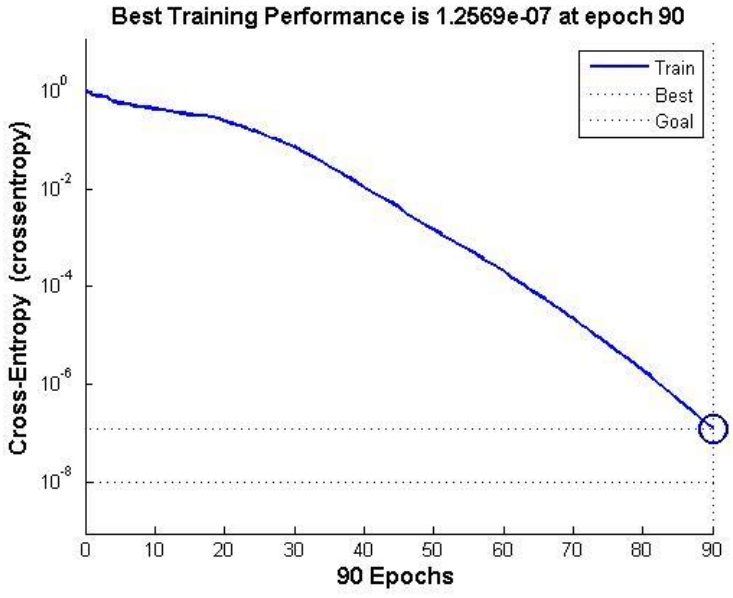
Q3.

My matric number is A0191818W, mod (18, 3) =0 (cat vs. airplane).

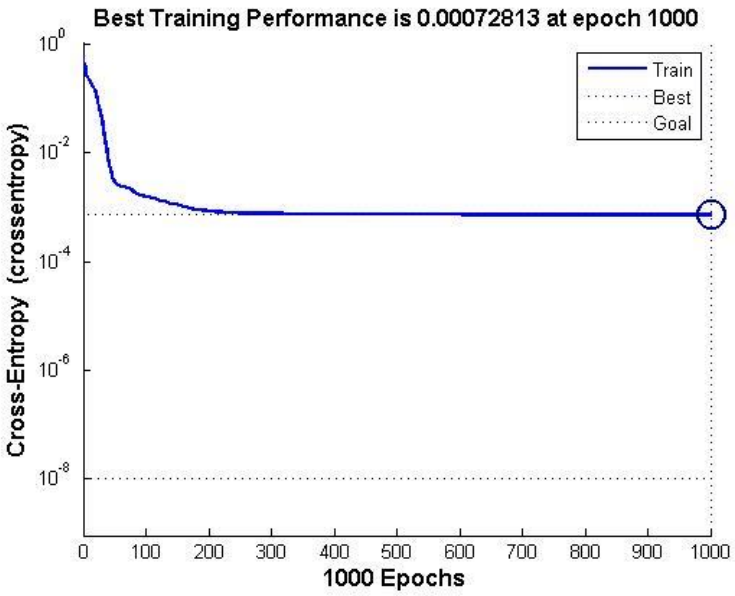
a) Apply Rosenblatt's perceptron (single layer perceptron) to the dataset of my assigned group 0. After the training procedure, the classification accuracy for both the training set and validation set are shown below. (the number of epochs:100)
The classification accuracy for the training set: **80.45%**
The classification accuracy for the validation set: **63%**
From the data, we could find that the classification accuracy for the training set is almost 80%, but the average value of the classification accuracy for the training set is 63%, which means that the performance of the network about the training set is quite well and the performance of the network about the validation set is not enough for well classification.

b) For global mean and variance method, the classification accuracy for both the training set and validation set are shown below.
The mean value = **0.5095**; The variance value = **0.0604**;
The classification accuracy for the training set: **86.56%**
The classification accuracy for the validation set: **65%**
From the data, we could find that the accuracy has been improved a little with the help of preprocessing.
Since the preprocessing could eliminate the effects of noise. Because of that, the calculation could run without the other noise.

c) Apply MLP perceptron (batch mode training) to the dataset of my assigned group 0. After the training procedure, the classification accuracy for the training set is **100%** and the classification accuracy for the training set is **73%**.(with 100 hidden neurons)



d) Since the MATLAB has the limitation to prevent the overfitting, the trained MLP has not shown the property of overfitting.
Apply MLP perceptron (batch mode training) to the dataset of my assigned group 0. After the training procedure, the classification accuracy for the training set is **100%** and the classification accuracy for the training set is **68%**. It need more time to run the code but prevent the overfitting problem.



e) Apply MLP (sequential mode training) to the dataset of my assigned group 0. After the training procedure, the classification accuracy for the training set is **78.11%** and the classification accuracy for the testing set is **68%**.(with 50 hidden neurons and 1000 epochs)
Comparing the result to part c), in my opinion, the selection of batch mode or sequential mode depend on the conditions given. In this experiment, batch mode training is quite faster than the sequential mode training. However, the batch mode need more place for training comparing to the sequential mode.
As a result, when the data set is small and the memory place is enough for training, the batch mode training is preferred. When the data set is large and the time is enough for training, the sequential mode training is preferred.

f) As far as I am concerned, that getting more data for training or using more hidden neurons could help to improve the performance of my MLP. More training data could increase the accuracy of weights. More hidden neurons but not exceed the limit of overfitting could make the classification more accuracy.

The following is the MATLAB code of Rosenblatt's perceptron:

```
%%
%INPUT
clear all;
close all;
load('testing_0.mat');load('testing_1.mat');load('training_0.mat');lo
ad('training_1.mat');

x0 = ones(900,1);
x1 = cat(1,training_0,training_1);
x2 = im2double(x1);
x = cat(2,x0,x2);%input

rowrank = randperm(size(x, 1));
x = x(rowrank,:);

y1 = zeros(450,1);
y2 = ones(450,1);
Y = cat(1,y1,y2);%design output
Y = Y(rowrank,:);

g = 1;
eta = 0.01;%learning speed
w(:,g)=rand(1025,1); %weight
epoch = 100;
%%
%CACULATION
for n = 0:epoch
   for i = 1:900
       v(i) = x(i,:)*w(:,g);
       if (v(i)>=1)
          h(i)=1;
       else
          h(i)=0;
       end
       e(i) = Y(i)-h(i);
       w(:,g+1) = w(:,g)+(eta*e(i)*x(i,:))';
       g=g+1;
   end
end
%%
%EVALUATE THE PERFORMANCE OF THE NETWORK
%TRAINING SET TEST
num_accuracy_train = 0;
for j = 1:900
```

```
    test1(j)=x(j,:)*w(:,g);
    if  test1(j)>=0.8
        y(j) = 1.0;
    else
        y(j) = 0;
    end
    if (((Y(j)== 0) && (y(j)== 0))||((Y(j)== 1.0) && (y(j)==1.0)))
        num_accuracy_train = num_accuracy_train +1;
    end
end

%VALIDATION SET TEST
num_accuracy_test = 0;
x0_test = ones(100,1);
x1_test = cat(1,testing_0,testing_1);
x2_test = im2double(x1_test);
x_test = cat(2,x0_test,x2_test);%test input
y1_test = zeros(50,1);
y2_test = ones(50,1);
Y_test = cat(1,y1_test,y2_test);%design test output
for j = 1:100
    test2(j)=x_test(j,:)*w(:,g);
    if  test2(j)>=0.8
        y_test(j) = 1.0;
    else
        y_test(j) = 0;
    end
    if (((Y_test(j)== 0) && (y_test(j)== 0))||((Y_test(j)== 1.0) &&
(y_test(j)==1.0)))
        num_accuracy_test  = num_accuracy_test +1;
    end
end
```

The following is the MATLAB code of means and variance:

```
%%
%INPUT
clear all;
close all;
load('testing_0.mat');load('testing_1.mat');load('training_0.mat');lo
ad('training_1.mat');

x0 = ones(900,1);
x1 = cat(1,training_0,training_1);
x2 = im2double(x1);
x = cat(2,x0,x2);%input
rowrank = randperm(size(x, 1));
x = x(rowrank,:);

y1 = zeros(450,1);
y2 = ones(450,1);
Y = cat(1,y1,y2);%design output
Y = Y(rowrank,:);

g = 1;
eta = 0.01;%learning speed
w(:,g)=rand(1025,1); %weight
epoch = 150;
%%
%PREPROCESS
X = mean(x(:));
V = std2(x)^2;
x=(x-X)./V;
%%
%CACULATION
for n = 0:epoch
   for i = 1:900
       v(i) = x(i,:)*w(:,g);
       if (v(i)>=1)
          h(i)=1;
       else
          h(i)=0;
       end
       e(i) = Y(i)-h(i);
       w(:,g+1) = w(:,g)+(eta*e(i)*x(i,:))';
       g=g+1;
   end
```

```
end
%%
%EVALUATE THE PERFORMANCE OF THE NETWORK
%TRAINING SET TEST
num_accuracy_train = 0;
for j = 1:900
    test1(j)=x(j,:)*w(:,g);
    if  test1(j)>=0.8
        y(j) = 1.0;
    else
        y(j) = 0;
    end
    if (((Y(j)== 0) && (y(j)== 0))||((Y(j)== 1.0) && (y(j)==1.0)))
        num_accuracy_train = num_accuracy_train +1;
    end
end

%VALIDATION SET TEST
num_accuracy_test = 0;
x0_test = ones(100,1);
x1_test = cat(1,testing_0,testing_1);
x2_test = im2double(x1_test);
x_test = cat(2,x0_test,x2_test);%test input
x_test=(x_test-X)./V;

y1_test = zeros(50,1);
y2_test = ones(50,1);
Y_test = cat(1,y1_test,y2_test);%design test output
for j = 1:100
    test2(j)=x_test(j,:)*w(:,g);
    if  test2(j)>=0.8
        y_test(j) = 1.0;
    else
        y_test(j) = 0;
    end
    if (((Y_test(j)== 0) && (y_test(j)== 0))||((Y_test(j)== 1.0) &&
(y_test(j)==1.0)))
        num_accuracy_test  = num_accuracy_test +1;
    end
end
```

The following is the MATLAB code of MLP using batch mode training:

```
%%
%INPUT
clear all;
close all;
load('testing_0.mat');load('testing_1.mat');load('training_0.mat');lo
ad('training_1.mat');
i=100;

x1 = cat(1,training_0,training_1);
x2 = im2double(x1);

y1 = zeros(450,1);
y2 = ones(450,1);
Y = cat(1,y1,y2);%design output

x1_test = cat(1,testing_0,testing_1);
x2_test = im2double(x1_test);

y1_test = zeros(50,1);
y2_test = ones(50,1);
Y_test = cat(1,y1_test,y2_test);%design test output
%%
%CACULATION
net = patternnet(i);
net.layers{1}.transferFCn='tansig';% set hidden layer 'logsig'
net.layers{2}.transferFCn='logsig';% set output layer 'logsig'
% net.performParam.regularization = 0.5;
net.trainparam.lr=0.01;
net.trainparam.epochs=1000;
net.trainparam.goal=1e-8;
net.divideParam.trainRatio=0.01;
net.divideParam.valRatio=0.0;
net.divideParam.testRatio=0.0;
[net,tr] = train(net,x2',Y');
plotperform(tr);
net_output = sim(net,x2');
%%
%EVALUATE THE PERFORMANCE OF THE NETWORK
%TRAINING SET TEST
num_accuracy_train = 0;
for j = 1:900
    if net_output(j) >=0.8
```

```
        y(j) = 1.0;
    else
        y(j) = 0;
    end
    if (((Y(j)== 0) && (y(j)== 0))||((Y(j)== 1.0) && (y(j)==1.0)))
        num_accuracy_train = num_accuracy_train +1;
    end
end


%VALIDATION SET TEST
num_accuracy_test = 0;
x0_test = ones(100,1);
x1_test = cat(1,testing_0,testing_1);
x2_test = im2double(x1_test);
x_test = cat(2,x0_test,x2_test);%test input
y1_test = zeros(50,1);
y2_test = ones(50,1);
Y_test = cat(1,y1_test,y2_test);%design test output
net_output2 = sim(net,x2_test');

for j = 1:100
    if  net_output2(j)>=0.8
        y_test(j) = 1.0;
    else
        y_test(j) = 0;
    end
    if (((Y_test(j)== 0) && (y_test(j)== 0))||((Y_test(j)== 1.0) &&
(y_test(j)==1.0)))
        num_accuracy_test  = num_accuracy_test +1;
    end
end
```

The following is the MATLAB code of MLP using sequential mode training:

```
%%
%INPUT
clear all;
close all;
load('testing_0.mat');load('testing_1.mat');load('training_0.mat');lo
ad('training_1.mat');
i=50;

x1 = cat(1,training_0,training_1);
x2 = im2double(x1);

y1 = zeros(450,1);
y2 = ones(450,1);
Y = cat(1,y1,y2);%design output

x1_test = cat(1,testing_0,testing_1);
x2_test = im2double(x1_test);

y1_test = zeros(50,1);
y2_test = ones(50,1);
Y_test = cat(1,y1_test,y2_test);%design test output
%%
%CACULATION
net = patternnet(i);
net.layers{1}.transferFCn='tansig';% set hidden layer 'logsig'
net.layers{2}.transferFCn='logsig';% set output layer 'logsig'
% net.performParam.regularization = 0.5;
net.trainparam.lr=0.01;
net.trainparam.epochs=1000;
net.trainparam.goal=1e-8;
net.divideParam.trainRatio=0.01;
net.divideParam.valRatio=0.0;
net.divideParam.testRatio=0.0;
% [net,tr]=adapt(net,x2',Y');
for i = 1 : 1000
    index = randperm(size(x2, 1));
    net = adapt(net,x2(index,:)',Y(index,:)');
end

% [net,tr] = train(net,x2',Y');
```

```
% plotperform(tr);
net_output = sim(net,x2');
%%
%EVALUATE THE PERFORMANCE OF THE NETWORK
%TRAINING SET TEST
num_accuracy_train = 0;
for j = 1:900
    if net_output(j) >=0.8
        y(j) = 1.0;
    else
        y(j) = 0;
    end
    if (((Y(j)== 0) && (y(j)== 0))||((Y(j)== 1.0) && (y(j)==1.0)))
        num_accuracy_train = num_accuracy_train +1;
    end
end


%VALIDATION SET TEST
num_accuracy_test = 0;
x0_test = ones(100,1);
x1_test = cat(1,testing_0,testing_1);
x2_test = im2double(x1_test);
x_test = cat(2,x0_test,x2_test);%test input
y1_test = zeros(50,1);
y2_test = ones(50,1);
Y_test = cat(1,y1_test,y2_test);%design test output
net_output2 = sim(net,x2_test');

for j = 1:100
    if  net_output2(j)>=0.8
        y_test(j) = 1.0;
    else
        y_test(j) = 0;
    end
    if (((Y_test(j)== 0) && (y_test(j)== 0))||((Y_test(j)== 1.0) &&
(y_test(j)==1.0)))
        num_accuracy_test  = num_accuracy_test +1;
    end
end
```