* **1.Single Layer Perceptron:**
* Number of Epochs: 977
* Accuracy: 100%
* Confusion Matrix:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 10 | 0 |  |
|  | 0 | 10 |  |

* Decision Surface

Chart

Description automatically generated

* 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);

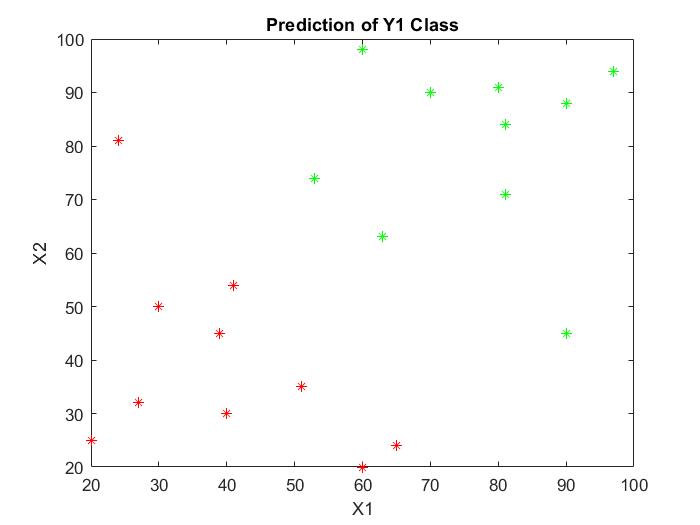
neuron\_outputs(neuron\_inputs < 0)= 0;

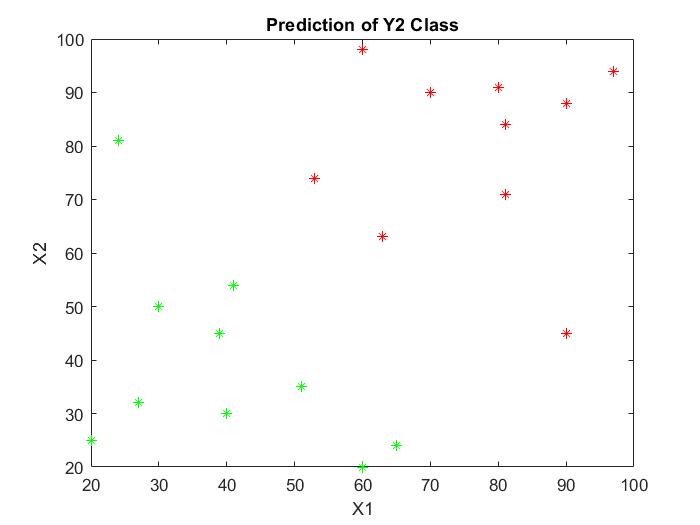
end

* **2.Multi-Layer Perceptron:**
* Number of Epochs: 6
* Accuracy: 92.5%
* Confusion Matrix:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 20 | 0 |  |
|  | 3 | 17 |  |

* Class Prediction Plots:





* 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