CE793: Deep Learning for Engineers
Assignment 4

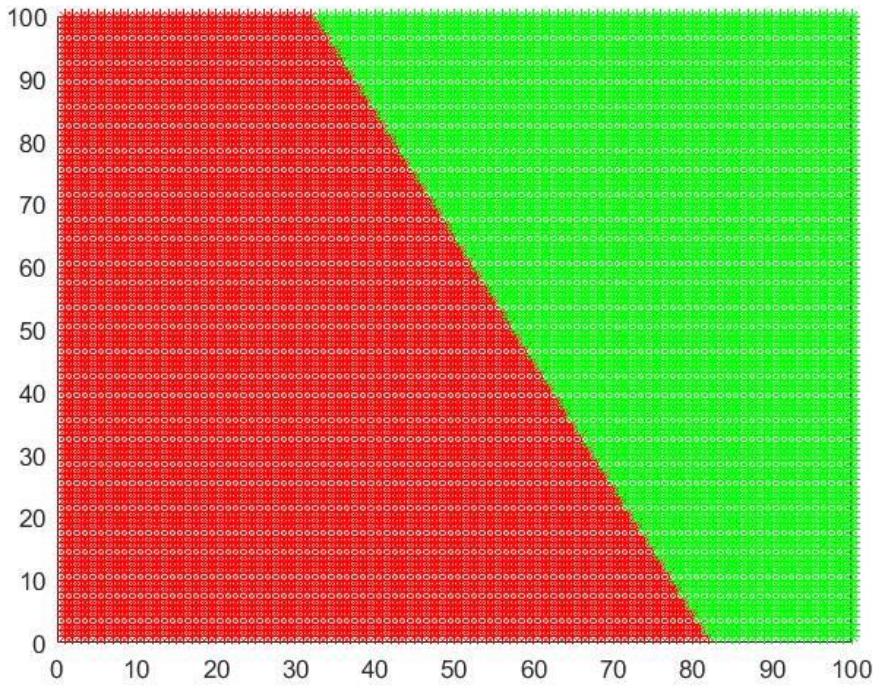
Dharanidharan Arumugam

➢ 1.Single Layer Perceptron:

- Number of Epochs: 977
- Accuracy: 100%
- Confusion Matrix:

$$\begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$$

- Decision Surface



- MATLAB Code

```
clc;clearvars;close all;
infile      = 'Perceptron_single_target.xlsx';

datatable   = readtable(infile);
headers     = datatable.Properties.VariableNames; headers(:,end)=[];
inputs      = datatable.Variables; clear datatable;
target      = inputs(:,end); inputs(:,end)=[];
[no_of_instances,no_of_features]     = size(inputs);
inputs      = [ones(no_of_instances,1),inputs];

weights     = zeros(no_of_features+1,1);
max_epoch       = 1000;
target_accuracy = 95;
total_classifications = no_of_instances;
```

```
for epoch = 1:max_epoch

  for instance_number = 1:no_of_instances

      instance_vector = inputs(instance_number,:);
      z  = instance_vector*weights;
      y_predicted = signum_activation(z);
      y_actual    = target(instance_number);
      error       = y_actual - y_predicted;
      weights     = weights + error*instance_vector';

  end

      Y_predicted = signum_activation(inputs*weights);
      error_all   = target-Y_predicted;
      misclassifieds = sum(abs(error_all));
      accuracy = (total_classifications-misclassifieds)/total_classifications*100;
      if accuracy >= target_accuracy
           break;
      end

end

confusion_matrix=zeros(2);

for instance_number = 1:no_of_instances
      y_predicted = Y_predicted(instance_number);
      y_actual    = target(instance_number);

      m = y_predicted+1; n = y_actual+1;
      confusion_matrix(m,n)=confusion_matrix(m,n)+1;
end

confusion_matrix,accuracy, epoch


for i=1:100
    for j=1:100
        instance_vector = [1 i j];
        z = dot(weights,instance_vector);
        if z<0
           plot(i,j,'*r')
        else
           plot(i,j,'*g')
        end
        hold on
    end
end

function neuron_outputs = signum_activation(neuron_inputs)

    [no_of_neurons,no_of_instances] = size(neuron_inputs);
    neuron_outputs  = ones(no_of_neurons,no_of_instances);
    neuron_outputs(neuron_inputs < 0)= 0;

end
```

CE793: Deep Learning for Engineers
Assignment 4

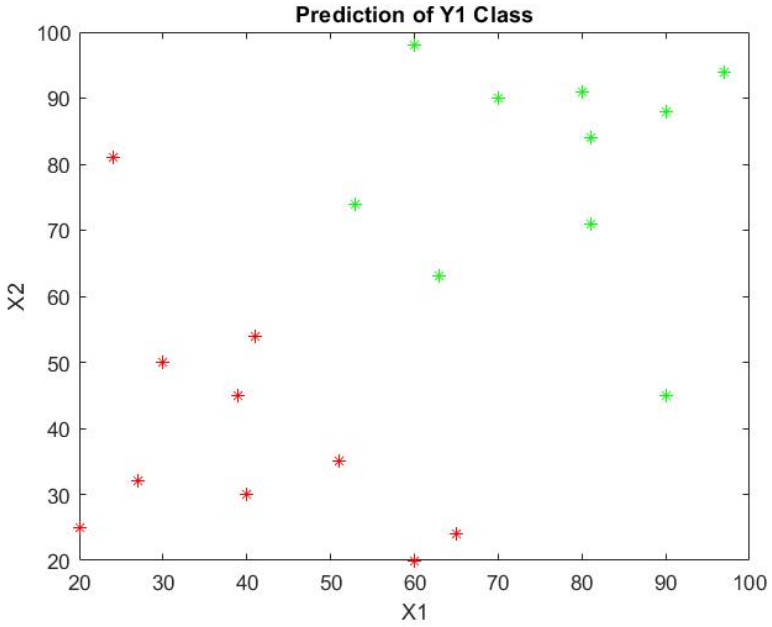
Dharanidharan Arumugam

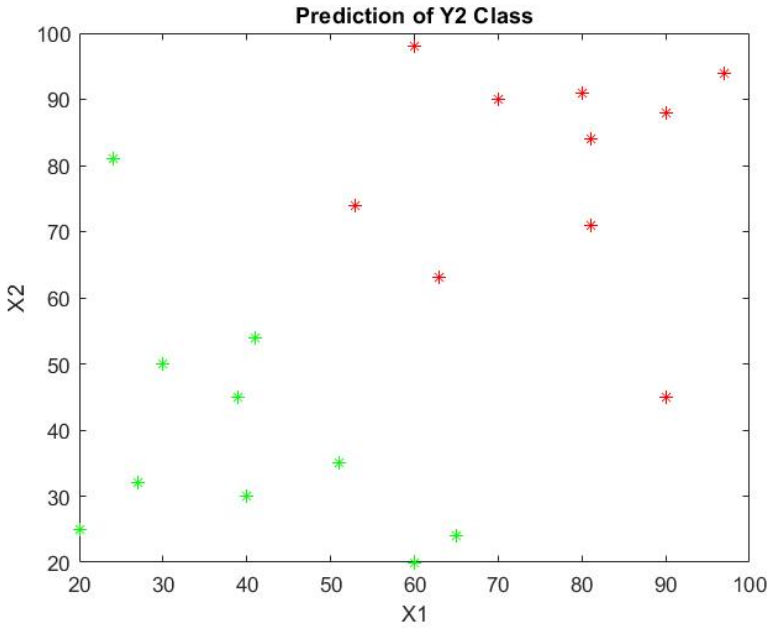
➢ 2.Multi-Layer Perceptron:

- Number of Epochs: 6
- Accuracy: 92.5%
- Confusion Matrix:

$$\begin{bmatrix} 20 & 0 \\ 3 & 17 \end{bmatrix}$$

- Class Prediction Plots:





Dharanidharan Arumugam

- MATLAB Codes:

```
clc;clearvars;close all;
infile      = 'Perceptron_multi_target.xlsx';
no_of_targets = 2;
datatable   = readtable(infile);
headers     = datatable.Properties.VariableNames; headers(:,end)=[];
inputs      = datatable.Variables; clear datatable;
targets     = inputs(:,end-no_of_targets+1:end);
inputs(:,end-no_of_targets+1:end)=[];
[no_of_instances,no_of_features]     = size(inputs);
inputs_normalized = (inputs-mean(inputs))./std(inputs);
inputs_normalized = [ones(no_of_instances,1),inputs_normalized];
weights     = zeros(no_of_features+1,no_of_targets);
learning_rate   = 0.01;
max_epoch       = 10000;
target_accuracy = 90;
total_classifications = no_of_instances*no_of_targets;

for epoch = 1:max_epoch
    for instance_number = 1:no_of_instances
        instance_vector = inputs_normalized(instance_number,:);
        z           = instance_vector*weights;
        y_predicted = signum_activation(z);
        y_actual    = targets(instance_number,:);
        error       = y_actual - y_predicted;
        weights     = weights+learning_rate*(transpose(instance_vector)*error);
    end
        Y_predicted = signum_activation(inputs_normalized*weights);
        error_all   = targets-Y_predicted;
        misclassifieds = 0.5*sum(sum(abs(error_all)));
        accuracy = (total_classifications-
misclassifieds)/total_classifications*100;
        if accuracy >= target_accuracy
            break;
        end
end
confusion_matrix = zeros(2);

for target_number = 1:no_of_targets
    for instance_number=1:no_of_instances
       y_predicted = Y_predicted(instance_number,target_number);
       y_actual = targets(instance_number,target_number);
       m = 0.5*(y_predicted+1)+1; n = 0.5*(y_actual+1)+1;
       confusion_matrix(m,n)=confusion_matrix(m,n)+1;
       x1=inputs(instance_number,1);x2=inputs(instance_number,2);
       if y_predicted==-1
           plot(x1,x2,'*r')
       else
           plot(x1,x2,'*g')
       end
       hold on
    end
    figure;
end
```

```
function neuron_outputs = signum_activation(neuron_inputs)

    [no_of_neurons,no_of_instances] = size(neuron_inputs);
    neuron_outputs  = ones(no_of_neurons,no_of_instances);
    neuron_outputs(neuron_inputs < 0)= -1;

end
```