Table of Contents


```
clc; clear;
```

Part 1.1

Complete and turn in the function princomp.m. Do not use the built-in function for PCA (but do use, e.g., eigs or svd). Notes: This function removes the mean from the data—this makes computing the covariance matrix easier. princomp.m should then simply return the leading eigenvectors in W (and associated eigenvalues λ) of said covariance matrix. You can also easily project the zero-mean data onto these to get lower dimensional representations (the latent variables zi for each xi). To later reconstruct data (approximations) from latent data, use zi and principal components; do not forget to add the mean back, at this time! **

I implemented PCA algorithm using eigs in princomp.m. To test that it works we compare to the built in function on the hald.ingredients dataset. The first two columns show my output, and the 2nd two columns show the output of the built in PCA function.

```
load hald
M = 2;
X = ingredients;
[W, Z, mu, lambda] = princomp(X,M);
Q = pca(X');
high_dim_case = [W Q(:, 1:M)]

X = X';
[W, Z, mu, lambda] = princomp(X,M);
Q = pca(X');
low_dim_case = [W Q(:, 1:M)]
clear;  % remove the dummy variables from the example
```

```
high_dim_case =

    0.1289    0.5651    0.1289    0.5651
```

```
    0.1453    0.4630    0.1453    0.4630
    0.3097   -0.0673    0.3097   -0.0673
    0.1418    0.3611    0.1418    0.3611
    0.3035    0.1325    0.3035    0.1325
    0.3000   -0.0418    0.3000   -0.0418
    0.4139   -0.2805    0.4139   -0.2805
    0.1384    0.3390    0.1384    0.3390
    0.3007   -0.0152    0.3007   -0.0152
    0.2236    0.0005    0.2236    0.0005
    0.1959    0.1787    0.1959    0.1787
    0.3738   -0.2076    0.3738   -0.2076
    0.3939   -0.2079    0.3939   -0.2079


low_dim_case =

   -0.0678    0.6460   -0.0678   -0.6460
   -0.6785    0.0200   -0.6785   -0.0200
    0.0290   -0.7553    0.0290    0.7553
    0.7309    0.1085    0.7309   -0.1085
```

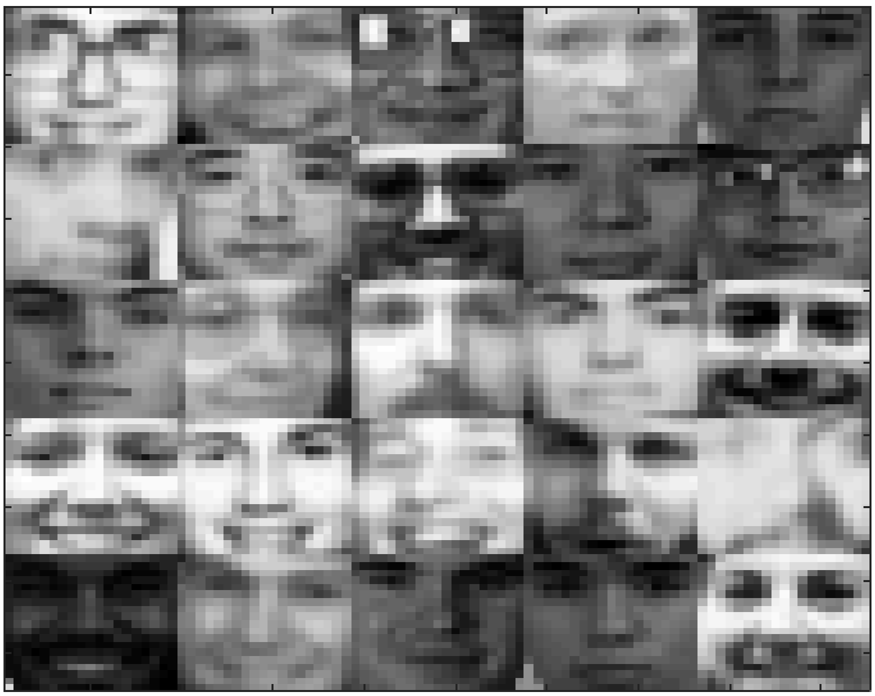
Part 1.2

Load and look at the CBCL faces data set to get a feel for it, for example by using the provided function imgrid.m with the raw data (e.g., visualize an array of 25 randomly selected faces).

```
cbcl = load('cbcl.mat');
faces = randperm(2900, 25)
display_data = cbcl.X(:, faces);

figure()
imgrid(display_data, cbcl.dims, [5, 5]);
title("25 Random Faces from CBCL");
```

```
faces =

  Columns 1 through 6

         990        1761         556        2140         704
 2656

  Columns 7 through 12

         779        2215         546         832         264
 1665

  Columns 13 through 18

        1974        1579        1229        1860        1868
 1958
```

```
  Columns 19 through 24

        1833        2724         602         2043         680
344

  Column 25

        1747
```



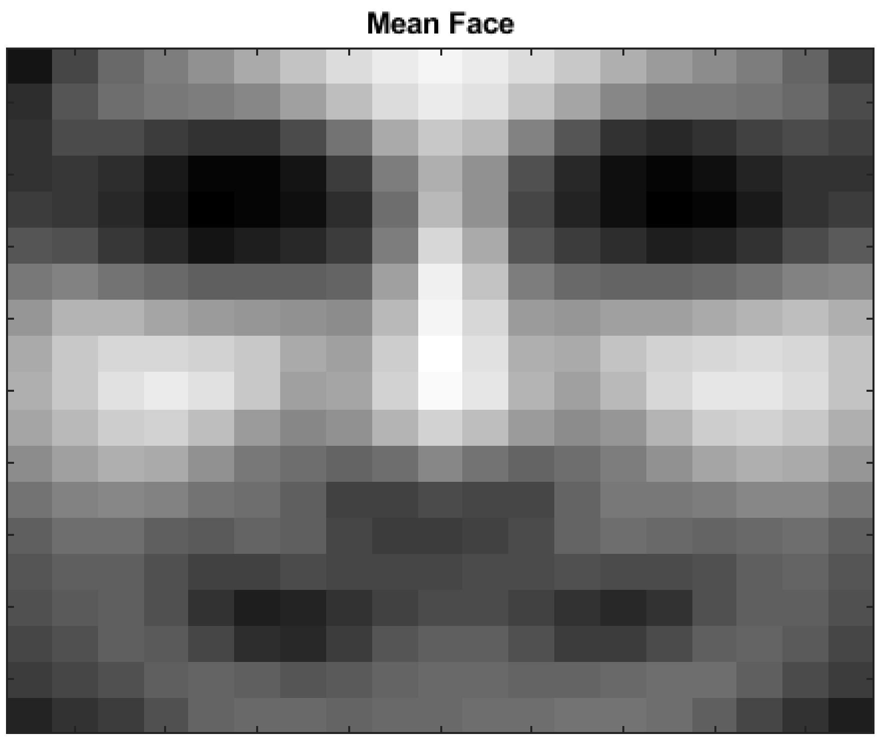
Part 1.3

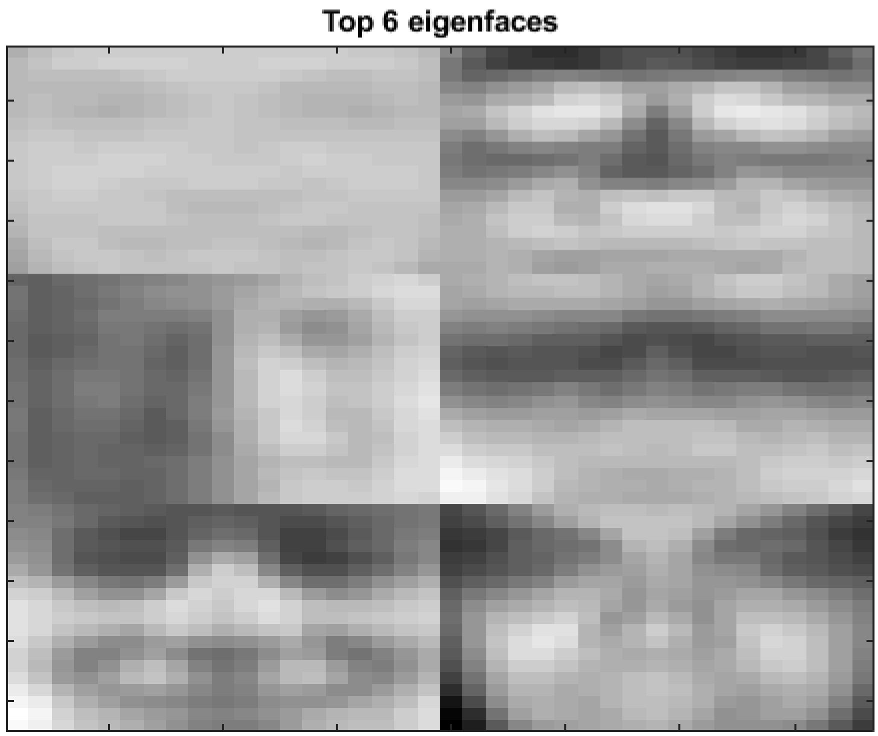
Run PCA on the CBCL faces data set and turn in a figure with the mean face and the first five principal components (leading eigenfaces). Explain why you think they look the way they do, and what sort of information in the data they are modeling. Note: for higher components this becomes more difficult. Explain as much as you can
****

I notice that the mean face has very strong forehead, nose, and cheeks. Its eyes are very vague and black and also has a very dark mouth. This is probably because there is a lot of variation in these areas while the nose, forehead and cheeks are all typically quite well lit. The eigenfaces are a bit harder to interpret. The first one (top left) is just a broad outline of a face, while the 2nd one (top right) is the eyes and mouth. I have no idea what the third one is. The 4th one is just the eyes. And the fith looks pretty normal, I would say it is more on the mouth structure. Then the 6th we can see some eyebrows and the full face is not in the frame, so perhaps capturing more facial hair or people with small heads.

```
X = cbcl.X;
```

```
M = 6;
[W, Z, mu, lambda] = princomp(X,M);
figure()
imgrid(mu, cbcl.dims, [1, 1]);  % Mean face (kind of insulting)
title("Mean Face");
figure()
imgrid(W, cbcl.dims, [3, 2]);  % 5 eigenfaces
title("Top 6 eigenfaces");
```

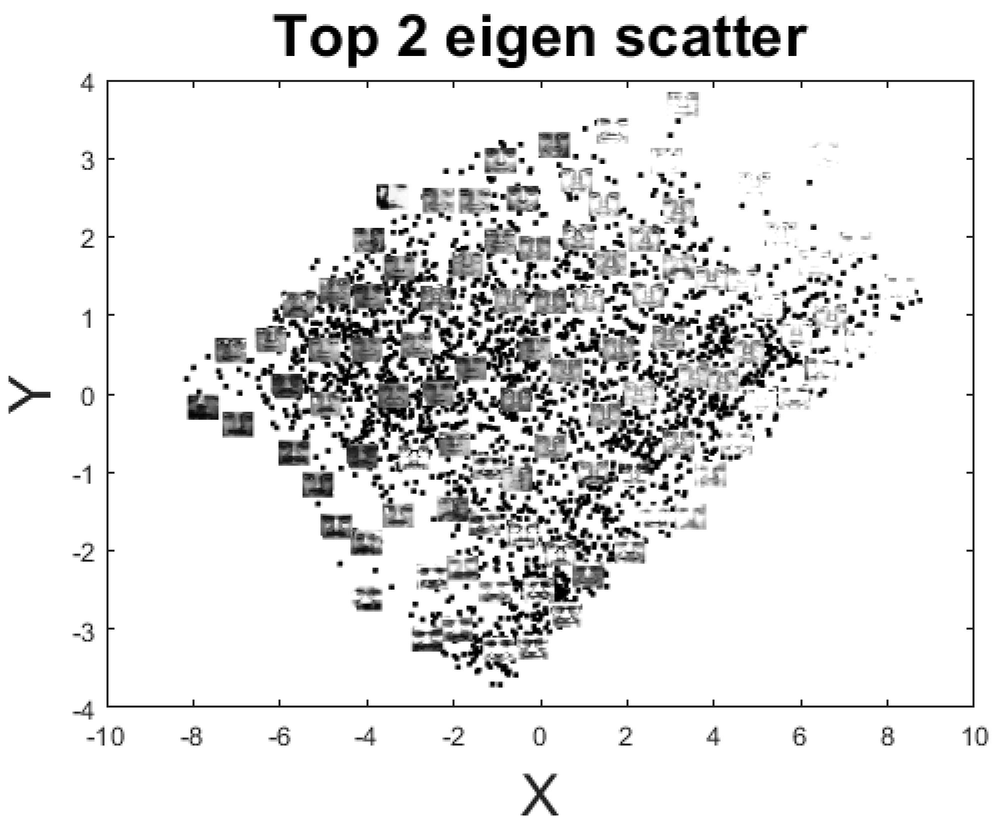




Part 1.4

Project the faces onto the first two principal components and plot/turn in the result as a point cloud in 2D (the best representation of the data in two dimensions), using either scatter or imcloud.m.
****
I notice that from left to right the pictures get brighter and darker (from the first principal component), then in the second component (vertical) we see more variation in the brightness of the eyes and mouth.

```
M = 2;
[W, Z, mu, lambda] = princomp(X,M);
figure()
imcloud(Z, X, cbcl.dims)
title("Top 2 eigen scatter");
```

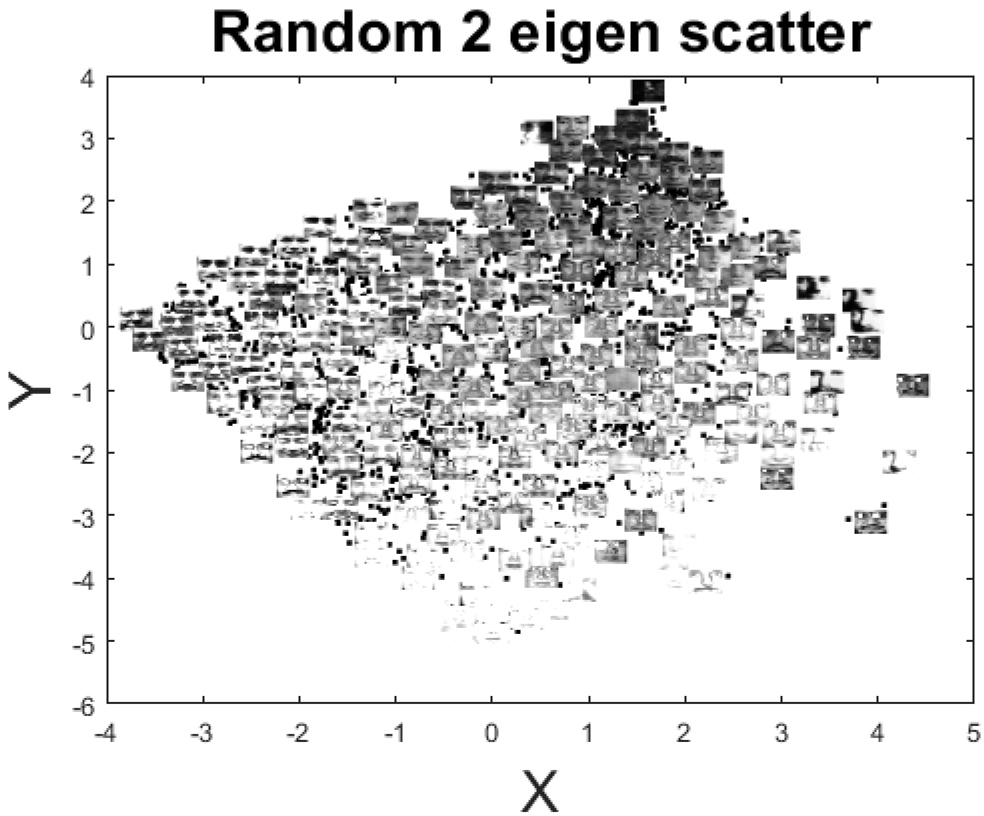


Part 1.5

Use the imcloud.m function to create this plot for projections onto other significant PCs, and explain the results you see as they compare to your response from looking at the PC vectors. How might you change or improve your description of the information captured by the principal components?

```
M = 6;
[W, Z, mu, lambda] = princomp(X,M);
comps = randperm(6, 2)
W2 = W(:, comps);
P = W2' \ (W2' * W2);  % projection matrix formula
Z2 = P' * (X - mu);
figure()
imcloud(Z2, X, cbcl.dims)
title("Random 2 eigen scatter")
```

```
comps =

     2     5
```



Part 1.6

Plot the spectrum of your PCA: trace the eigenvalues of the covariance matrix in descending order. Does this maybe tell you how many principal components should be selected for "optimal" dimensionality reduction?
****

We see that the eigenvalues steadily decrease. I don't see a super major drop-off in the eigenvalues. If there was, it would be best to take all the values before the drop-off for good dimensionality reduction.

```
lambda
```

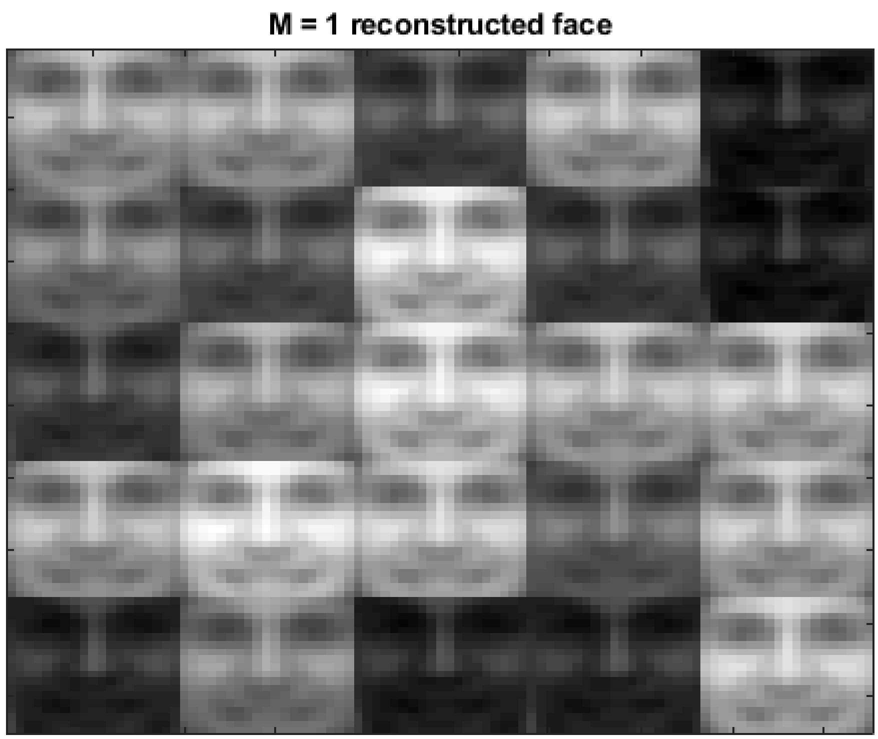
```
lambda =

   1.0e+04 *

    2.2697
    0.4478
    0.2512
    0.1322
    0.1107
    0.0978
```

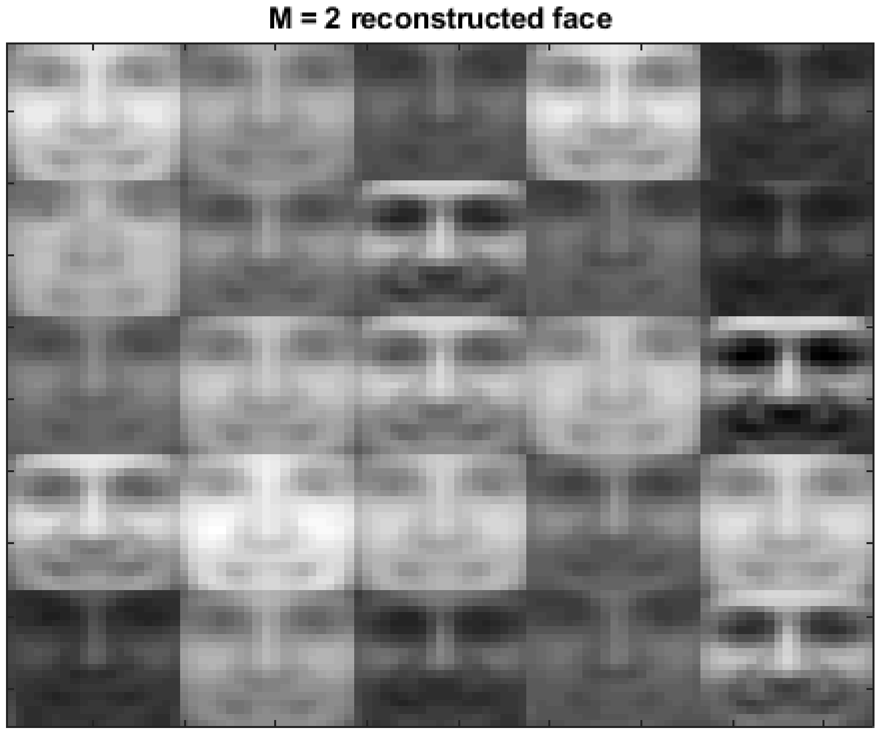
Part 1.7

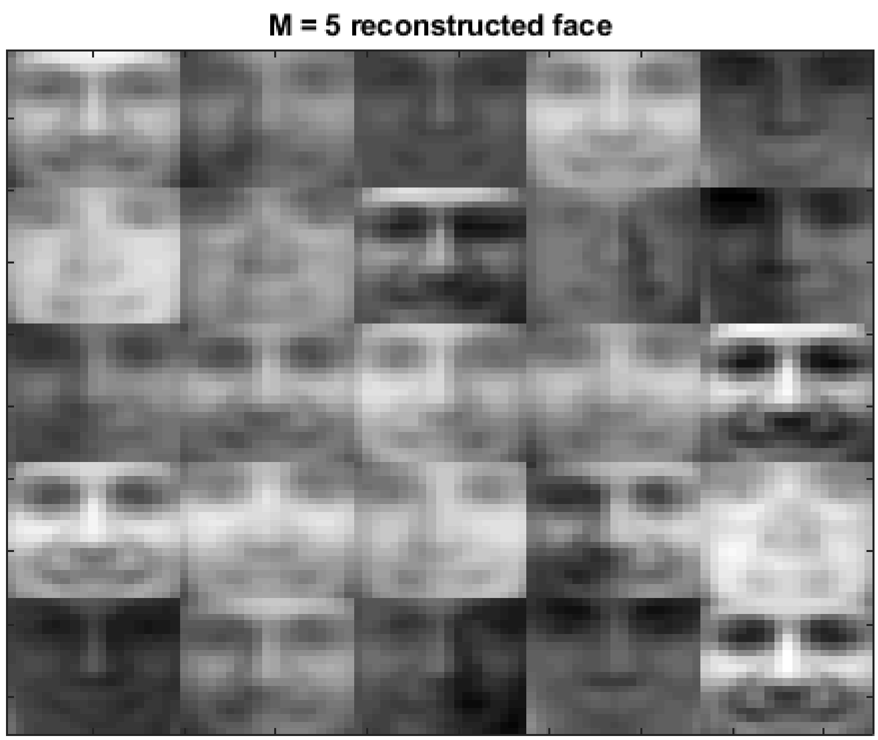
Reconstruct faces from varying number of principal components. For example, try M = 1,2,5,10,25, and compare the reconstructions against original faces (you can chose the same 25 faces as under item 1, for example, and use imgrid.m with the reconstructed data). What do you observe?
****

It does a pretty good job of reconstructing the faces. At M=25 they are quite recognizable as the same face. M=1 is jsut lighter and darker versions of the same (mean) face. But as M increases, more defining characteristics become visible. It is interesting how some (presumably) more rare characteristics, like glasses are still not present at M = 25.

```
for m = [1, 2, 5, 10, 25]
    [W, Z, mu, lambda] = princomp(X,m);
    Re = W * Z + mu;

    display_data = Re(:, faces);
    figure()
    imgrid(display_data, cbcl.dims, [5, 5]);
    title("M = " + m + " reconstructed face");
end
```




M = 2 reconstructed face


M = 5 reconstructed face

M = 10 reconstructed face

M = 25 reconstructed face

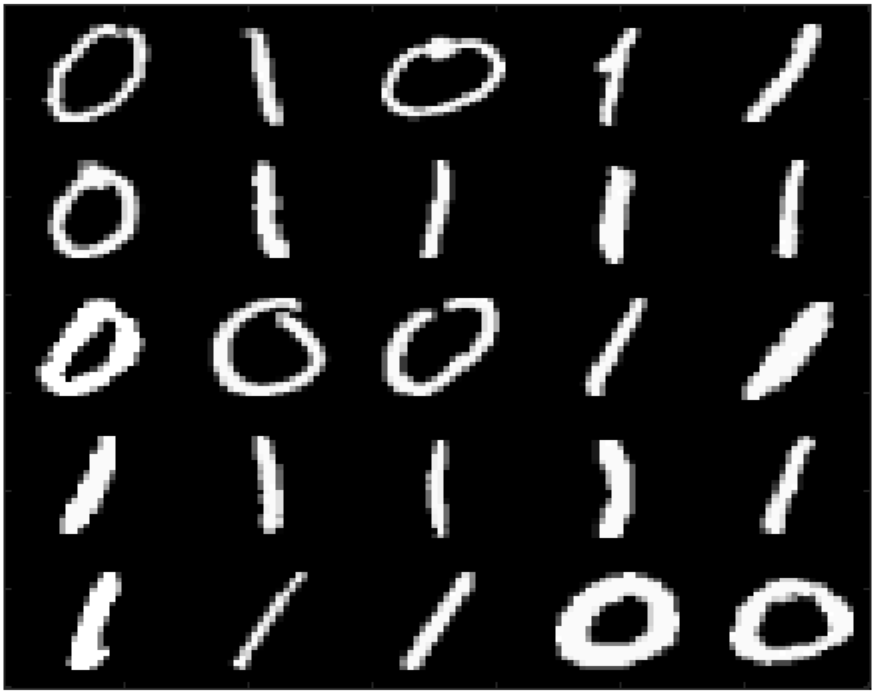
Part 2.2

Load and look at the MNIST numbers data set to get a feel for it, for example by using the provided function imgrid.m with the raw data (e.g., visualize an array of 25 randomly selected numbers).

```
mnist = load("mnist.mat");
faces = randperm(12665, 25)
display_data = mnist.X(:, faces);

figure()
imgrid(display_data, mnist.dims, [5, 5]);
title("25 Random Images from MNIST");
```

```
faces =

  Columns 1 through 6

        4436         8384         5270        10661        10546
 3247

  Columns 7 through 12

        7766         7371         6845        11010         3351
 4025

  Columns 13 through 18

        1509        11891         8167         6066         8087
 6890

  Columns 19 through 24

        8187         6878         9118         6607        12564
 2765

  Column 25

        1338
```

25 Random Images from MNIST



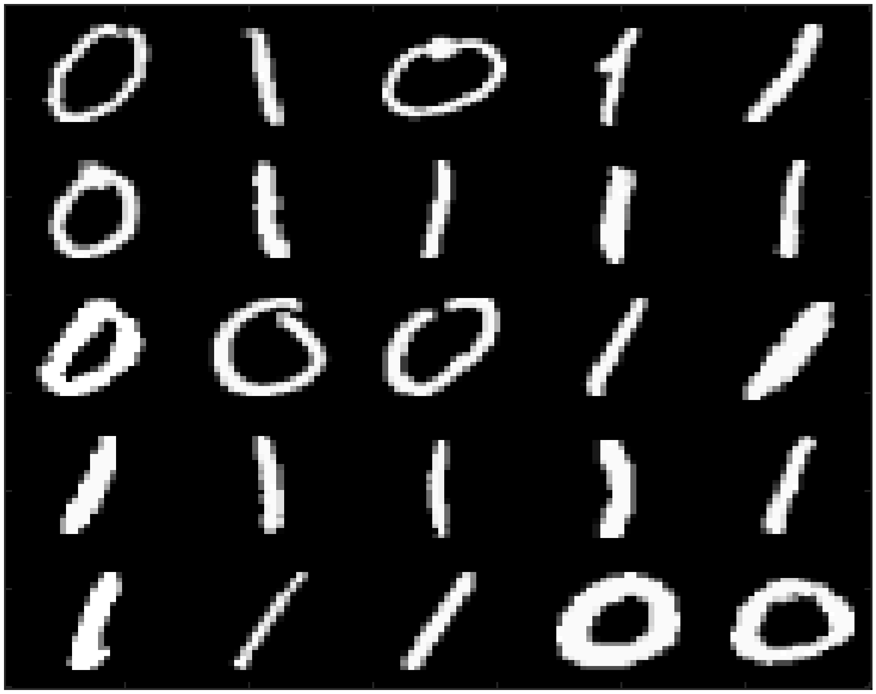
Part 2.3

Run PCA on the MNIST numbers data set and turn in a figure with the mean number and the first five principal components (leading eigennumbers). Explain why you think they look the way they do, and what sort of information in the data they are modeling. Note: for higher components this becomes more difficult. Explain as much as you can
****

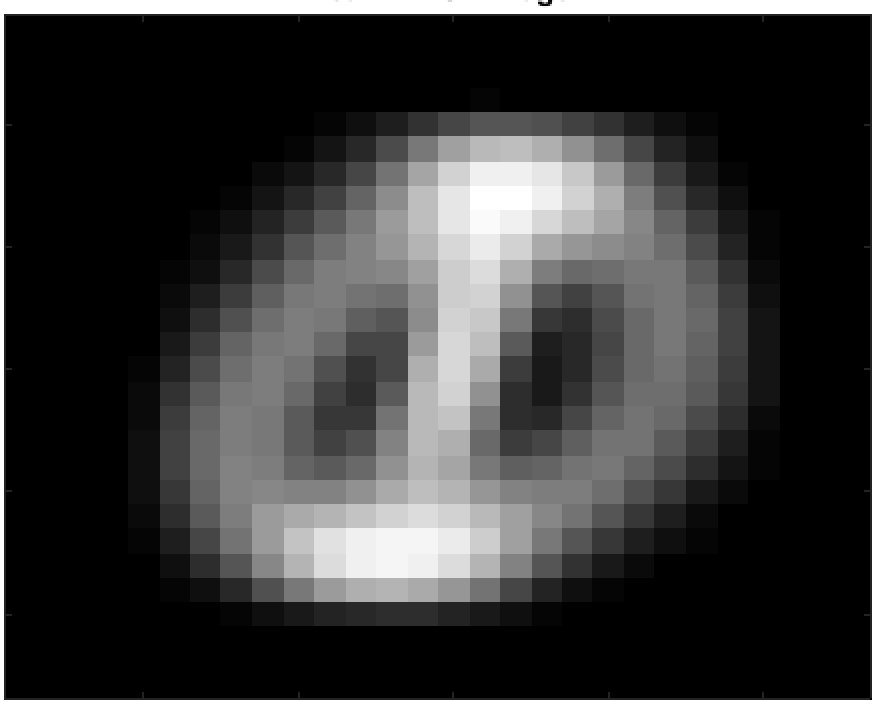
The mean image looks like a 1 with a 0 around it. This makes a ton of sense as this is what is in the dataset, just 1 and 0. So the mean would be the average of these, which is a circle with a line through it. The first two eigenvalues are pretty distinctive. The first one is mostly a zero, while the second one is a one.

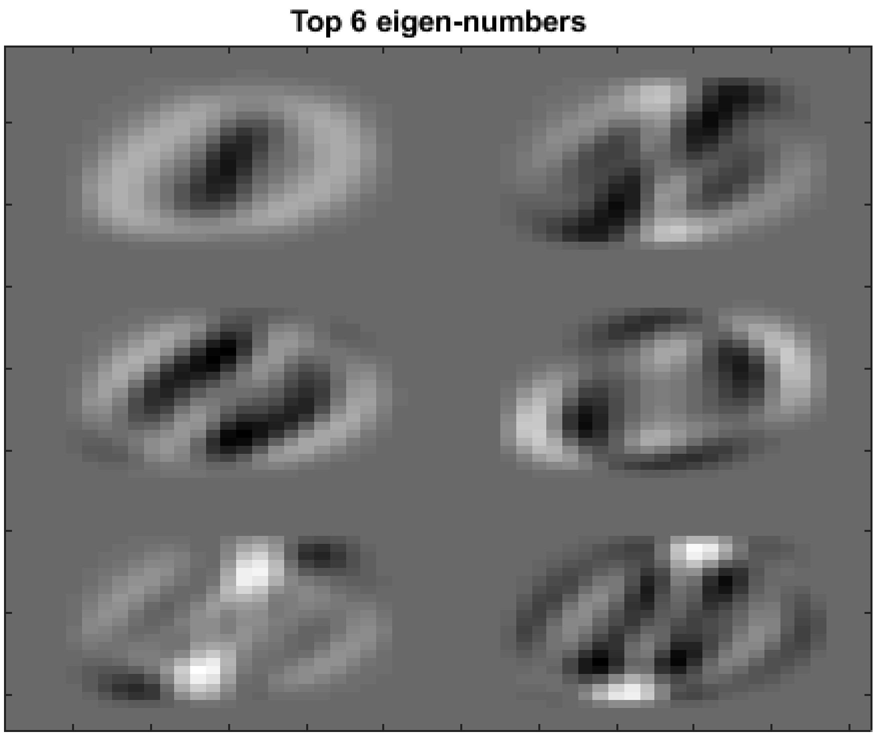
```
X = mnist.X;
M = 6;
[W, Z, mu, lambda] = princomp(X,M);
figure()
imgrid(mu, mnist.dims, [1, 1]);  % Mean face (kind of insulting)
title("Mean MNIST Image");
figure()
imgrid(W, mnist.dims, [3, 2]);  % 5 eigenfaces
title("Top 6 eigen-numbers");
```

25 Random Images from MNIST



Mean MNIST Image



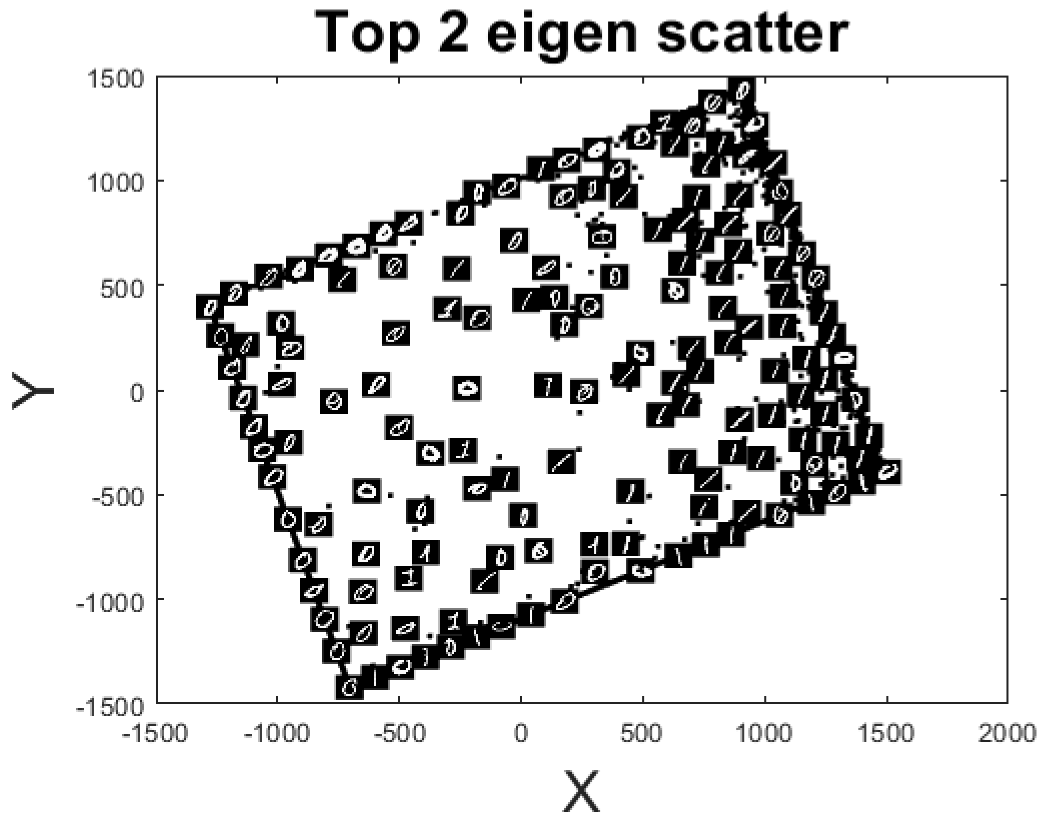


Part 2.4

Project the faces onto the first two principal components and plot/turn in the result as a point cloud in 2D (the best representation of the data in two dimensions), using either scatter or imcloud.m. **

I notice that from left to right there are more ones on the right than on the left. So the X eigenvector does a pretty good job of distinguishing between the two numbers. I am not sure what difference there is top to bottom, perhaps slantyness?

```
M = 2;
[W, Z, mu, lambda] = princomp(X,M);
figure()
imcloud(Z, X, mnist.dims)
title("Top 2 eigen scatter");
```

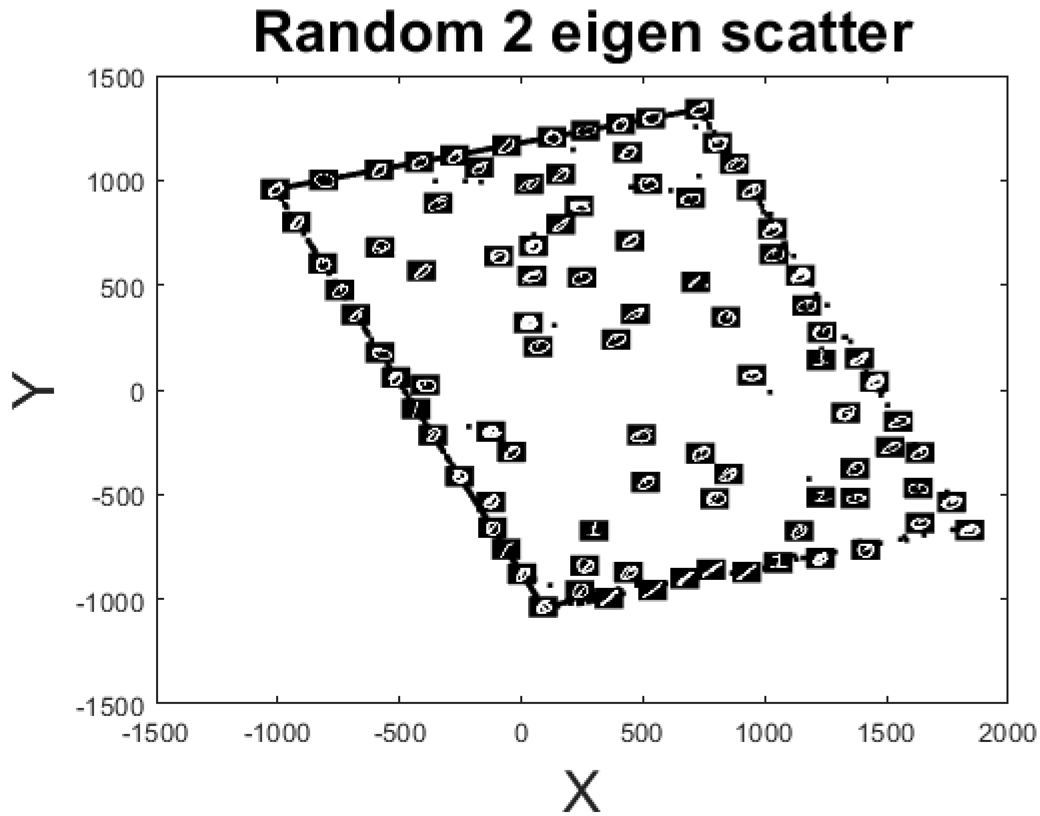


Part 2.5

Use the imcloud.m function to create this plot for projections onto other significant PCs, and explain the results you see as they compare to your response from looking at the PC vectors. How might you change or improve your description of the information captured by the principal components?

```
M = 6;
[W, Z, mu, lambda] = princomp(X,M);
comps = randperm(6, 2)
W2 = W(:, comps);
P = W2' \ (W2' * W2);  % projection matrix formula
Z2 = P' * (X - mu);
figure()
imcloud(Z2, X, mnist.dims)
title("Random 2 eigen scatter")
```

```
comps =

     4     1
```



Part 2.6

Plot the spectrum of your PCA: trace the eigenvalues of the covariance matrix in descending order. Does this maybe tell you how many principal components should be selected for "optimal" dimensionality reduction?
****
The vast majority is contained within the first eigenvector as it is 3 times that of even the second vector. This also helps explain the difference we see between zero and one left to right as it captures most of our variation.

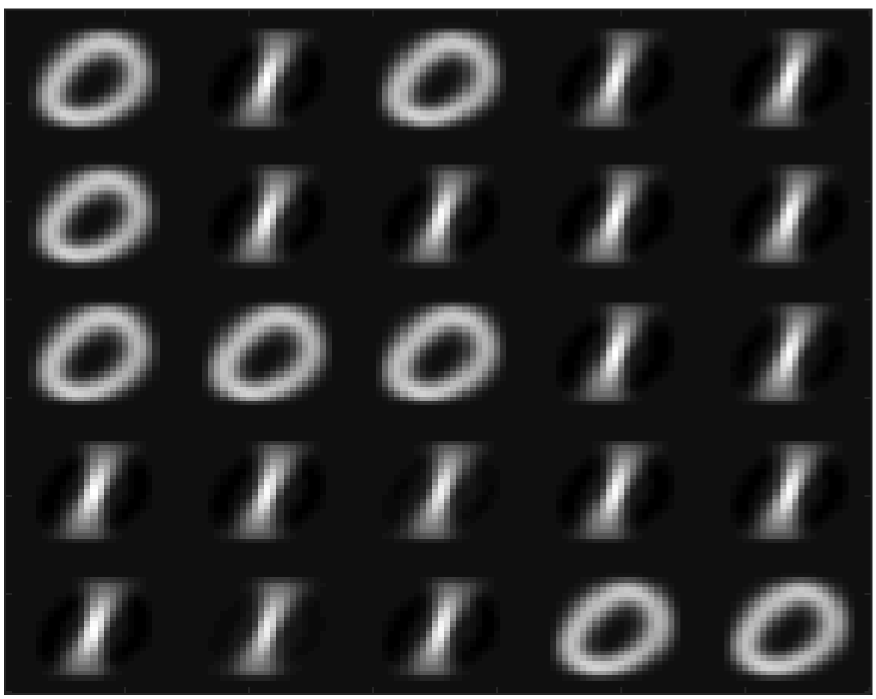
```
lambda


lambda =

   1.0e+10 *

    1.3090
    0.3697
    0.3298
    0.2265
    0.1594
    0.1383
```

Part 2.7

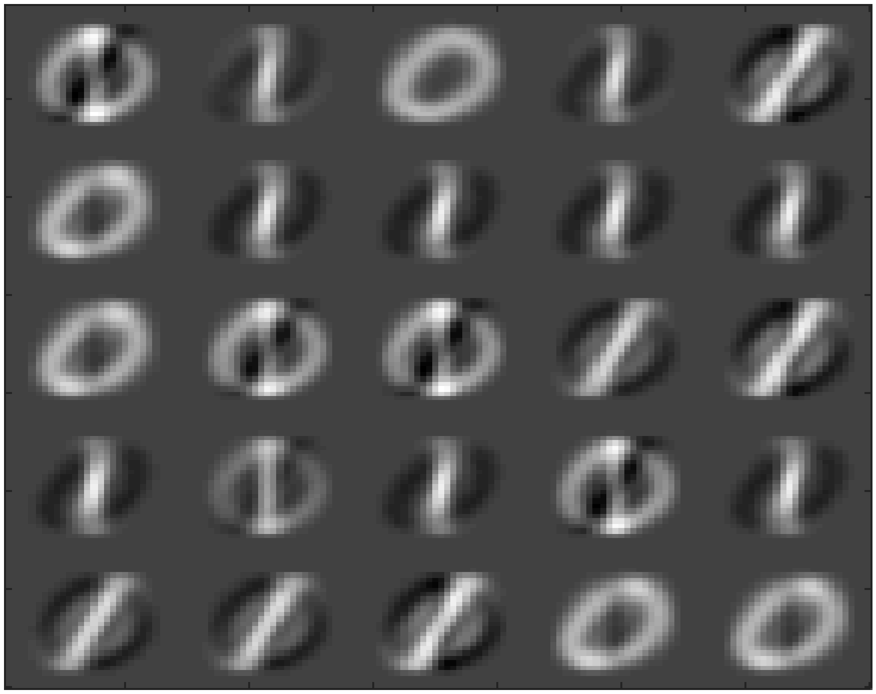
Reconstruct faces from varying number of principal components. For example, try M = 1,2,5,10,25, and compare the reconstructions against original numbers (you can chose the same 25 faces as under item 1, for example, and use imgrid.m with the reconstructed data). What do you observe?
****
It does a pretty good job of reconstructing the numbers. It is very very clean at M=1, which makes sense from 2.6.

```
for m = [1, 2, 5, 10, 25]
    [W, Z, mu, lambda] = princomp(X,m);
    Re = W * Z + mu;

    display_data = Re(:, faces);
    figure()
    imgrid(display_data, mnist.dims, [5, 5]);
    title("M = " + m + " reconstructed number");
end
```

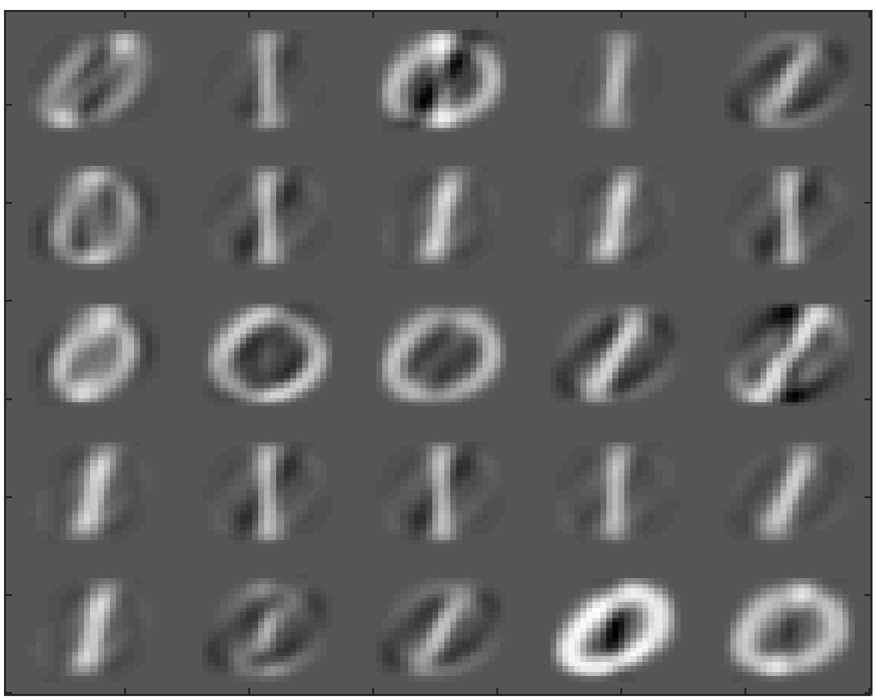
M = 1 reconstructed number



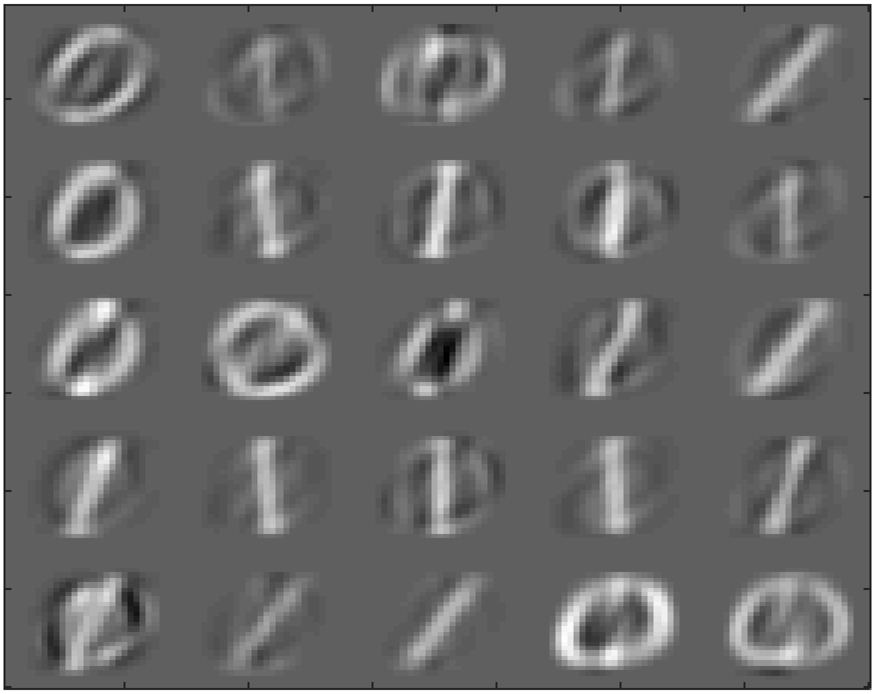
M = 2 reconstructed number



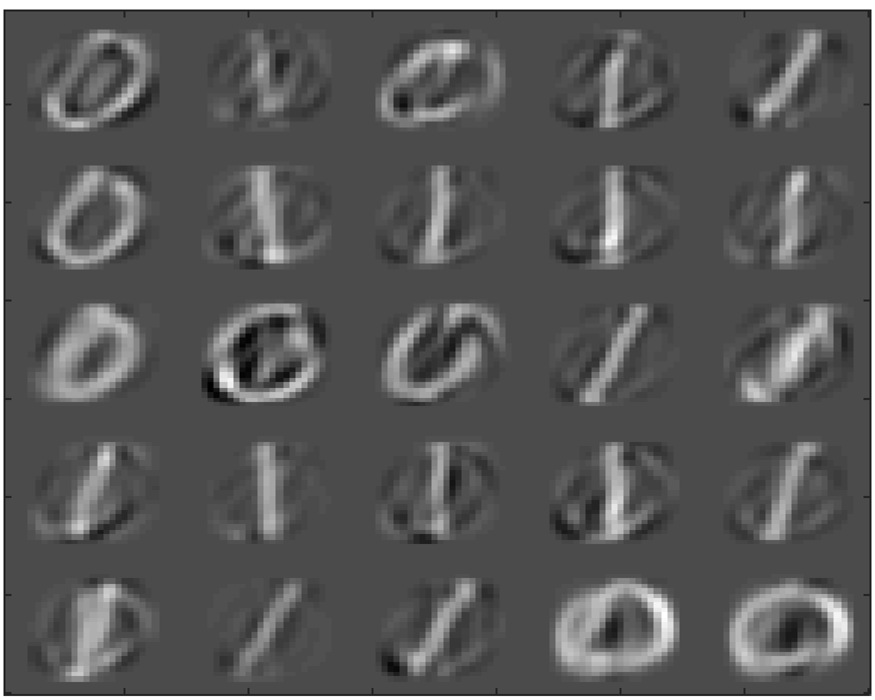
M = 5 reconstructed number



M = 10 reconstructed number



M = 25 reconstructed number



Published with MATLAB® R2021a