Onur Poyraz
2010401036
30 March 2015

Homework II

In this project, I deal with the Perceptron Learning Model. It is asked from us that we should implement 3-D perceptron model with single neuron. Which means that we should find a decision plane which separate class A and class B input from each other. In this project class A inputs are in the 1st quadrant and the class B inputs are in the 8th quadrant. I use 50 sample vectors in this project for the training set. I used Matlab to implement the algorithm and to take an graphical output.

I begin the project with assigning an initial weights with a random generator between absolute value of 0.1 and 0.2. This weights are small enough and randomly selected.

```
w=zeros(1,3);
for i=1:3
    w(i)=(rand+1)/10;
end
```

After construction of initial weights I define my sampler patterns. I pick 50 sampler patterns which are selected as 25 of it is in the first quadrant and 25 of it is in the eighth quadrant. Each entry of the each sampler pattern is randomly selected. In here I use the simple rand function because only important thing is that the quadrants of the inputs.And here I define the desired output for each sampler pattern.

```
for i=1:2:49
    for j=1:3
        X(i,j)=rand;                //Creation of class A sampler pattern
        x(i,j)=abs(X(i,j));
        A((i+1)/2,j)=x(i,j);
        d(i)=1;

        X(i+1,j)=rand;              //Creation of class B sampler pattern
        x(i+1,j)=-abs(X(i+1,j));
        B((i+1)/2,j)=x(i+1,j);
        d(i+1)=-1;
    end
end
```

After the definition of the sampler vectors I implement the algorithm that we discussed on it in the class. In here first of all I find the summation of w_i*x_i multiplication and I insert it to the hard-limiter. In here I define threshold '0'. Doing this I find the actual output of my single neuron perceptron model for given sampler vector. After that I find the error function subtraction actual output from the desired output for this sampler pattern. After finding of error function I update the weight matrix according to formula that given in class. In here, I select learning factor as a 1/length(sample patterns) which gives a usable small

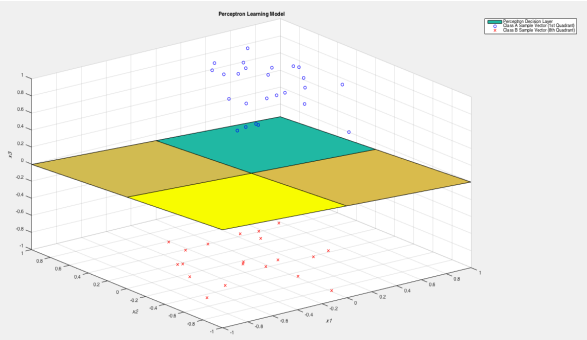
value. I make this whole story for 50 sample vectors that I define in the beginning of the project. After the first impact(going on the sample set once) I record the weights and I start the new impact. While the old weights not equal to the new weights I do that.

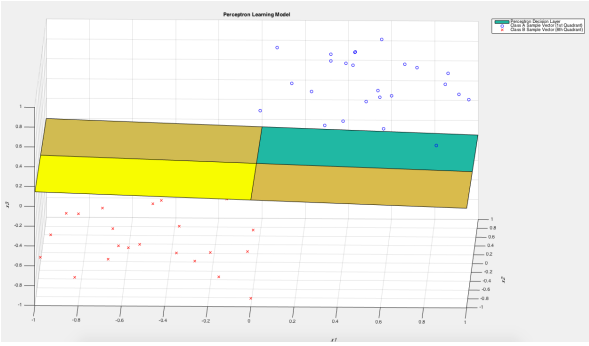
```
while (1)
y=zeros(50,1);          //initialization
Y=zeros(50,1);
e=zeros(50,1);
lf=1/length(d);
w_old=w;               //definiton of old weights
    for i=1:50
        for j=1:3
            Y(i,1)=Y(i,1)+x(i,j).*w(j);  //computation of output
        end
        if Y(i,1)>0                 //hard-limiter
            y(i,1)=1;
        else
            y(i,1)=-1;
        end
        e(i,1)=d(i,1)-y(i,1);       //calculation of error function
        for j=1:3
            w(j)=w(j)+ lf.*(e(i).*x(i,j));   //update of the weights
        end
    end
    if w==w_old   //control part of the new impact is necessary or not
        break;
    end
end
```

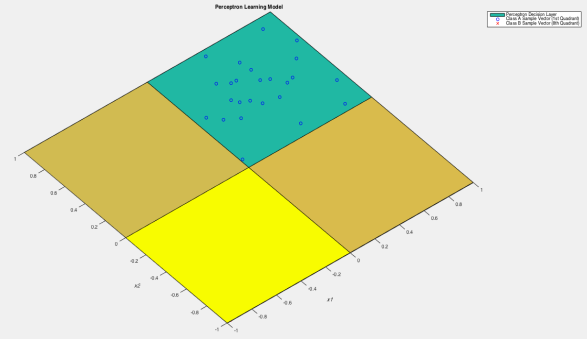
After completion of the update of the weight matrix I define the decision plane. It is obvious that my weight matrix is the normal of the my decision plane and my decision plane is going on the (0,0,0) point. So I can build my decision plane. In this part I build and plot the decision plane in 3-D graph and I plot each of my sample vector and I saw that my decision plane separate the class A inputs and class B inputs from each other.

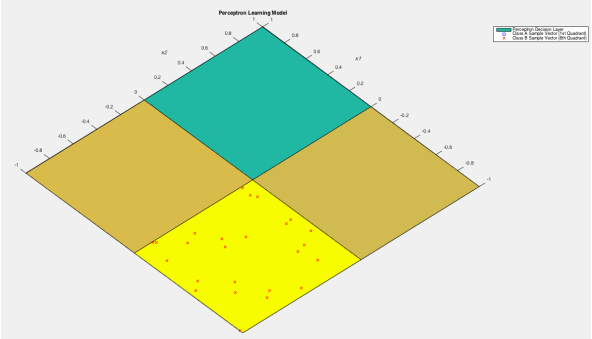
```
normal=w
[xx,yy]=ndgrid(-1:1,-1:1);
z=(-normal(1)*xx - normal(2)*yy)/normal(3);
figure
surf(xx,yy,z)
hold on;
plot3(A(:,1),A(:,2),A(:,3),'bo');
hold on;
plot3(B(:,1),B(:,2),B(:,3),'rx');
hold on;
xlabel('\it x1');
ylabel('\it x2');
zlabel('\it x3');
title('Perceptron Learning Model');
legend('Perceptron Decision Layer','Class A Sample Vector (1st
Quadrant)','Class B Sample Vector (8th Quadrant)')
```

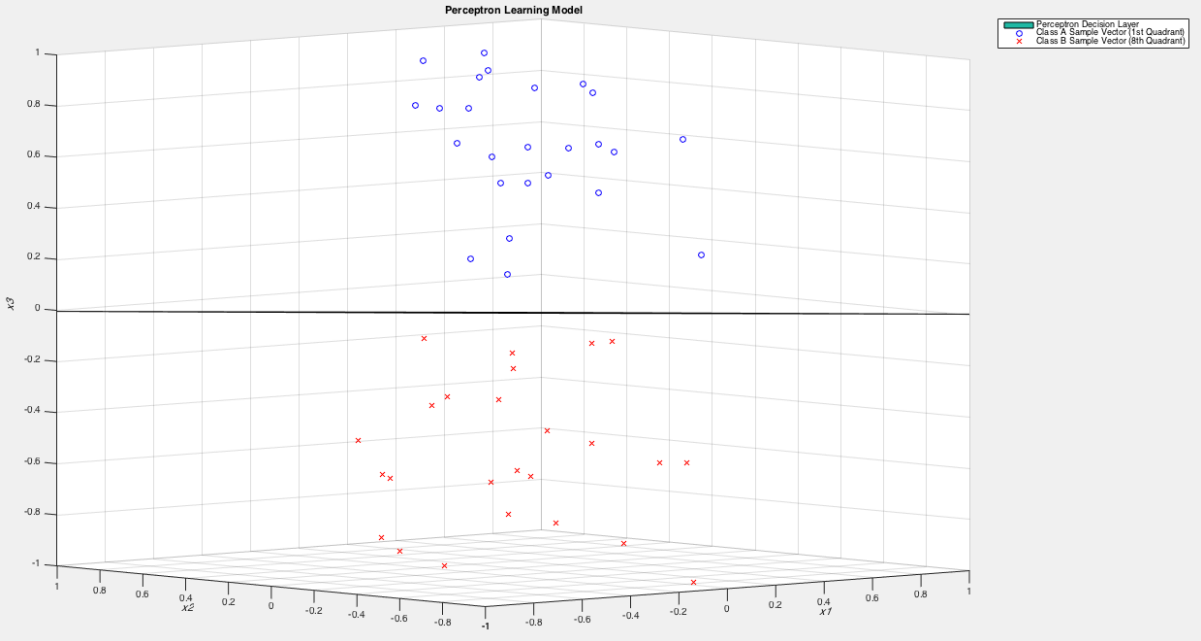
Now, I will put the output of my function with different aspects. Because the output is 3-D it is hard to show on paper. The blue ‘o’s represent the class A sample vectors and the red ‘x’s represent the class B sample vectors.


Perceptron Learning Model


Perceptron Learning Model


Perceptron Learning Model


Perceptron Learning Model


Perceptron Learning Model
Perceptron Decision Layer
Class A Sample Vector (1st Quadrant)
Class B Sample Vector (8th Quadrant)
x3
x2
x1