 | M = 5
random_faces = 93 | M = 10
random_faces = 93 | M = 25
random_faces = 107 | M = 1
random_faces = 107 | M = 2
random_faces = 107 | M = 5
random_faces = 107 | M = 10
random_faces = 107 | M = 25
random_faces = 131 | M = 1
random_faces = 131 | M = 2
random_faces = 131 | M = 5
random_faces = 131 | M = 10
random_faces = 131 | M = 25
random_faces = 132 | M = 1
random_faces = 132 | M = 2
random_faces = 132 | M = 5
random_faces = 132 | M = 10
random_faces = 132 | M = 25
random_faces = 138 | M = 1
random_faces = 138 | M = 2
random_faces = 138 | M = 5
random_faces = 138 | M = 10
random_faces = 138 | M = 25
y_coord
x_coord
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0
5
10
15
20

8 3.8

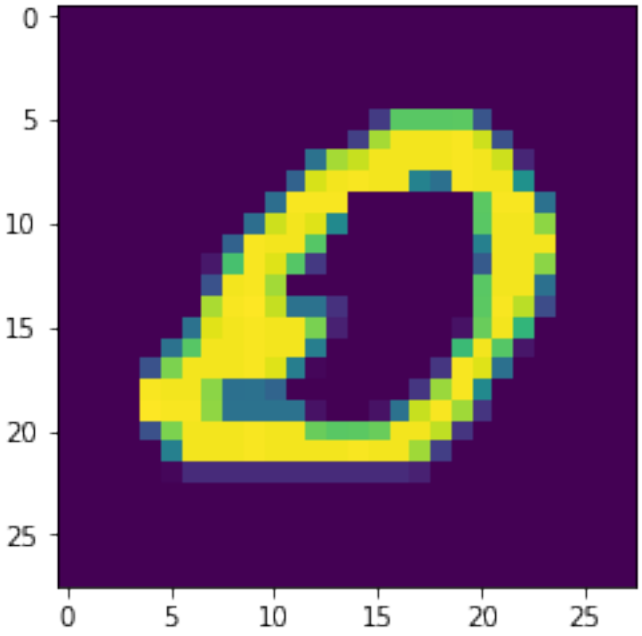
In [102]:
```
mat = scipy.loadmat('/Users/wyattmadden/Documents/school/' +
                    'MSU/2020/spring/m508/lab_info/lab_3/mnist.mat')
X = mat["X"]

def faceplot(X, i):
    face = [row[i] for row in X]
    face_mat = np.reshape(face, (28,28))
    fig = plt.imshow(face_mat)
    return(fig)

pca_mnist = pca(X, 1)
```

Out[102]: <matplotlib.image.AxesImage at 0x1a2fda6198>



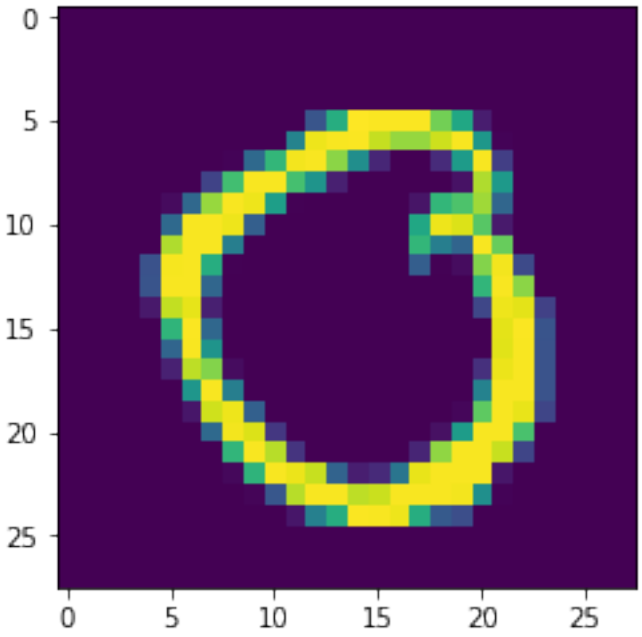
In [103]:
```
faceplot(X, 1)
```

Out[103]: <matplotlib.image.AxesImage at 0x1a264dd390>



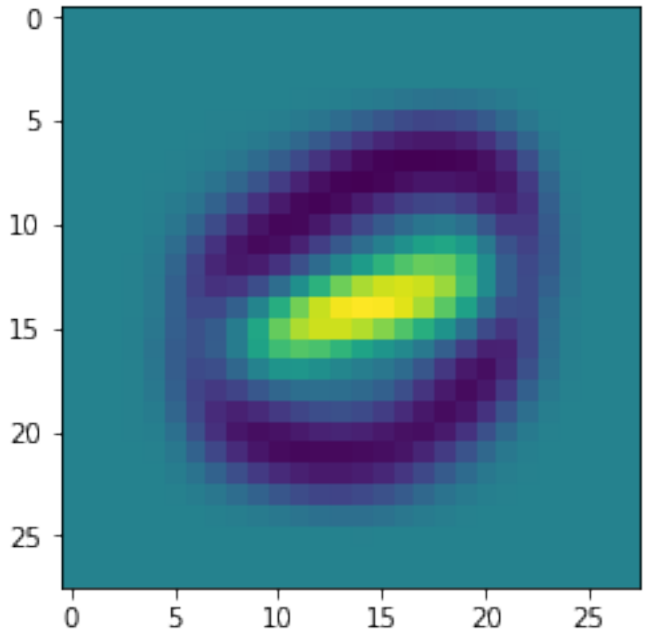

```
In [104]: faceplot(X, 100)
```

```
Out[104]: <matplotlib.image.AxesImage at 0x1a26546978>
```



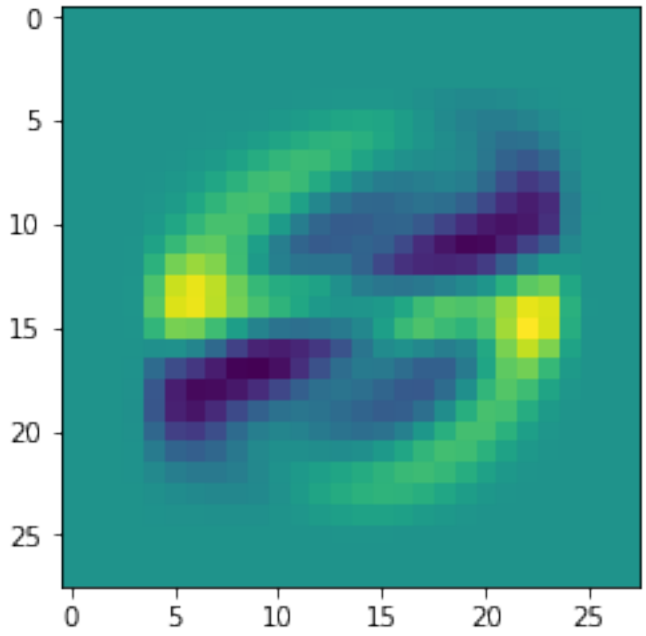
```
In [111]: pca_mnist = pca(X, 3)
          faceplot(pca_mnist[0][0:,], 0)
```

```
Out[111]: <matplotlib.image.AxesImage at 0x1a1d87b978>
```

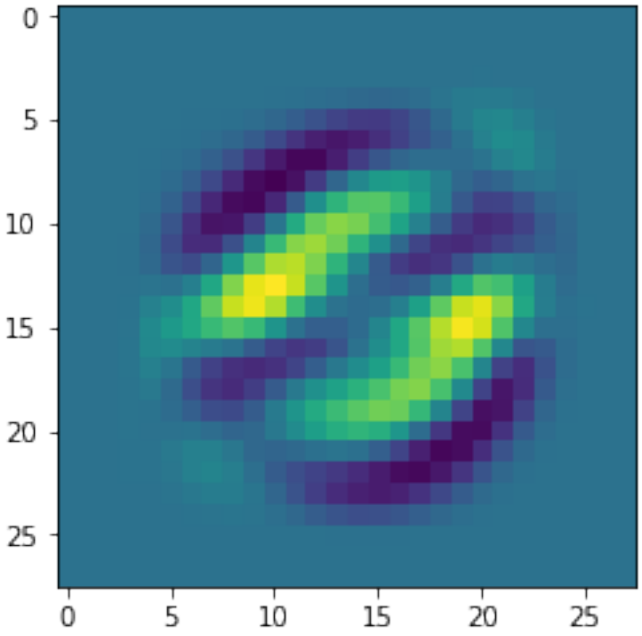



```
In [110]: faceplot(pca_mnist[0][0:,], 1)
```

```
Out[110]: <matplotlib.image.AxesImage at 0x1a2fd55898>
```



```
In [112]: faceplot(pca_mnist[0][0:,], 2)

Out[112]: <matplotlib.image.AxesImage at 0x1a1d9e47f0>
```

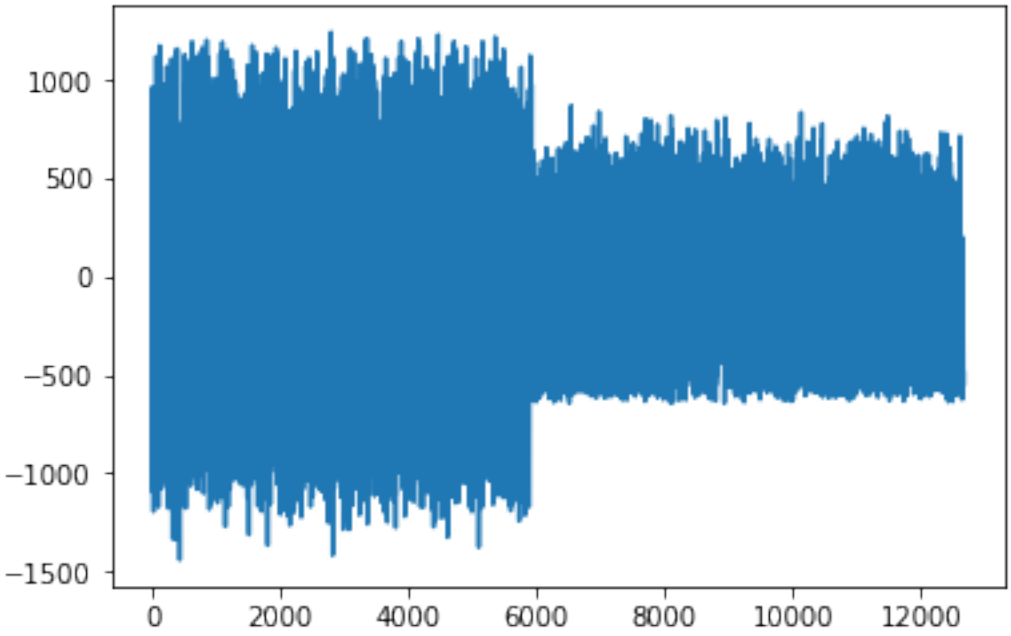


It does not appear like the letters are coming through very clearly in the principal components.

9 4

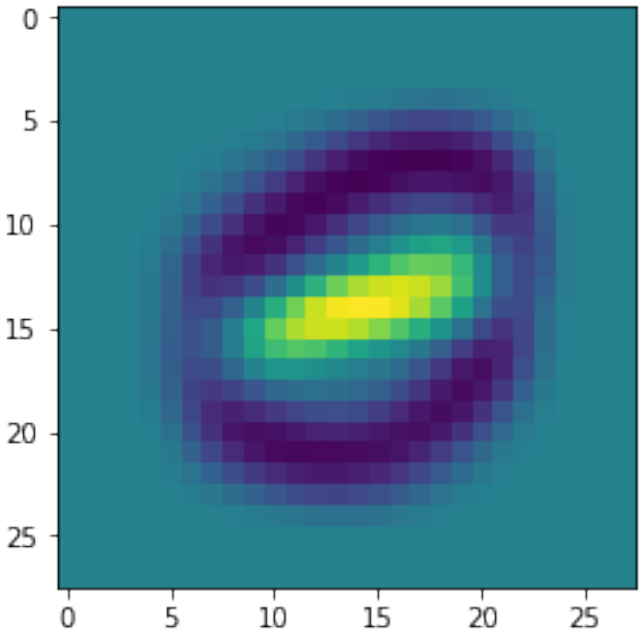
Visualizing the 3rd principle component it looks like a circle. The latent variable, z, should have high scores for the 0's in the dataset and low scores for the 1s. Look at pca[1], the z scores, then grab the 3rd part of the array, then just plot it.

```
In [114]: plt.plot(pca_mnist[1][2])

Out[114]: [<matplotlib.lines.Line2D at 0x1a1db79f98>]
```



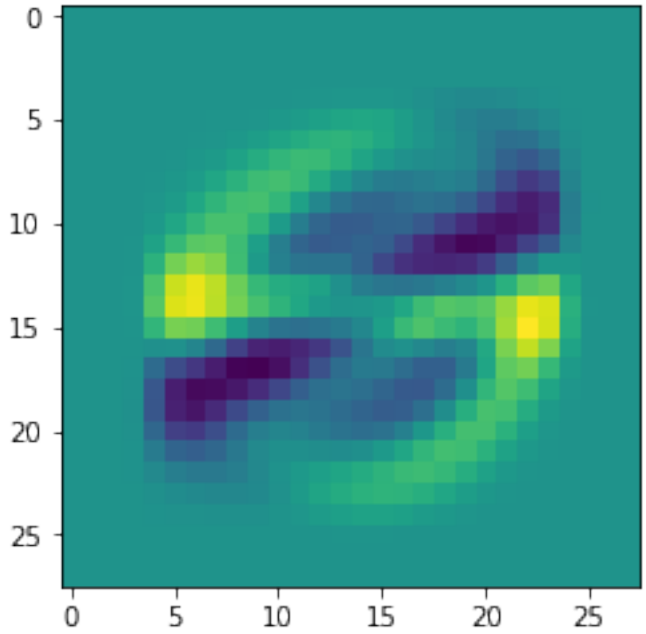

```
In [116]: pca_mnist = pca(X, 5)
          faceplot(pca_mnist[0][0:,], 0)
```

```
Out[116]: <matplotlib.image.AxesImage at 0x1a1dcafda0>
```



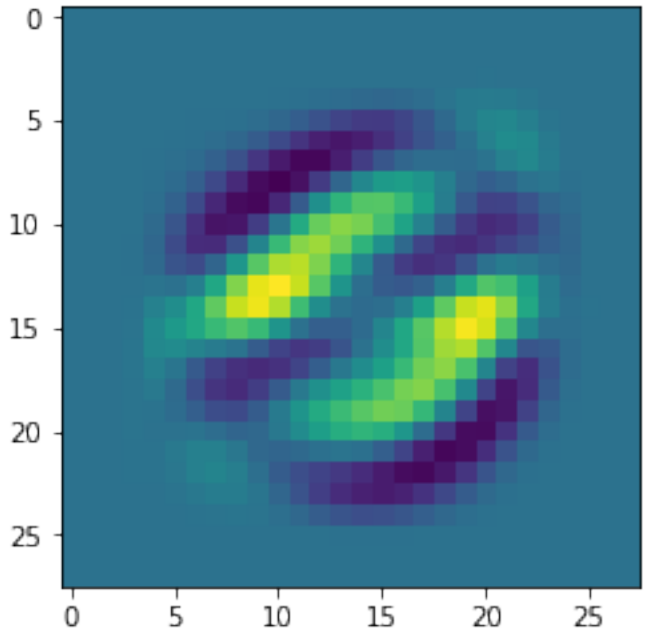
```
In [117]: faceplot(pca_mnist[0][0:,], 1)
```

```
Out[117]: <matplotlib.image.AxesImage at 0x1a2f235d30>
```



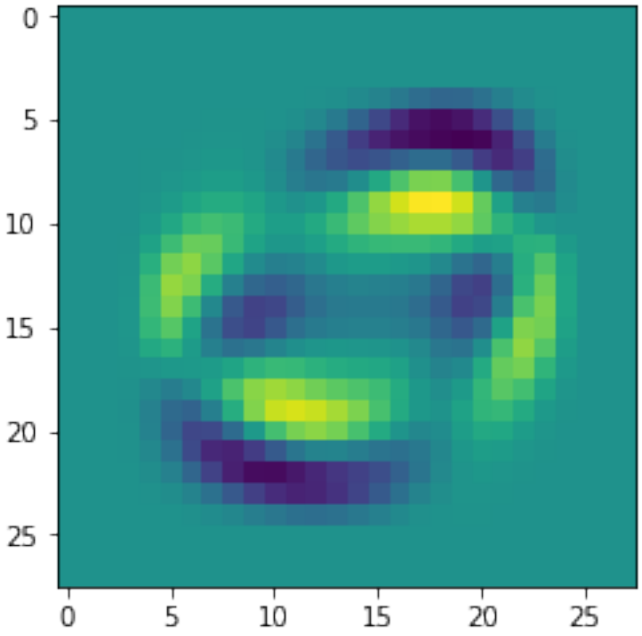

```
In [118]: faceplot(pca_mnist[0][0:,], 2)
```

```
Out[118]: <matplotlib.image.AxesImage at 0x1a2f2a8e10>
```



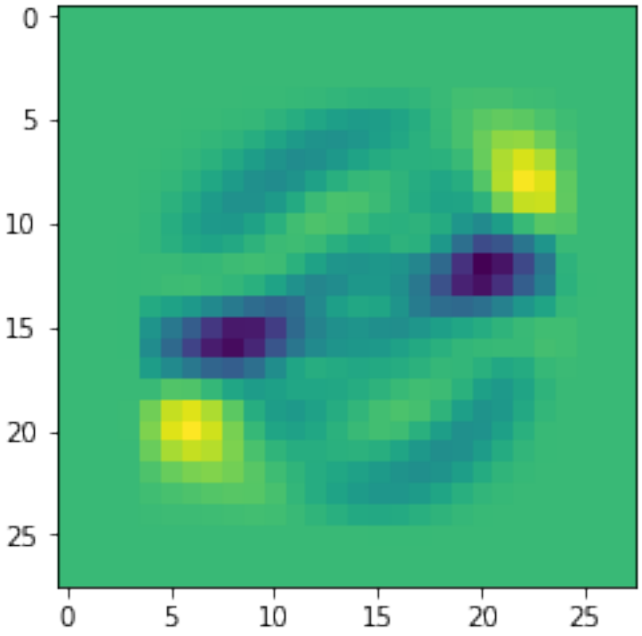
```
In [119]: faceplot(pca_mnist[0][0:,], 3)
```

```
Out[119]: <matplotlib.image.AxesImage at 0x1a2fdcdcc0>
```




```
In [120]: faceplot(pca_mnist[0][0:,], 4)
```

```
Out[120]: <matplotlib.image.AxesImage at 0x1a30123c50>
```


0
5
10
15
20
25
0
5
10
15
20
25