EE 482 Computational Neuroscience

Homework 4

Cemal Güven Adal

21703986

Name: Cemal Güven Adal
ID: 21703986

Question 1

In the first question a matrix which has the size of 1000x1024 is given. This matrix contains face images in the rows. I have worked with the hw4_data1 matrix as it is asked in the homework. Code is written in Matlab for using the dispImArray function which is given in the homework.

a) In part a of question 1 we are asked to do PCA in order to find the variance proportions of 1000 images that is given in the matrix. We will use PCA dimension reduction method and find the vectors which explains the variance the most.

In the following code matrix is loaded and PCA function used. Principal components for first twenty five images are found and displayed using dispImArray function that is given. These plots can be found below..Total explained variance is plotted. Code for these can be seen below:

```
% a_________
data=load('hw4_data1.mat');
face=data.faces();
meanface=mean(face,1);
face2=face-meanface;
[eigfaces,score,eigenvalues]=pca(face2);
yirmibeseig=eigfaces(:,1:25);
%25 res
figure()
dispImArray(yirmibeseig.',32)
title('First 25 Principal Components');
```

```
var=eigenvalues(1:100)/sum(eigenvalues);
figure()
plot(var)
title('Total Explained Variance by PC')
xlabel('PC');
ylabel('Variance');
size(meanface)
```

Resulting plots can be seen below:

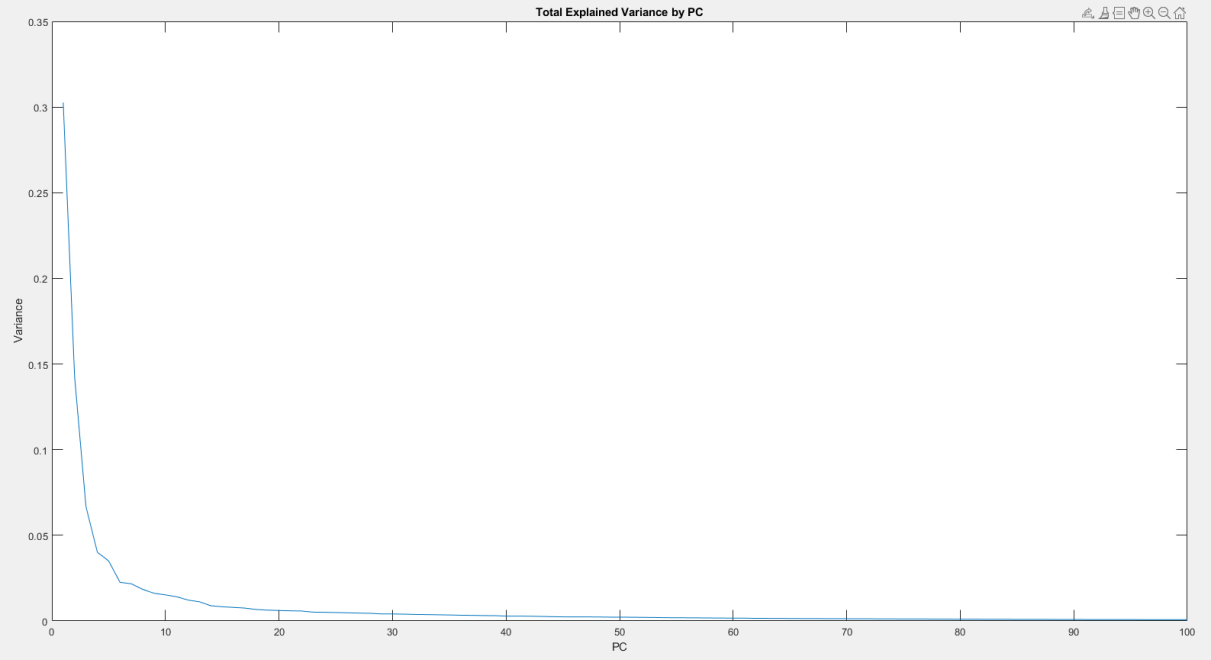


Figure 1: Total Explained Variance by PC

This plot shows that first 25 Principal Components are almost enough to evaluate the whole variance.

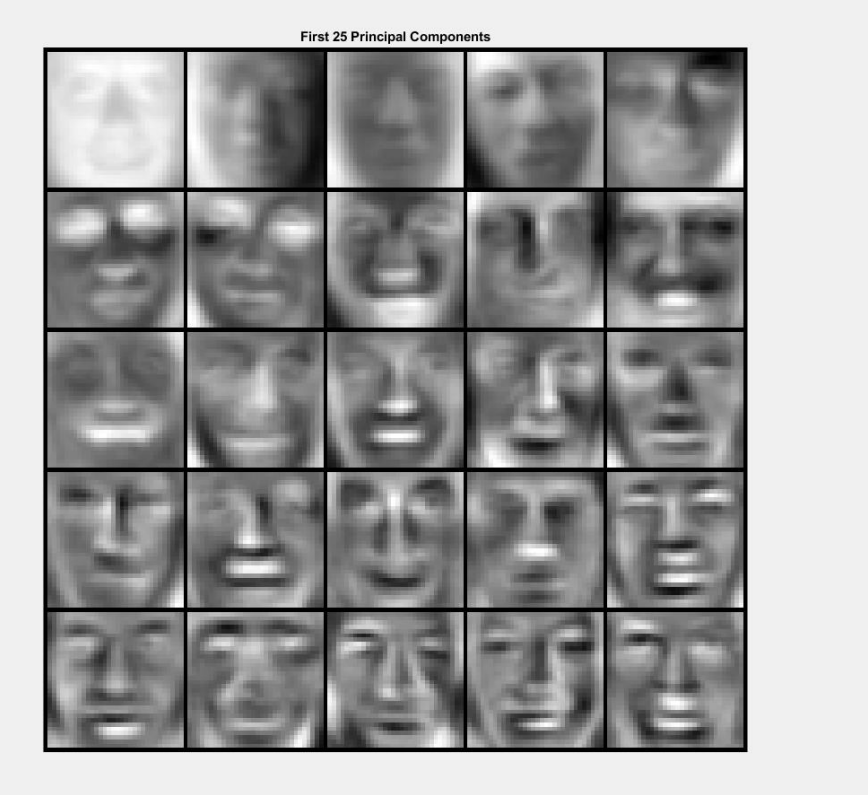


Figure 2: First 25 Principal Components

b) In this part I will reconstruct images the first 36 images for 10 25 and 50 as it is asked in the homework. To do this this formula is used:

$$Reconstruction = PC * Eigenvector^T + mean \quad (1)$$

This formula(1) is implemented in Matlab code and first 36 images. Code for it can be seen below:

```
% b----------------
eigten=eigfaces(:,1:10);
eigyirmibes=eigfaces(:,1:25);
eigelli=eigfaces(:,1:50);
innerproductten = face2*eigten ;
tendisp = innerproductten* eigten.' + meanface;
figure()
dispImArray(tendisp(1:36,:),32)
title('Images Reconstructed by 10 PC')
```

```
innerproductyirmibes = face2*eigyirmibes ;
yirmibesdisp = innerproductyirmibes* eigyirmibes.' +
meanface;
figure()
dispImArray(yirmibesdisp(1:36,:),32);
title('Images Reconstructed by 25 PC')
innerproductelli = face2*eigelli ;
ellidisp = innerproductelli* eigelli.' + meanface;
figure()
dispImArray(ellidisp(1:36,:),32);
title('Images Reconstructed by 50 PC')

figure()
dispImArray(face(1:36,:),32);
title('original faces');
```

Resulting images can be seen below:

Figure 3: Image Reconstructed by 10 PC

Figure 4: Image Reconstructed by 25 PC

Figure 5: Image Reconstructed by 50 PC

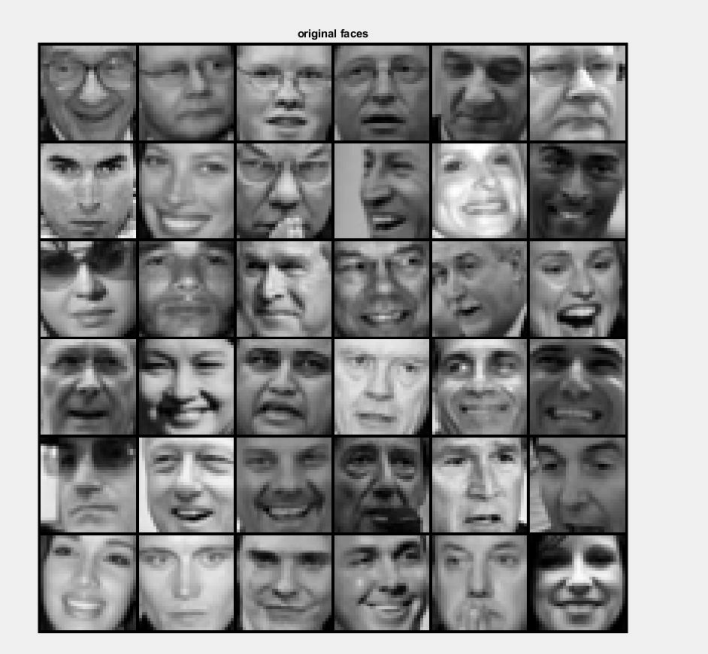


Figure 6: Original Face Images

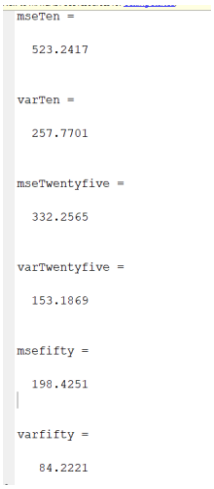
As it can be seen from the figures best result is given by reconstruction done by 50 when reconstructions compared with the original face images.

Next mean square error and standard deviation is found by 200 trials in the following code:

```
% mse var of ten
aten=sum((tendisp-face).*(tendisp-face),2)/1024;
mseTen=mean(aten)
varTen=std(aten)
% mse var of twentyfive
atwenty=sum((yirmibesdisp-face).*(yirmibesdisp-
face),2)/1024;
mseTwentyfive=mean(atwenty)
varTwentyfive=std(atwenty)
% mse var of fifty
afifty=sum((ellidisp-face).*(ellidisp-face),2)/1024;
mseTen=mean(afifty)
varTen=std(afifty)
```

Name: Cemal Güven Adal
ID: 21703986

The mean square error and standard deviation are found. For 10, mse = 523.2417, std= 257.7701. For 25 mse= 332.2565 and std=153.18. For 50 mse= 198.4251 and std= 84.2221. Matlab command results are given below:



From these results it can be seen that as the as the number of principal components increases both standard deviation and means square error decreases. Total variance can be best evaluated from 50 and after this point as it can be seen from the images and result mse will not decrease this rapidly because it gives a good insight for total variance. Images that are reconstructed got better and better as Principal Component number increased. I have plotted the original images and the result of 50 was really similar to original 36 images.

c) In part C we will use the ICA (Independent Component Analysis) instead of PCA since we have done so far to reconstruct images. For this FastIca package is given in moodle. FastICA package is used and first 36 images are reconstructed for first 10 ,25 and 50 principal components as it is done with PCA for part a and part b. Code for this can be seen below:

```
% part c
% 10
[tenIC, tenA, tenW] = fastica(face, 'lastEig', 50,'numOfIc',
10,'maxNumIterations', 5000);
figure(); sgtitle('IC 10');
dispImArray(tenIC, 32);
colorbar();
reconc=tenA*tenIC;
```

```
dispImArray(reconc(1:36,:),32);
% 25
[twentIC, twentA, twentW] = fastica(face, 'lastEig',
50,'numOfIc', 25,'maxNumIterations', 5000);
figure();
sgtitle('IC 25');
dispImArray(twentIC, 32);
colorbar();
rectwentc=twentA*twentIC;
dispImArray(rectwentc(1:36,:),32);
% 50
[elliIC, elliA, elliW] = fastica(face, 'lastEig',
50,'numOfIc', 50,'maxNumIterations', 5000);
figure();
sgtitle('IC 50');
dispImArray(elliIC, 32);
colorbar();
recellic=elliA*elliIC;
dispImArray(recellic(1:36,:),32);
```

Resulting images can be seen below:

Figure 7: ICA 10

Figure 8: ICA 25

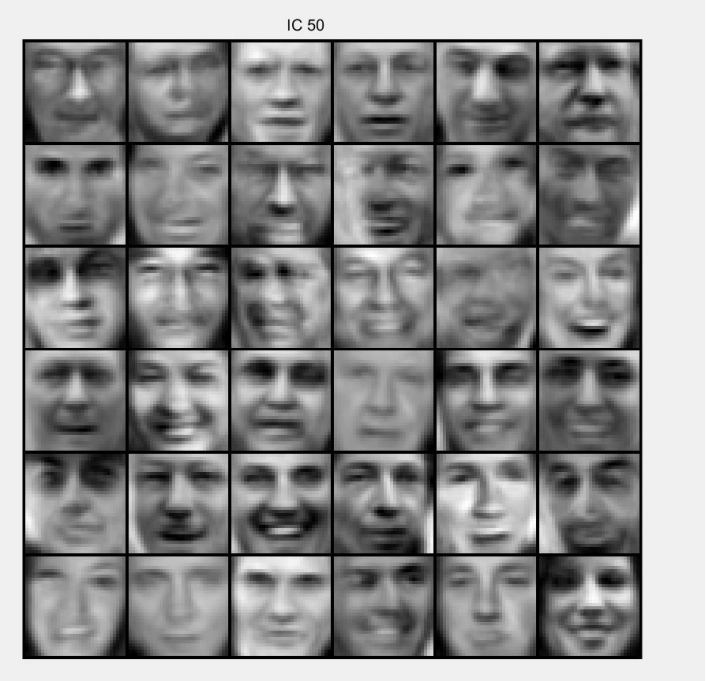


Figure 9: ICA 50

Then as it is asked in the homework mean square error and standard deviation is found for 10 , 25 and 50 principal components. Code for this can be seen below:

```
% mse,var 10
atenc=sum((reconc-face).*(reconc-face),2)/1024;
mseTen=mean(atenc)
varTen=std(atenc)

% mse,var 25
atwentc=sum((rectwentc-face).*(rectwentc-face),2)/1024;
msefifty=mean(atwentc)
varfifty=std(atwentc)
% mse,var 50
afiftyc=sum((recellic-face).*(recellic-face),2)/1024;
msefifty=mean(afiftyc)
varfifty=std(afiftyc)
```

Name: Cemal Güven Adal
ID: 21703986

The results are given below:

```
New to MATLAB? See resources for Getting Started.
mseTen =

   1.6689e+03


varTen =

  888.0073


msefifty =

   1.2057e+03


varfifty =

  730.0173


msefifty =

  554.5089


varfifty =

  535.9353
```

It can be seen from the results that for 10 mean of mse= 1668.9 and standard deviation mse=888.0073. For 25 mean of mse= 1205.7 and standard deviation mse=730.0173. For 50 mean of mse= 554.5089 and standard deviation mse=535.9353. As it can be evaluated from the result in ICA as the number of Principal components increases the mean square and std decreases as it did in the PCA. I would expect ICA would give similar results with PCA but as it can be seen from the results error is higher compared to PCA.

d) In part d I have used non negative matrix factorization in order to reconstruct the original images. For reconstructing the original matrix in nnmf method all pixels are made non negative by adding the minimum of the pixels to all pixels. In order to get the original matrix two lower rank matrices are multiplied and element that is added is subtracted. In the following code the non negative matrix factorization is used. W and reconstructed images using nnmf method is done by the following code:

```
face3=face+abs(min(face,[],'all'));
[W,H]=nnmf(face3.',10,'algorithm', 'mul');
figure()
dispImArray(W.',32);
```

```
title('W=10');
recon1=(W*H)-abs(min(face,[],'all'));
figure()
dispImArray(recon1(:,1:36).',32);
title('Non Negative Matrix Reconstruction of Images by 10');
% 25
[W25,H25]=nnmf(face3.',25,'algorithm', 'mul');
figure()
dispImArray(W25.',32);
title('W=25');
recon2=(W25*H25)-abs(min(face,[],'all'));
figure()
dispImArray(recon2(:,1:36).',32);
title('Non Negative Matrix Reconstruction of Images by 25');
% 50
[W50,H50]=nnmf(face3.',50,'algorithm', 'mul');
figure()
dispImArray(W50.',32);
title('W=50');
recon5=(W50*H50)-abs(min(face,[],'all'));
figure()
dispImArray(recon5(:,1:36).',32);
title('Non Negative Matrix Reconstruction of Images by 50');
```

Resulting images given below:

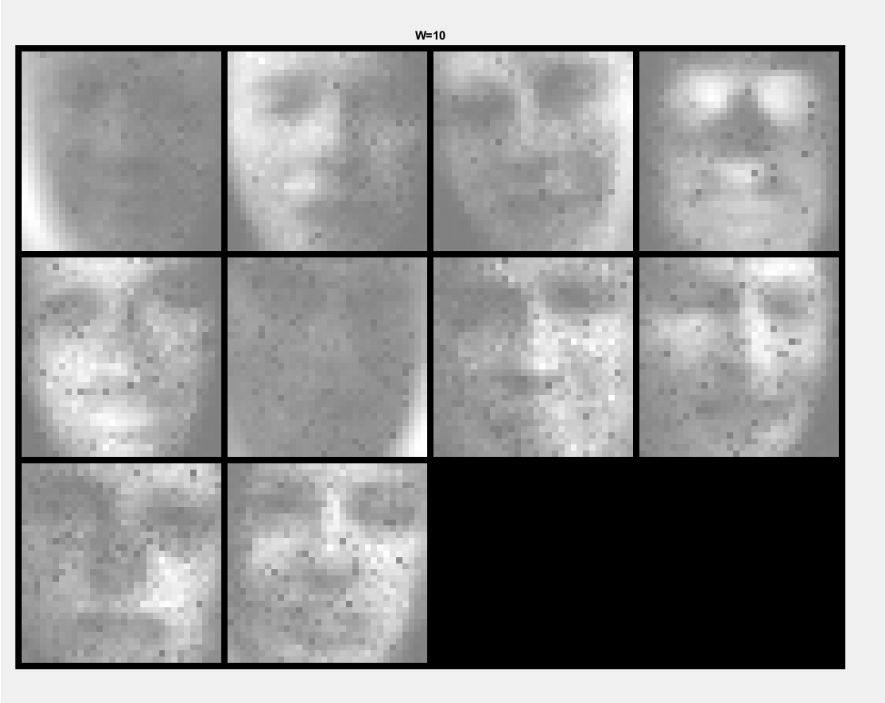


Figure 10: W 10

Figure 11: NNMF Reconstruction of Images by 10

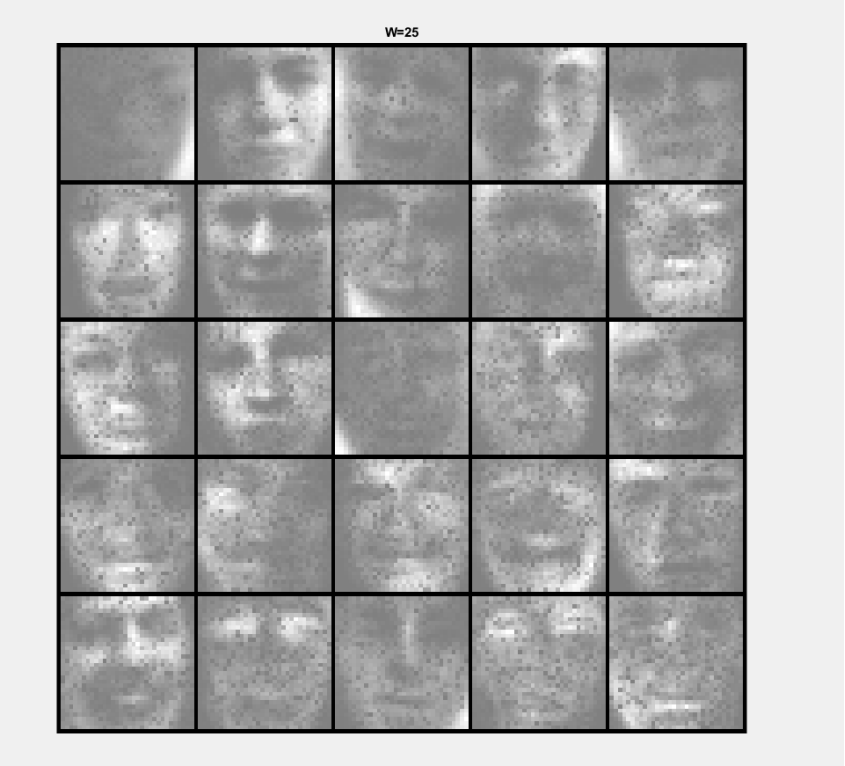


Figure 12: W 25

Figure 13: NNMF Reconstruction of Images by 25

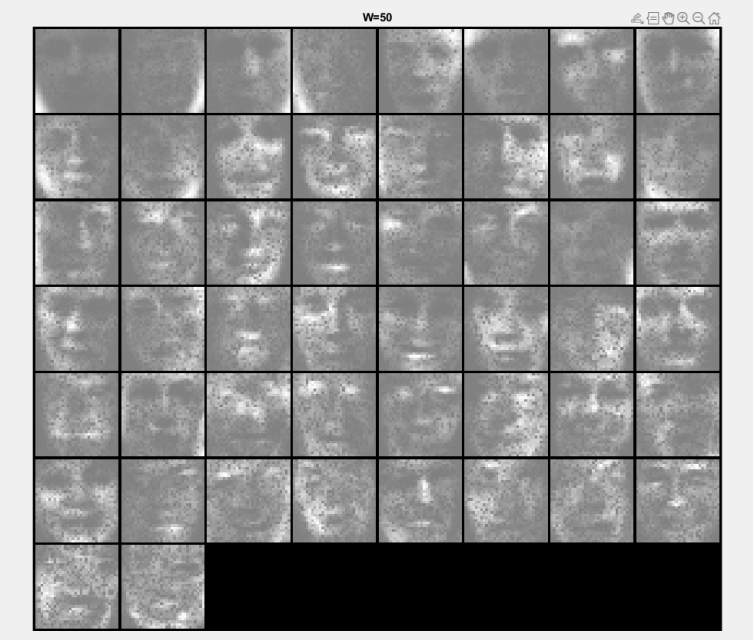


Figure 12: W 50

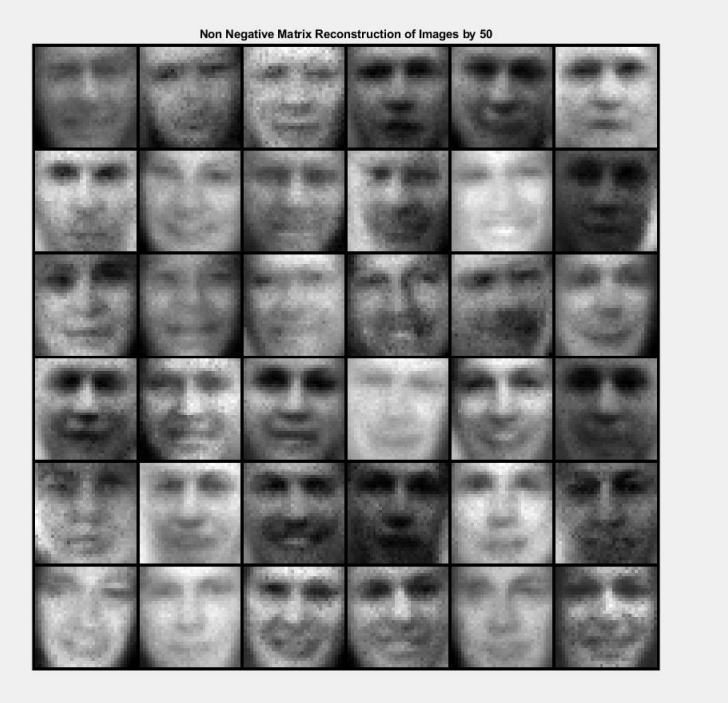


Figure 13: NNMF Reconstruction of Images by 50

As it is asked in the question mean and std of MSE is found. Following code for this can be seen below:

```
% mse,var 10
atend=sum((recon1.'-face).*(recon1.'-face),2)/1024;
```

```
mseTen=mean(atend)
varTen=std(atend)
% mse,var 25
atwentd=sum((recon2.'-face).*(recon2.'-face),2)/1024;
msetwentyfive=mean(atwentd)
vartwentyfive=std(atwentd)
% mse,var 50
afiftd=sum((recon5.'-face).*(recon5.'-face),2)/1024;
msefifty=mean(afiftd)
varfifty=std(afiftd)
```

From the result it can be seen that for 10 mse =664.7822 , std =342.3615. For 25 mean= 543.8575 and std= 268.1682. For 50 mse= 476.2839 and std=230.6808. As in the previous part as the number of principal components increases the mean and std of mse decreases. This method as it can be seen from the results is a bit worse compared to PCA in terms of error.Resulting output can be seen below from Matlab command window:

```
mseTen =

  664.7822


varTen =

  342.3615


msetwentyfive =

  543.8575


vartwentyfive =

  268.1682


msefifty =

  476.2839


varfifty =

  230.6808
```

From results and reconstructed images it can be seen that non negative matrix factorization method is not efficient for reconstructing these images as previous methods such as PCA but it is more efficient compared to ICA.

Name: Cemal Güven Adal
ID: 21703986

Question 2

a) In this question given tuning curves has a gaussian shape and function is given as:

$$f_i(x) = A \cdot e^{-(x-\mu_i)^2/(2\sigma_i^2)}$$

Given gaussian function for tuning curves are implemented in Matlab. And with this all tuning curves for the population are plotted in the same graph. Code for this can be seen below:

```
% Part a
u=[-10:10];
A=1
sigma=1
x=[-20:0.1:20];
figure()
for i=1:21
    f=A*exp(-((x-u(i)).^2)/(2*sigma));
    plot(x,f);
    hold on
end
hold off
title('Tuning Curves');
xlabel('Stimulus');
ylabel('Response');
```

Resulting tuning curve graph is given below:

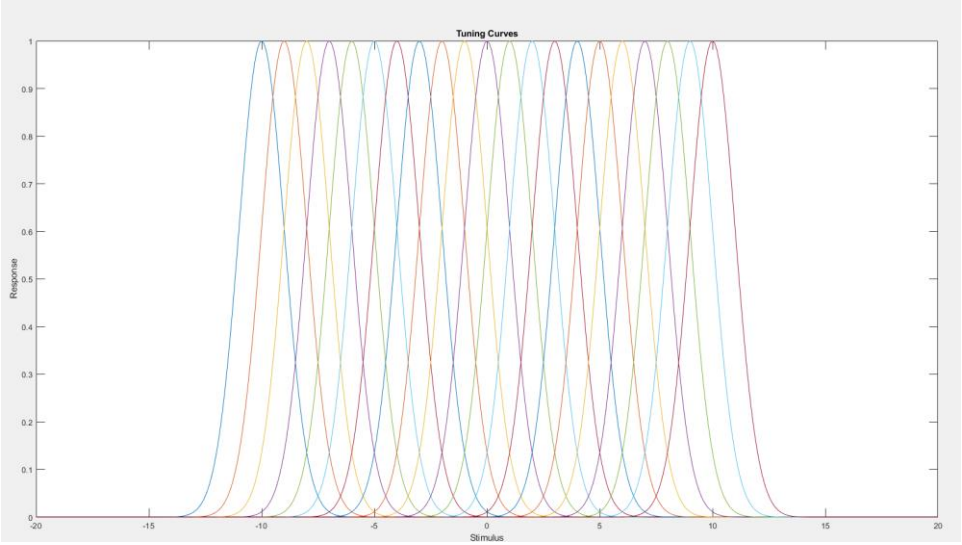


Figure 14: Tuning Curves

Then as it is asked population is simulated with x=1 and population response is plotted in the given code:

```
figure()
f1=A*exp(-((-1-u).^2)/(2*sigma));
plot(u,f1)
title('Preffered Stimulus Value');
ylabel('Response');
xlabel('Preffered Stimulus');
```

Result graph can be seen below:

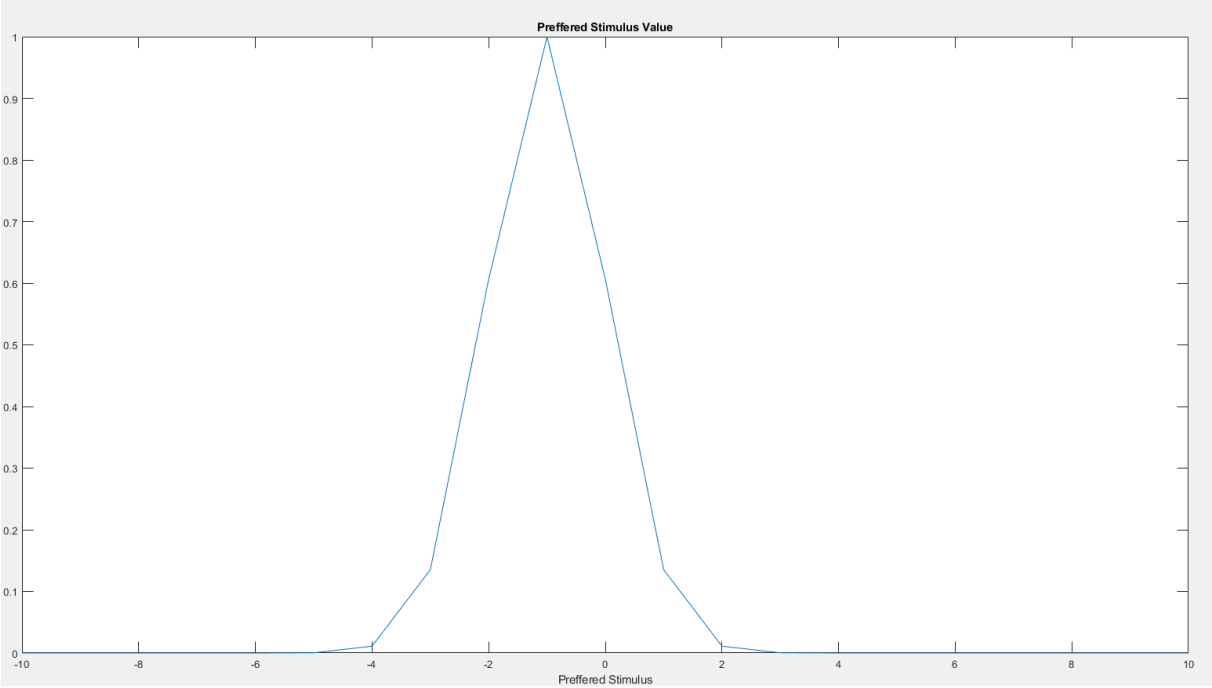


Figure 15: Preferred Stimulus Value

b) In this part 200 trials are simulated and winner take all decoder will be used in order to estimate the stimulus input. Winner take all decoder will estimate the input by the neuron with highest response. Code for this estimator is given below:

```
% b
r=unifrnd(-5,5,1,200);
noise=0.05*randn(1,200);
noisedr=r+noise;
estimate=zeros(200,1);
for j=1:200
  [M,I] =min(abs(noisedr(j)-u));
estimate(j)=u(I);
end
figure()
scatter(1:200,estimate)
hold on
scatter(1:200,r)
title('Actual Values vs Winner Take all decoder Estimated
Values');
```

```
xlabel('Trials');
ylabel('Stimulus');
```

Resulting figures are given below:

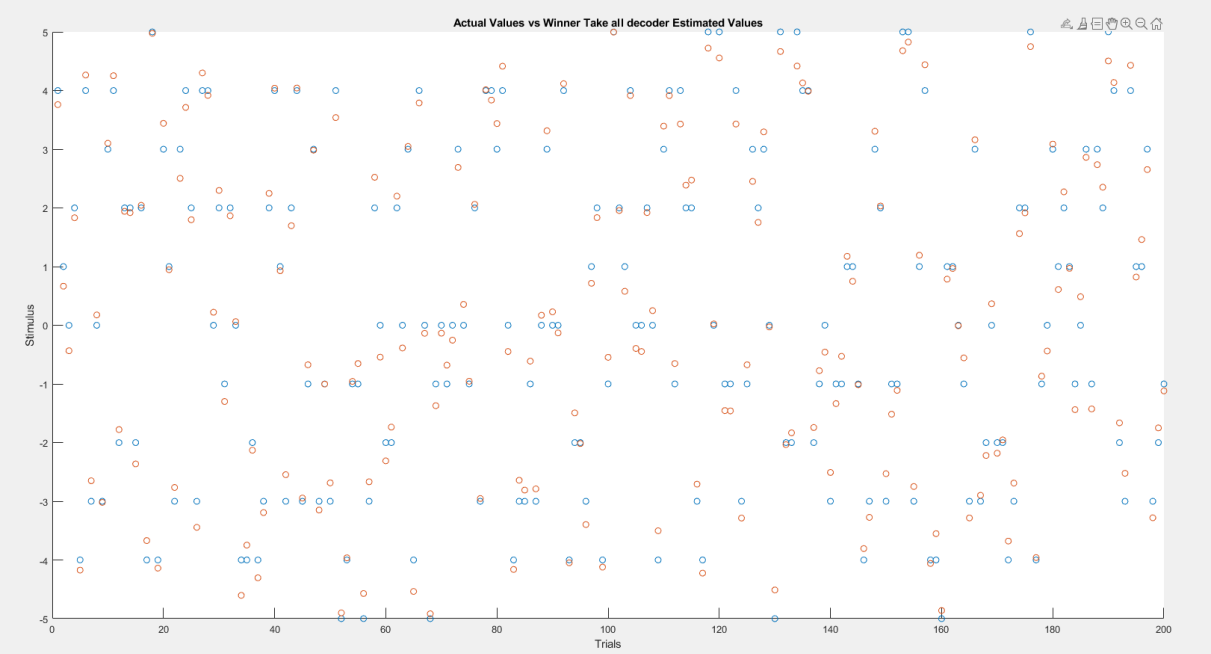


Figure 16: Actual vs Winner Take All Decoder Estimated Values

Then estimate error mean and estimate error standard deviation is found by following code:

```
var5=estimate-r;
estimateerror=mean(var5,'all')
estimatestd=std(var5(:))
```

The result is error mean of estimate error = -0.0233

Error standard deviation of estimate error = 4.1722

Matlab output is given below:

```
estimateerror =

   -0.0233


estimatestd =

    4.1722
```

c) In part c same applications are done with Maximum Likelihood Decoder. Maximum Likelihood Decoder is applied by taking noised version as estimate. Code is given for this application below:

```
% part c
r=unifrnd(-5,5,1,200);
noise=0.05*randn(1,200);
noisedr=r+noise;
estimate=zeros(200,1);
for j=1:200
  [M,I] =min(abs(noisedr(j)-u));
estimate(j)=u(I);
end
figure()
scatter(1:200,noisedr)
hold on
scatter(1:200,r)
title('Actual Values vs ML Estimated Values');
xlabel('Trials');
ylabel('Stimulus');
var6=estimate-noisedr;
estimateerror=mean(var6,'all')
estimatestd=std(var6(:))
```

Resulting figure is given below:

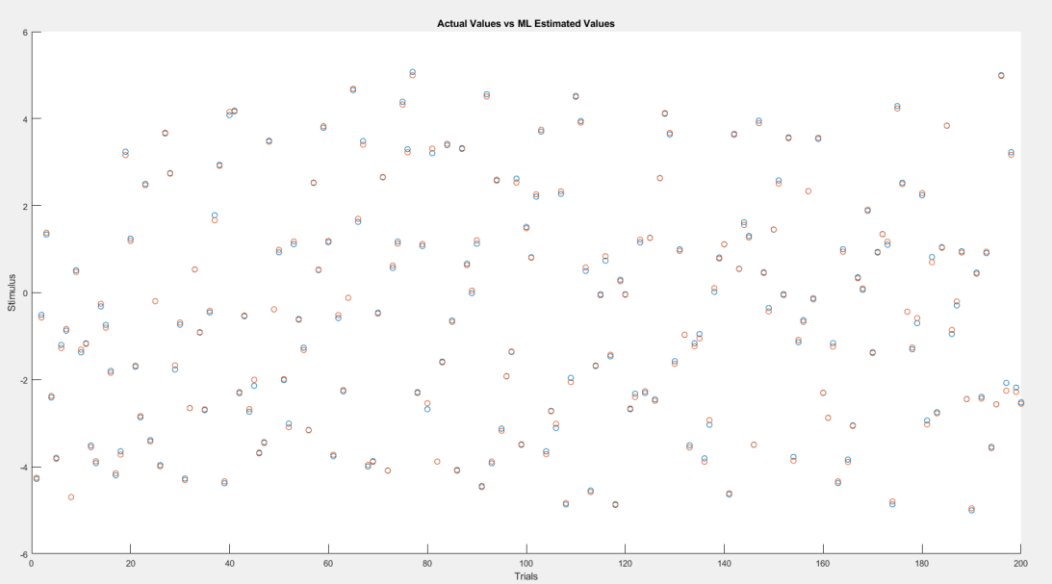


Figure 17: Actual vs MLE Decoder Estimated Values

The result shows us that:

The result is error mean of estimate error = 0.0112

Error standard deviation of estimate error = 3.7664

```
estimateerror =

    0.0112


estimatestd =

    3.7664
```

d) In part d Maximum Posterior Decoder is implemented. To do this input with the maximized posterior probability is found. Code for this can be seen below:

```
x=[-5:0.01:5];
prior=(1/(sqrt(2*pi)*2.5))*exp((-x.*x)/(2*2.5*2.5));
mapestimate=zeros(1,200);
for j=1:200
 mapestimate(j)= decoder(prior,r(j));

end
figure()
scatter(1:200,mapestimate);
hold on
```

```
scatter(1:200,r);
legend('mapestimate','originaltrial')
var7=abs(mapestimate-r);
estimateerror=mean(var7,'all')
estimatestd=std(var7(:))
```

Resulting figure can be seen below:

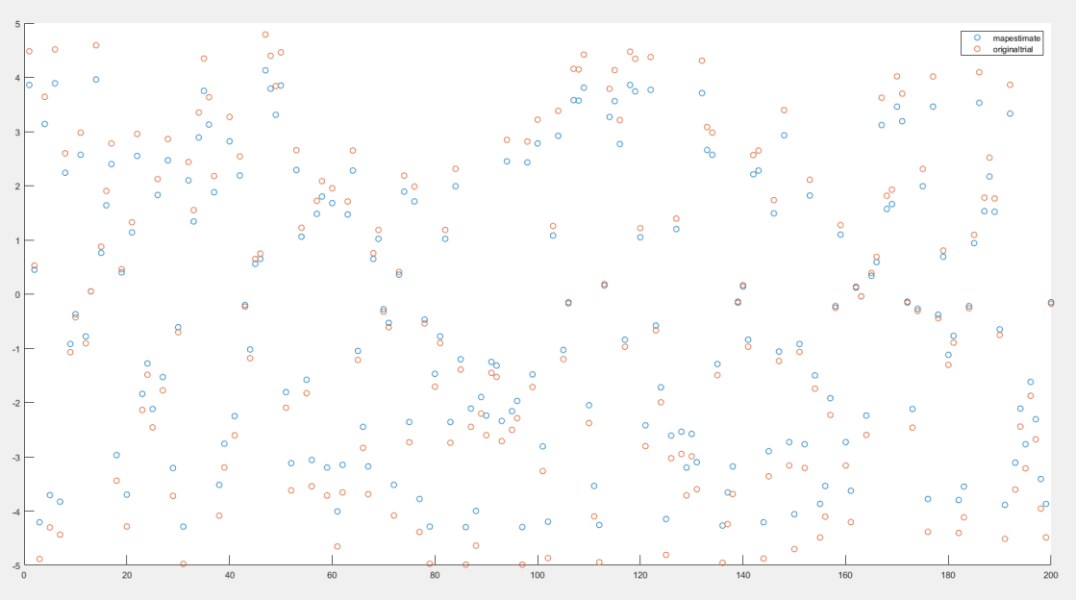


Figure 17: Actual vs MAP Decoder Estimated Values

Results show that mean of estimate error 2.5882 and standard deviation of estimate error is equal to 1.4392

```
estimateerror =

    2.5882


estimatestd =

    1.4392

>>
```

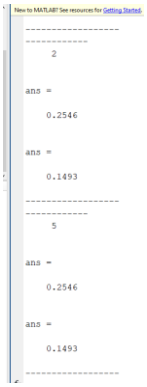
e) In part e ML decoder is applied. Sigma values will change with the given values in the assignment. Mean and standard deviations are found by the following code:

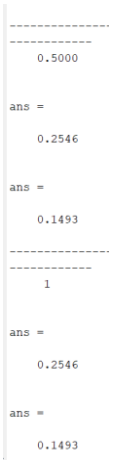
```
% part e
u=[-10:10];

sigma=[0.1,0.2,0.5,1,2,5];
for j=1:6
```

Name: Cemal Güven Adal
ID: 21703986

```
Mlestimate=zeros(1,200);
for i=1:200
    f5=exp((-(noisedr(i)-u).^2)/(2*sigma(j)*sigma(j)));
    [W,I]=max(f5);
    Mlestimate(1,i)=u(I);

end
disp("------------");
disp(sigma(j))
mean(abs(Mlestimate-r))
std(abs(Mlestimate-r))
disp("------------------");

end


end
```

I was expecting there to be more error as the standard deviation increased but in general there was no linear relation between error and sigma. But because wider tuning curves prone to more error makes more sense. Results can be seen below with the sigma value displayed on top.





Name: Cemal Güven Adal
ID: 21703986

References

Greenberg, M. D. (1988). *Advanced engineering mathematics*. Englewood Cliffs, NJ: Prentice-Hall Internat.

Bertsekas, D. P., & Tsitsiklis, J. N. (2008). *Introduction to probability: Dimitri P. Bertsekas and John N. Tsitsikis*. Belmont, Massachussets: Athena Scientific.

Appendix

Code

```
question = input("Enter Question Number",'s')
CemalGuven_Adal_21703986_hw4(question)
function CemalGuven_Adal_21703986_hw4(question)
clc
close all

switch question
    case '1'
    disp('1')
        % a_________
data=load('hw4_data1.mat');
face=data.faces();
meanface=mean(face,1);
face2=face-meanface;
[eigfaces,score,eigenvalues]=pca(face2);
yirmibeseig=eigfaces(:,1:25);
%25 res
figure()
dispImArray(yirmibeseig.',32)
title('First 25 Principal Components');

var1=eigenvalues(1:100)/sum(eigenvalues);
figure()
plot(var1)
title('Total Explained Variance by PC')
xlabel('PC');
ylabel('Variance');
size(meanface)
% b---------------
eigten=eigfaces(:,1:10);
eigyirmibes=eigfaces(:,1:25);
eigelli=eigfaces(:,1:50);
innerproductten = face2*eigten ;
tendisp = innerproductten* eigten.' + meanface;
figure()
dispImArray(tendisp(1:36,:),32)
```

```
title('Images Reconstructed by 10 PC')
innerproductyirmibes = face2*eigyirmibes ;
yirmibesdisp = innerproductyirmibes* eigyirmibes.' +
meanface;
figure()
dispImArray(yirmibesdisp(1:36,:),32);
title('Images Reconstructed by 25 PC')
innerproductelli = face2*eigelli ;
ellidisp = innerproductelli* eigelli.' + meanface;
figure()
dispImArray(ellidisp(1:36,:),32);
title('Images Reconstructed by 50 PC')

figure()
dispImArray(face(1:36,:),32);
title('original faces');

% mse var of ten
aten=sum((tendisp-face).*(tendisp-face),2)/1024;
mseTen=mean(aten)
varTen=std(aten)
% mse var of twentyfive
atwenty=sum((yirmibesdisp-face).*(yirmibesdisp-face),2)/1024;
mseTwentyfive=mean(atwenty)
varTwentyfive=std(atwenty)
% mse var of fifty
afifty=sum((ellidisp-face).*(ellidisp-face),2)/1024;
msefifty=mean(afifty)
varfifty=std(afifty)


% part c
% 10
[tenIC, tenA, tenW] = fastica(face, 'lastEig', 50,'numOfIc',
10,'maxNumIterations', 5000);
figure(); sgtitle('IC 10');
dispImArray(tenIC, 32);
colorbar();
reconc=tenA*tenIC;
dispImArray(reconc(1:36,:),32);
% 25
[twentIC, twentA, twentW] = fastica(face, 'lastEig',
50,'numOfIc', 25,'maxNumIterations', 5000);
figure();
sgtitle('IC 25');
dispImArray(twentIC, 32);
colorbar();
rectwentc=twentA*twentIC;
dispImArray(rectwentc(1:36,:),32);
% 50
```

Name: Cemal Güven Adal
ID: 21703986

```
[elliIC, elliA, elliW] = fastica(face, 'lastEig',
50,'numOfIc', 50,'maxNumIterations', 5000);
figure();
sgtitle('IC 50');
dispImArray(elliIC, 32);
colorbar();
recellic=elliA*elliIC;
dispImArray(recellic(1:36,:),32);
% mse,var 10
atenc=sum((reconc-face).*(reconc-face),2)/1024;
mseTen=mean(atenc)
varTen=std(atenc)

% mse,var 25
atwentc=sum((rectwentc-face).*(rectwentc-face),2)/1024;
msefifty=mean(atwentc)
varfifty=std(atwentc)
% mse,var 50
afiftyc=sum((recellic-face).*(recellic-face),2)/1024;
msefifty=mean(afiftyc)
varfifty=std(afiftyc)


% partd
% 10
face3=face+abs(min(face,[],'all'));
[W,H]=nnmf(face3.',10,'algorithm', 'mul');
figure()
dispImArray(W.',32);
title('W=10');
recon1=(W*H)-abs(min(face,[],'all'));
figure()
dispImArray(recon1(:,1:36).',32);
title('Non Negative Matrix Reconstruction of Images by 10');
% 25
[W25,H25]=nnmf(face3.',25,'algorithm', 'mul');
figure()
dispImArray(W25.',32);
title('W=25');
recon2=(W25*H25)-abs(min(face,[],'all'));
figure()
dispImArray(recon2(:,1:36).',32);
title('Non Negative Matrix Reconstruction of Images by 25');
% 50
[W50,H50]=nnmf(face3.',50,'algorithm', 'mul');
figure()
dispImArray(W50.',32);
title('W=50');
recon5=(W50*H50)-abs(min(face,[],'all'));
figure()
dispImArray(recon5(:,1:36).',32);
```

Name: Cemal Güven Adal
ID: 21703986

```
title('Non Negative Matrix Reconstruction of Images by 50');

% mse,var 10
atend=sum((recon1.'-face).*(recon1.'-face),2)/1024;
mseTen=mean(atend)
varTen=std(atend)
% mse,var 25
atwentd=sum((recon2.'-face).*(recon2.'-face),2)/1024;
msetwentyfive=mean(atwentd)
vartwentyfive=std(atwentd)
% mse,var 50
afiftd=sum((recon5.'-face).*(recon5.'-face),2)/1024;
msefifty=mean(afiftd)
varfifty=std(afiftd)
    y = myfunction(3,5)
    case '2'
    disp('2')
        % QUESTION2-------------------------------------
% Part a
u=[-10:10];
A=1
sigma=1
x=[-20:0.1:20];
figure()
for i=1:21
    f=A*exp(-((x-u(i)).^2)/(2*sigma));
    plot(x,f);
    hold on
end
hold off
title('Tuning Curves');
xlabel('Stimulus');
ylabel('Response');

figure()
f1=A*exp(-((-1-u).^2)/(2*sigma));
plot(u,f1)
title('Preffered Stimulus Value');
ylabel('Response');
xlabel('Preffered Stimulus');
% b

r=unifrnd(-5,5,1,200);
noise=0.05*randn(1,200);
noisedr=r+noise;
estimate=zeros(200,1);
for j=1:200
  [M,I] =min(abs(noisedr(j)-u));
estimate(j)=u(I);
end
figure()
```

Name: Cemal Güven Adal
ID: 21703986

```
scatter(1:200,estimate)
hold on
scatter(1:200,r)
title('Actual Values vs Winner Take all decoder Estimated
Values');
xlabel('Trials');
ylabel('Stimulus');
var5=estimate-r;
estimateerror=mean(var5,'all')
estimatestd=std(var5(:))
% part c
r=unifrnd(-5,5,1,200);
noise=0.05*randn(1,200);
noisedr=r+noise;
estimate=zeros(200,1);
for j=1:200
  [M,I] =min(abs(noisedr(j)-u));
estimate(j)=u(I);
end
figure()
scatter(1:200,noisedr)
hold on
scatter(1:200,r)
title('Actual Values vs ML Estimated Values');
xlabel('Trials');
ylabel('Stimulus');
var6=estimate-noisedr;
estimateerror=mean(var6,'all')
estimatestd=std(var6(:))
% part d
x=[-5:0.01:5];
prior=(1/(sqrt(2*pi)*2.5))*exp((-x.*x)/(2*2.5*2.5));
mapestimate=zeros(1,200);
for j=1:200
 mapestimate(j)= decoder(prior,r(j));

end
figure()
scatter(1:200,mapestimate);
hold on
scatter(1:200,r);
legend('mapestimate','originaltrial')
var7=abs(mapestimate-r);
estimateerror=mean(var7,'all');
estimatestd=std(var7(:));
% part e
u=[-10:10];

sigma=[0.1,0.2,0.5,1,2,5];
for j=1:6
Mlestimate=zeros(1,200);
```

Name: Cemal Güven Adal
ID: 21703986

```
for i=1:200
    f5=exp(-((noisedr(i)-u).^2)/(2*sigma(j)*sigma(j)));
    [W,I]=max(f5);
    Mlestimate(1,i)=u(I);

end
disp("------------");
disp(sigma(j))
mean(abs(Mlestimate-r))
std(abs(Mlestimate-r))
disp("------------------");

end


end

end

function y = myfunction(a,b)
y = a+b;
end



function mapdecoder=decoder(prior,u)
x=[-5:0.01:5];
likelihood=exp((-(x-u).*(x-u))/2);
posterior=likelihood.*prior;
[W,I]=max(posterior);
mapdecoder=x(I);
end
```

Name: Cemal Güven Adal
ID: 21703986