Assignment 4: CS 215

Devansh Jain	Harshit Varma
190100044	190100055

November 13, 2020

Question 4

Instructions for running the code:

1. Unzip and `cd` to `q4/code`, under this find the file named `q4.m`
2. Run the file, required plots for each digit will be generated and saved to the `q4/results` folder
   `eigenvalues_x.jpg` - plotting sorted Eigenvalues.
   `eigenvalues_log_x.jpg` - plotting sorted Eigenvalues with log-scaled x-axis.
   `trip_img_x.jpg` - Three images side-by-side.

First, we extract the images of handwritten digits from dataset `mnist.mat` and recast the training data into double and normalize it.
Normalization helps to reduce errors which otherwise occur due to high range of values in covariance matrix.
We also reshape the images into a vector for ease of computation.

```
DATA_PATH = "../data/mnist.mat";
load(DATA_PATH, "-mat"); % Load data
N = length(digits_train);
WIDTH = size(digits_train, 1);
SIZE = WIDTH^2;
% Reshape, Recast, Normalize image intensity
train_data = cast(reshape(digits_train, [SIZE N]), 'double')/255;
```

Now, we have to tasks ahead, to find the mean and covariance and then find the principal mode of variations (eigenvectors of covariance matrix).

```
count = sum(labels_train==digit);
digit_data = train_data(:, labels_train == digit);
mean = sum(digit_data, 2)/count; % sample mean
cov = (digit_data-mean)*(digit_data'-mean')/(count-1); % sample cov
```

The above code snippet is inside a for loop iterating over `digit` which stores the digit we are working on. First, we `count` the number of images (N) of the digit in the training data and then find sample mean $\mu = \sum_{i=1}^{N} x_i/N$ and sample covariance $C = \sum_{i=1}^{N}(x_i - \mu)(x_i - \mu)^T/(N-1)$.

Next, we use `eig` function for Eigen Decomposition of the covariance matrix.

```
[Q, D] = eig(cov);
dia = diag(D); % Get the diagonals elements as a vector

% Find the highest eigenvalue and its corresponding eigenvector
[lamb1, index1] = max(dia);
v1 = Q(:,index1);

% Sort the eigenvalues to analyse their distribution
dia = sort(dia,'descend');
dia(dia<0)=0; % Fixing precision error due to eig()
```

It was observed that `eig` returned eigenvalues like -10^{-10}, which we know is only precision error as eigenvalues of a Positive semi-definite matrix are non-negative.

We plot the sorted eigenvalues (784 in total) and we observe a sudden dip very soon. To further analyse, we plot it with log scale on X-axis and observe that by the time we reach 10, the values have fallen to less than 20% of highest value (located at 1) for all digits.
For some digits (especially 1), the decrease is more drastic than others.
We compute the number of eigenvalues having value more than 1% of the highest value and here is the result. (Also printed on Console in MATLAB)

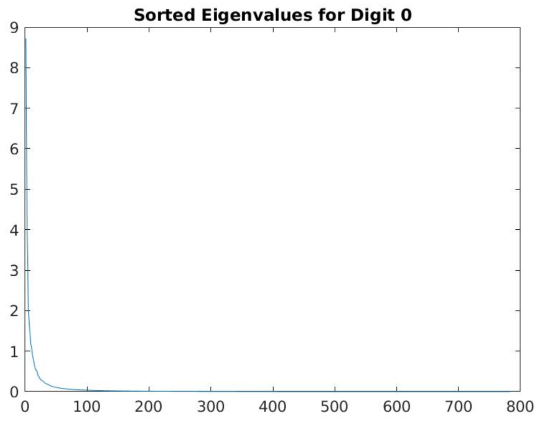
```
56 Significant modes of variation for Digit 0 (Number of Eigenvalues > 1% of Max Eigenvalue)
27 Significant modes of variation for Digit 1 (Number of Eigenvalues > 1% of Max Eigenvalue)
86 Significant modes of variation for Digit 2 (Number of Eigenvalues > 1% of Max Eigenvalue)
86 Significant modes of variation for Digit 3 (Number of Eigenvalues > 1% of Max Eigenvalue)
83 Significant modes of variation for Digit 4 (Number of Eigenvalues > 1% of Max Eigenvalue)
69 Significant modes of variation for Digit 5 (Number of Eigenvalues > 1% of Max Eigenvalue)
60 Significant modes of variation for Digit 6 (Number of Eigenvalues > 1% of Max Eigenvalue)
63 Significant modes of variation for Digit 7 (Number of Eigenvalues > 1% of Max Eigenvalue)
89 Significant modes of variation for Digit 8 (Number of Eigenvalues > 1% of Max Eigenvalue)
63 Significant modes of variation for Digit 9 (Number of Eigenvalues > 1% of Max Eigenvalue)
```

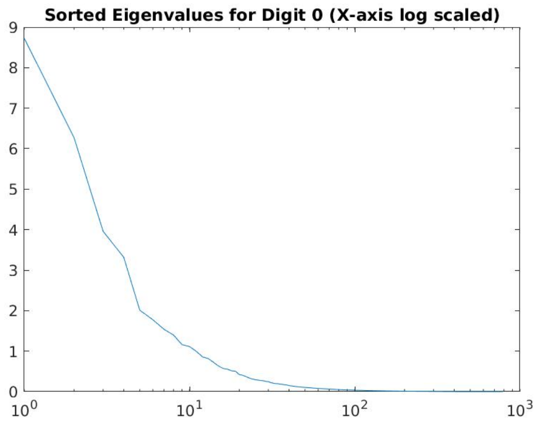
The observed values are far less than $28^2 = 784$.
One of the many obvious reason is the consistent background. There are several pixels in the background which are more or less left untouched by the person drawing the digits.
Another reason is that it is a common knowledge on how to draw a given digit and so the human-error is not prominent either.

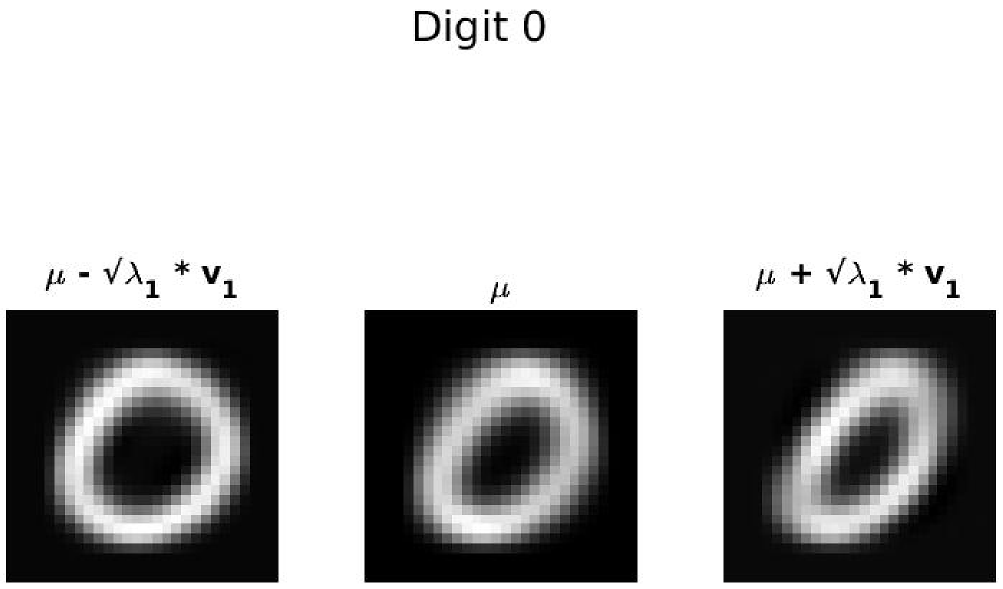
For the next part, we use the calculated principal mode of variation `v1` and it corresponding eigenvalue `lambda1` to draw[1] the 3 images of form: $\mu - \sqrt{\lambda_1}v_1$, μ and $\mu + \sqrt{\lambda_1}v_1$.
The range $(\mu - \sqrt{\lambda_1}v_1, \mu + \sqrt{\lambda_1}v_1)$ is analogous to the range $(\mu - \sigma, \mu + \sigma)$ for a univariate RV (Values lying within 1 standard deviation from μ). Here, $\sqrt{\lambda_1}v_1$ acts as the σ .
Thus, the "image range" $(\mu - \sqrt{\lambda_1}v_1, \mu + \sqrt{\lambda_1}v_1)$ acts as a range which captures the maximum variance in the way that people draw the image.
In all digits except 0 and 2, we observe that the three images mostly differ by rotation of the digit.

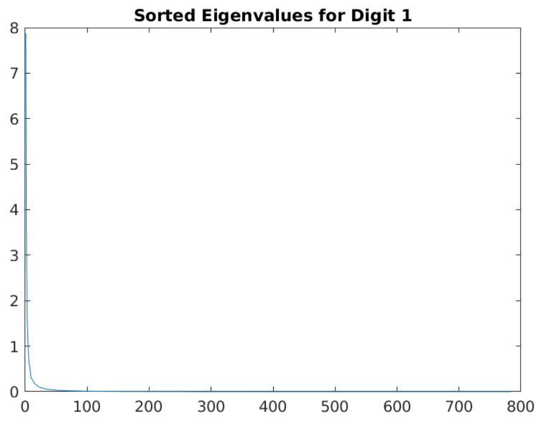
[1] I have used `pbaspect([1 1 1])` in place of suggested `axis equal`, here both are equivalent as range of both axes is (1,28)

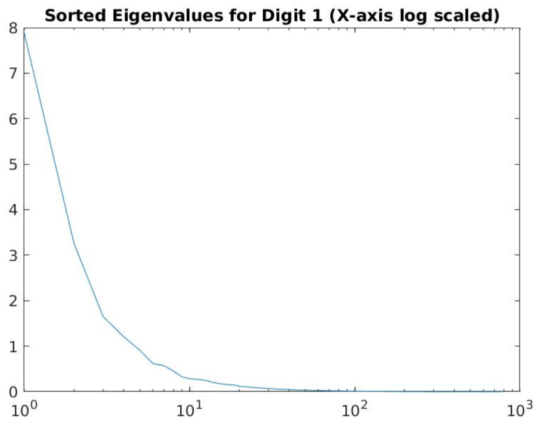
In digit 0, in addition to the difference in rotation, we also observe some circular to elliptical transform (compressing along the horizontal axis).
In digit 2, we observe that the lobe at the bottom turn while drawing digit varies in size along with a observable rotation (though less significant)
For digit 1, the principal mode of variation tells that people write 1 more like a straight line which maybe slanted (forward slash). Maximum variation is observed in angle of the line. Thus we can conclude that most people who drew "1" in this dataset draw it varying between a forward slash and vertical line.



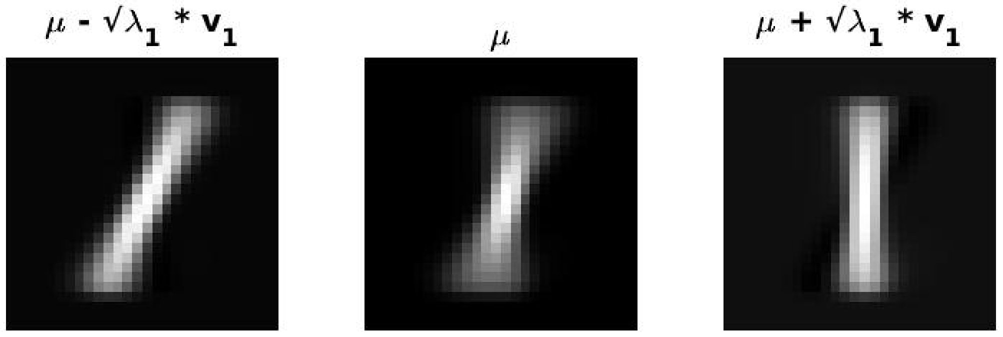


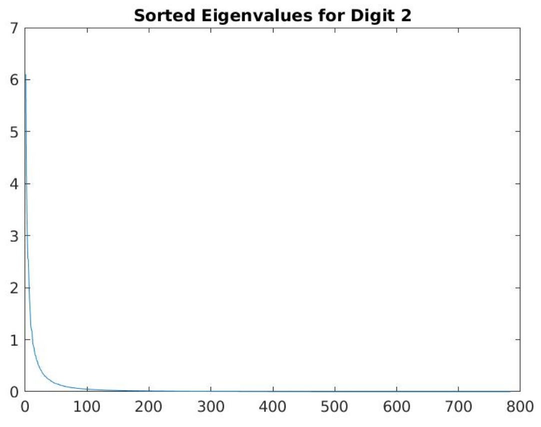

Digit 0
$\mu - \sqrt{\lambda_1} * \mathbf{v_1}$
μ
$\mu + \sqrt{\lambda_1} * \mathbf{v_1}$

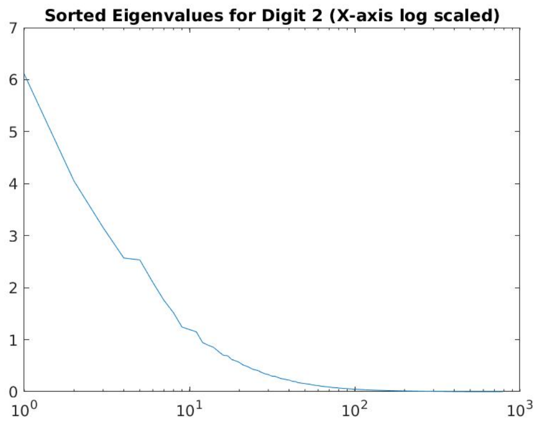

Sorted Eigenvalues for Digit 1
8
7
6
5
4
3
2
1
0
0
100
200
300
400
500
600
700
800


Sorted Eigenvalues for Digit 1 (X-axis log scaled)
8
7
6
5
4
3
2
1
0
10^0
10^1
10^2
10^3

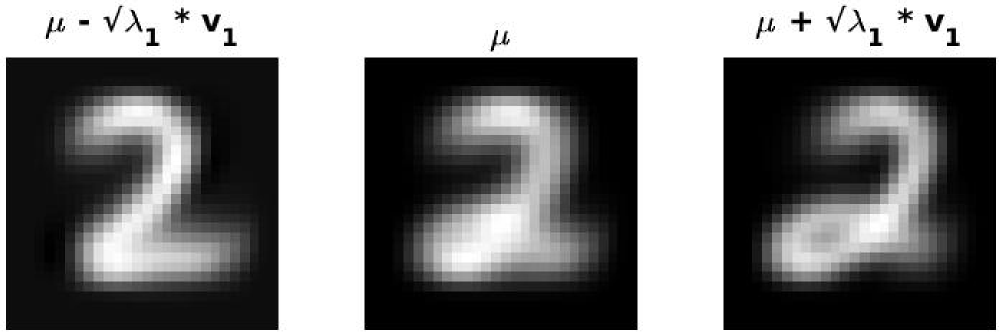
Digit 1

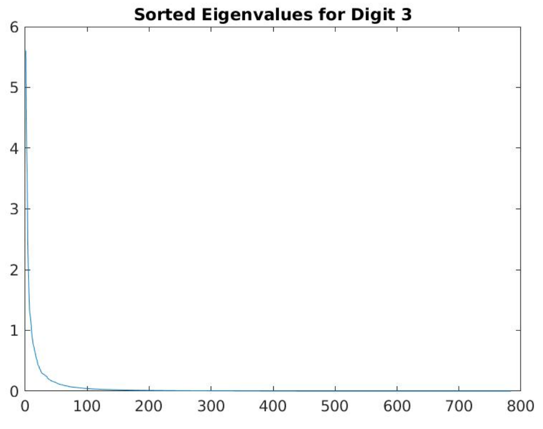

$\mu - \sqrt{\lambda_1} * \mathbf{v_1}$
μ
$\mu + \sqrt{\lambda_1} * \mathbf{v_1}$

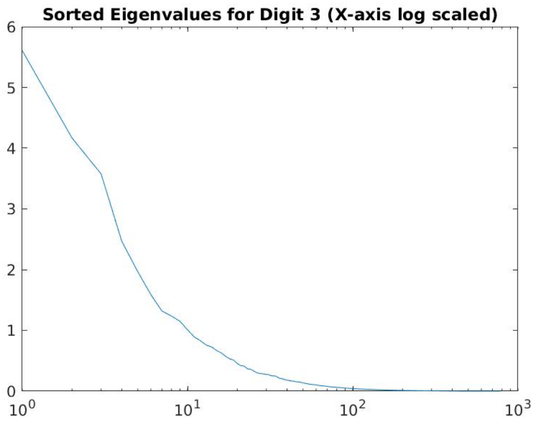

Sorted Eigenvalues for Digit 2
0
1
2
3
4
5
6
7
0
100
200
300
400
500
600
700
800

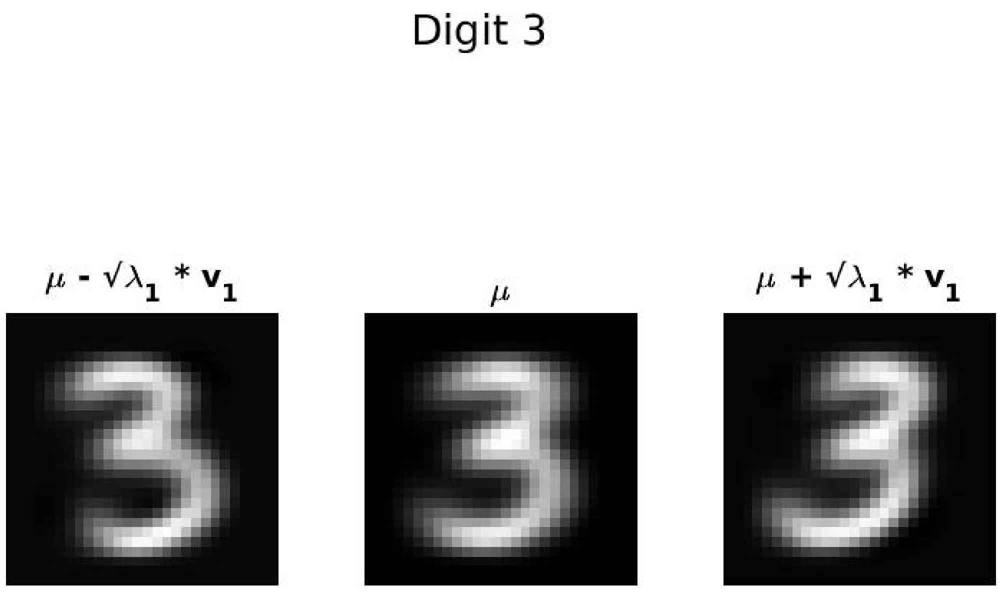

Sorted Eigenvalues for Digit 2 (X-axis log scaled)
0
1
2
3
4
5
6
7
10^0
10^1
10^2
10^3

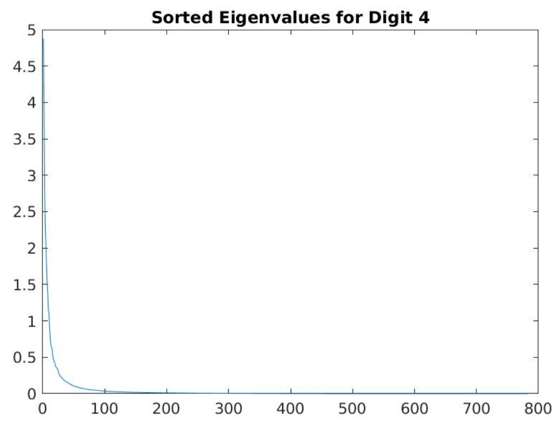
Digit 2

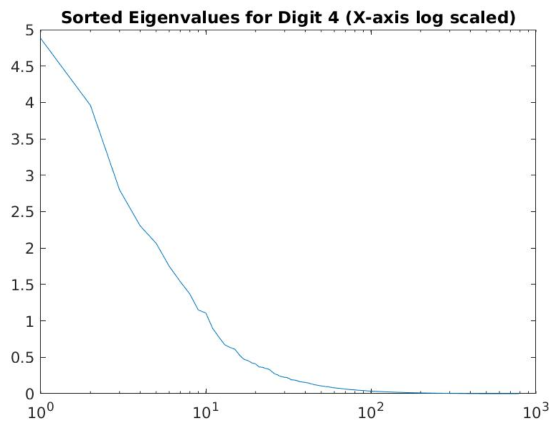

μ - √λ1 * v1
μ
μ + √λ1 * v1

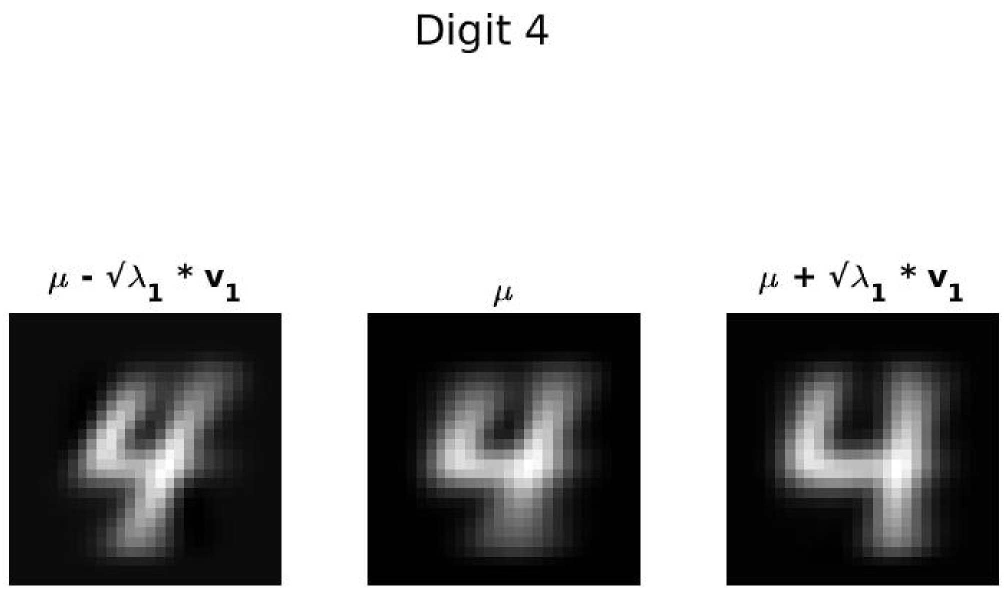

Sorted Eigenvalues for Digit 3
0
1
2
3
4
5
6
0
100
200
300
400
500
600
700
800

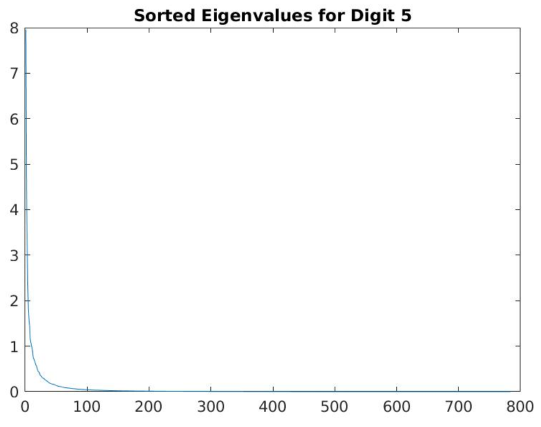

Sorted Eigenvalues for Digit 3 (X-axis log scaled)
0
1
2
3
4
5
6
10^0
10^1
10^2
10^3

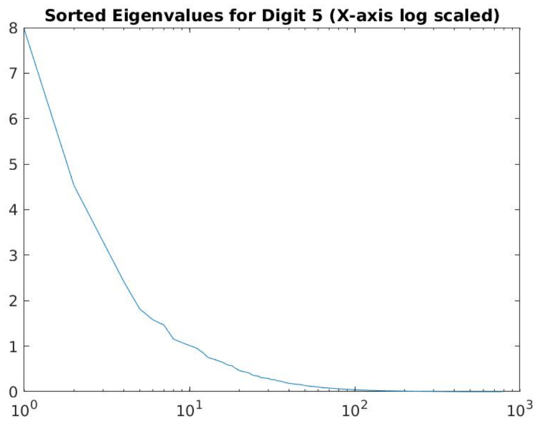

Digit 3
μ - √λ1 * v1
μ
μ + √λ1 * v1

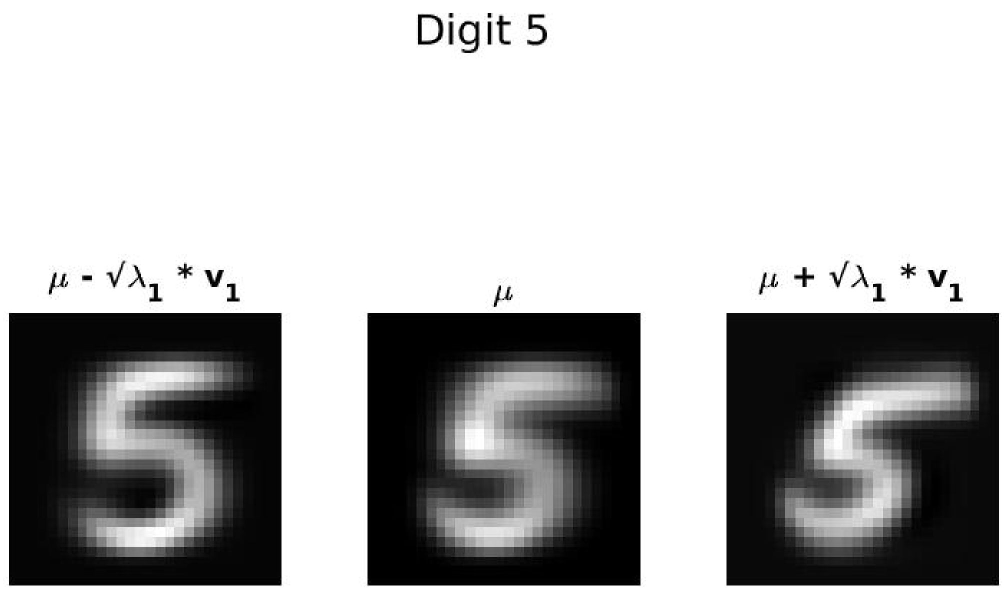

Sorted Eigenvalues for Digit 4
0
100
200
300
400
500
600
700
800
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5

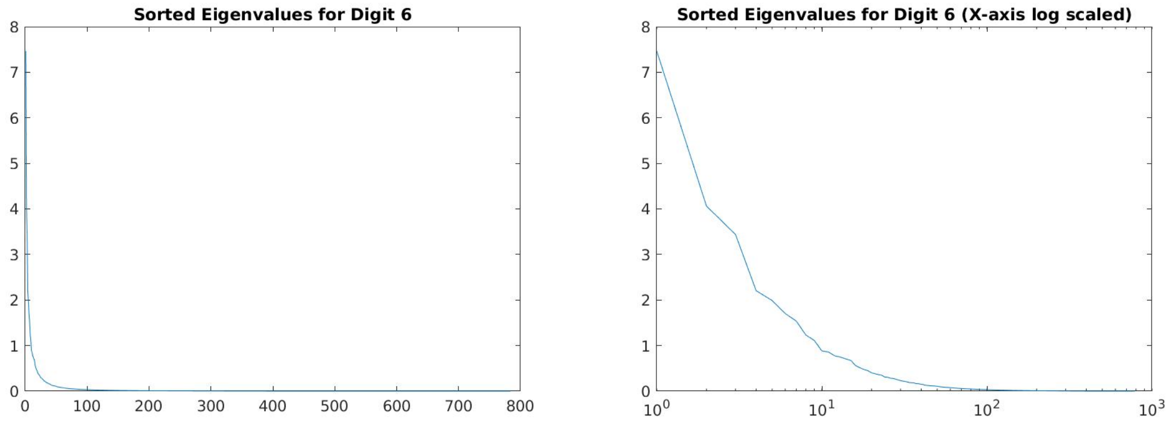

Sorted Eigenvalues for Digit 4 (X-axis log scaled)
10^0
10^1
10^2
10^3
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5

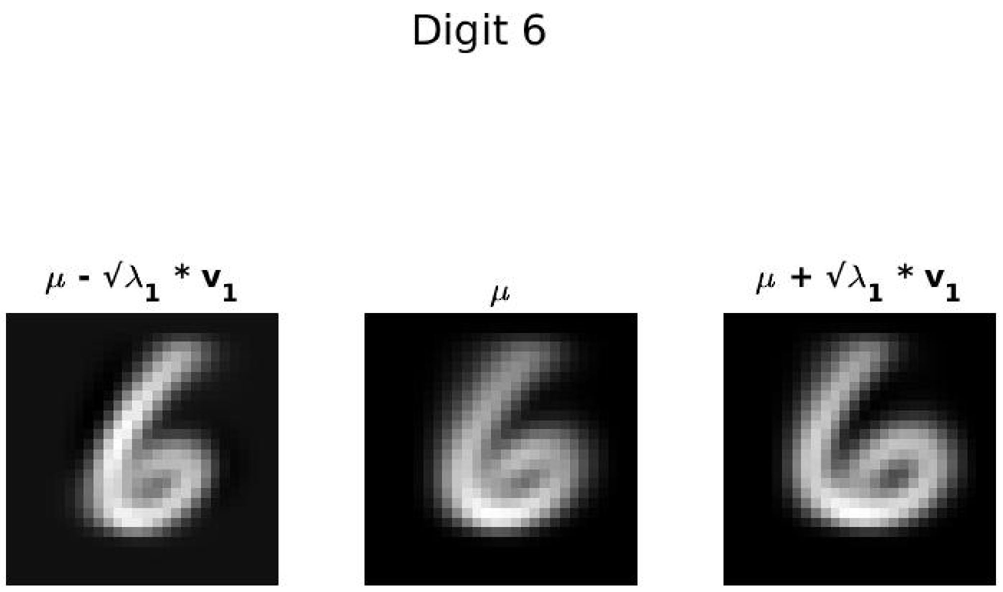

Digit 4
$\mu - \sqrt{\lambda_1} * \mathbf{v_1}$
μ
$\mu + \sqrt{\lambda_1} * \mathbf{v_1}$

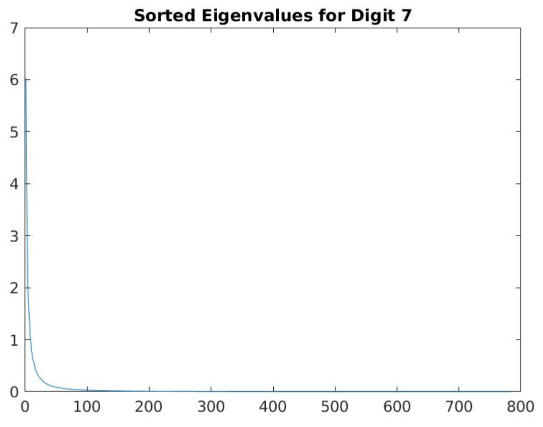

Sorted Eigenvalues for Digit 5
8
7
6
5
4
3
2
1
0
0
100
200
300
400
500
600
700
800

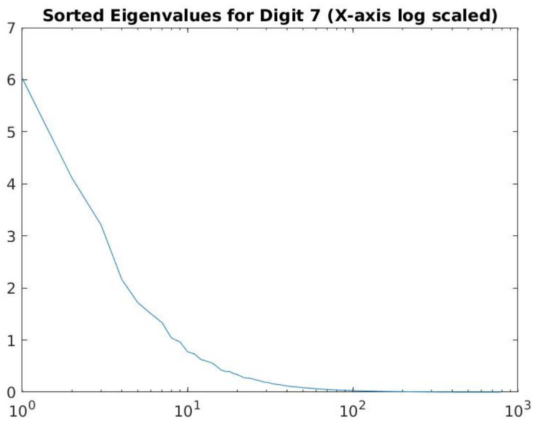

Sorted Eigenvalues for Digit 5 (X-axis log scaled)
8
7
6
5
4
3
2
1
0
10^0
10^1
10^2
10^3

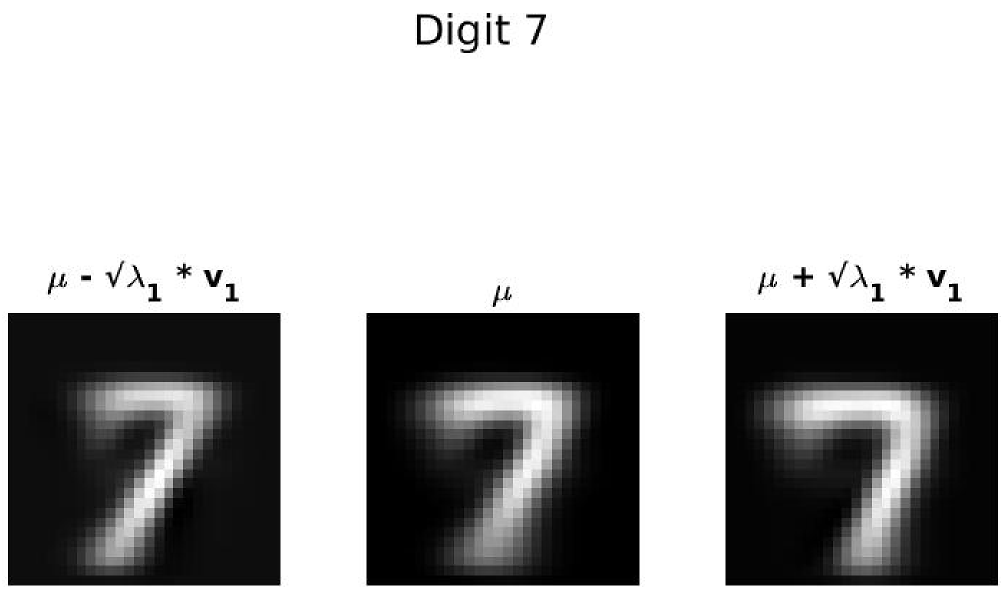

Digit 5
$\mu - \sqrt{\lambda_1} * \mathbf{v_1}$
μ
$\mu + \sqrt{\lambda_1} * \mathbf{v_1}$

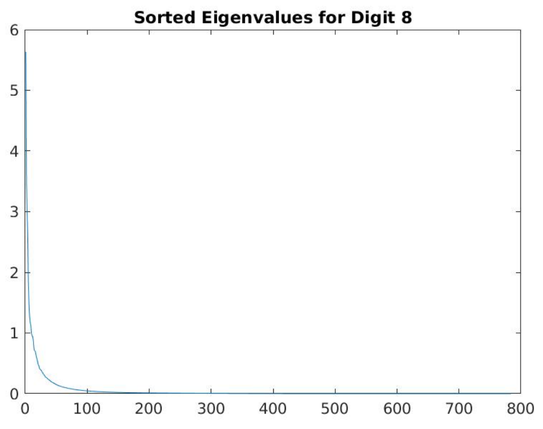

Sorted Eigenvalues for Digit 6
0
1
2
3
4
5
6
7
8
0
100
200
300
400
500
600
700
800
Sorted Eigenvalues for Digit 6 (X-axis log scaled)
10^0
10^1
10^2
10^3

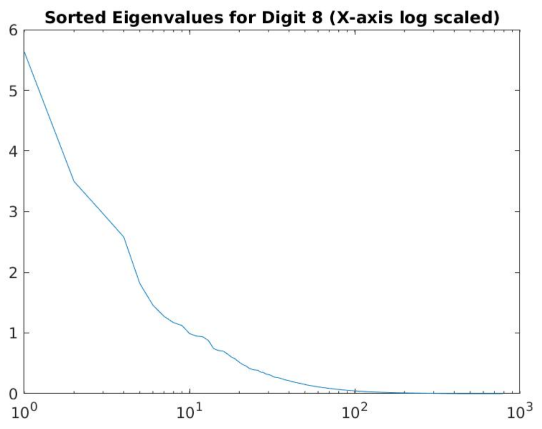

Digit 6
$\mu - \sqrt{\lambda_1} * \mathbf{v}_1$
μ
$\mu + \sqrt{\lambda_1} * \mathbf{v}_1$


Sorted Eigenvalues for Digit 7
0
1
2
3
4
5
6
7
0
100
200
300
400
500
600
700
800

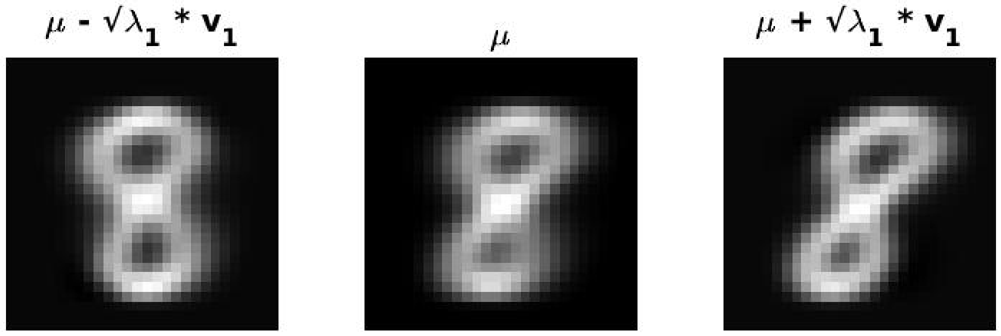

Sorted Eigenvalues for Digit 7 (X-axis log scaled)
0
1
2
3
4
5
6
7
10^0
10^1
10^2
10^3

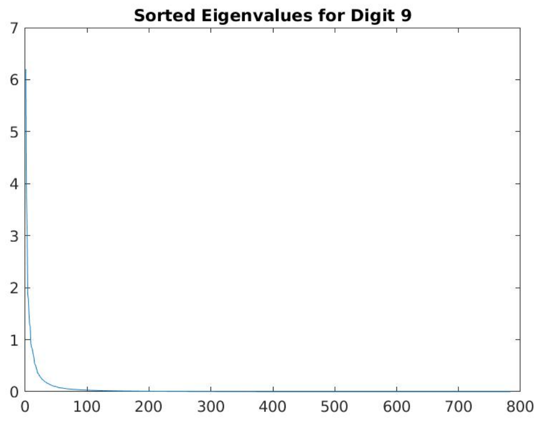

Digit 7
μ - $\sqrt{\lambda_1}$ * $\mathbf{v_1}$
μ
μ + $\sqrt{\lambda_1}$ * $\mathbf{v_1}$

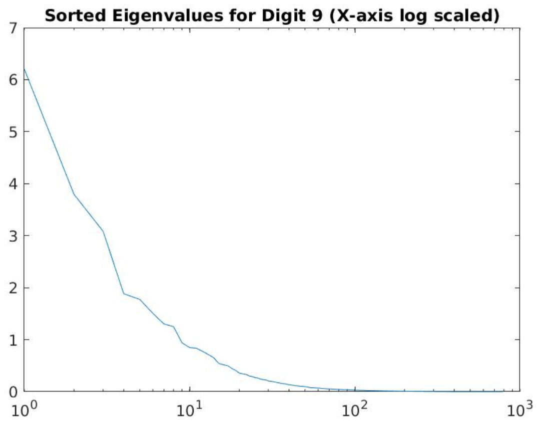

Sorted Eigenvalues for Digit 8
0
100
200
300
400
500
600
700
800
0
1
2
3
4
5
6

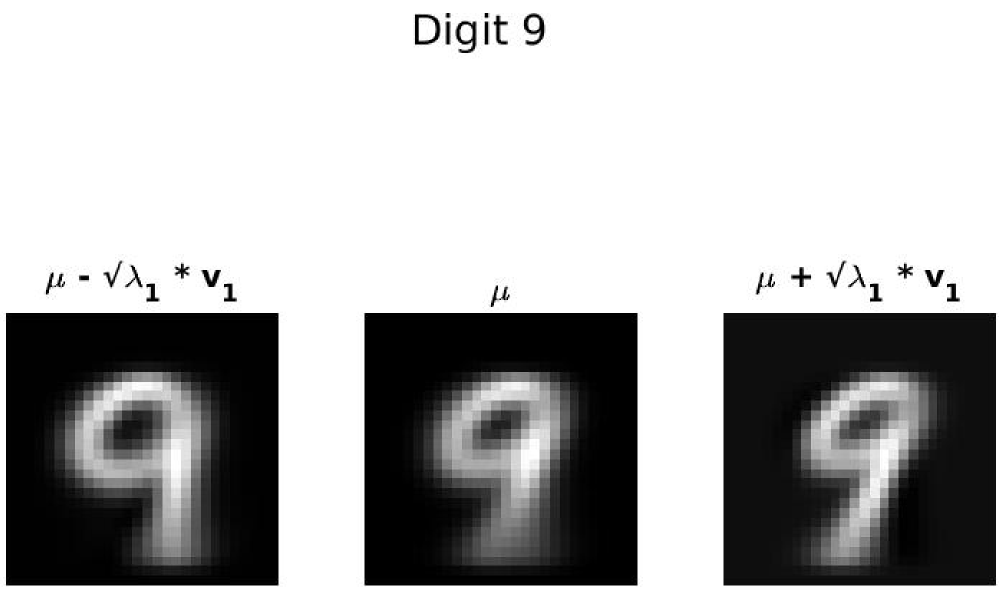

Sorted Eigenvalues for Digit 8 (X-axis log scaled)
10^0
10^1
10^2
10^3
0
1
2
3
4
5
6

Digit 8


$\mu - \sqrt{\lambda_1} * \mathbf{v_1}$
μ
$\mu + \sqrt{\lambda_1} * \mathbf{v_1}$


Sorted Eigenvalues for Digit 9
0
1
2
3
4
5
6
7
0
100
200
300
400
500
600
700
800


Sorted Eigenvalues for Digit 9 (X-axis log scaled)
0
1
2
3
4
5
6
7
10^0
10^1
10^2
10^3


Digit 9
$\mu - \sqrt{\lambda_1} * \mathbf{v_1}$
μ
$\mu + \sqrt{\lambda_1} * \mathbf{v_1}$