Optimization Methods for Machine Learning: Homework #1

Due on Monday, October 24, 2016

Laura Palagi - Ing. Umberto Dellepiane

Alessandro Gallo

Contents

Instructions **3**

Question 1 **3**

1.1 Report the values of the dataset . . . 3
1.2 Initial line of the perceptrons and misclassified samples . . . 3
1.3 New line of division and new number of misclassified samples . . . 5
1.4 Repeat process until having six lines (achieves perfect classification) . . . 7

Question 2 **9**

2.4 Complete implement of the simple perceptron and steps until perfect classification . . . 9
2.5 Averaged perceptron algorithm, iterations to get a separating hyperplane and algorithm performance . . . 11
2.6 Change the labels of some points. Average perceptron will never converge . . . 15

Question 3 **18**

3.7 Average quadratic error and minimization of E(W,b) . . . 18

Instructions

In the folder HW1 there are the files .m for every solved question of the homework (q11.m, q12.m, q13.m, q14.m, q24.m, q25.m, q26.m, q37a.m, q37b.m) + the commands file .m (commands.m) to run them. In this report you can find, for every question, exctatly the input commands to run the functions that solve the questions,the functions itselfs with every line commented, the outputs and the plots.

Question 1

1.1 Report the values of the dataset

The sample is contained in a txt file called sample.txt and it's imported in the Matlab environment.

File q11.m / Command Window

```
>> file='sample.txt';      % file txt containing the dataset
data=importdata(file); % importing the dataset
```

Output:

```
data =

    0.1500    0.7500    1.0000
    0.2000    0.4000    1.0000
    0.4500    0.6500    1.0000
    0.6500    0.1500    1.0000
    0.6500    0.9500   -1.0000
    0.8500    1.0000   -1.0000
    0.9500    0.5500   -1.0000
    1.0500    0.2500   -1.0000
```

1.2 Initial line of the perceptrons and misclassified samples

We initialiaze the parameters for the weights: a weights vector for w_1 and w_2 and the b weight. In the function we order the dataset in case the length is > 8 or rows are unsorted (based on the 3 column). We write the function of the linear separating hyperplane and apply the first *sign* activation function to know the mislclassified points.

Command Window

```
>> file='sample.txt';
w=[1 -1];                  % weights vector of w1 and w2
b=0.2;                     % weight value of b
[mis1] = q12(file,b,w)     % output: n misclassified points
```

File q12.m

```
function [mis1] = q12(file,b,w) % Inputs/Outputs
data=importdata(file);          % importing the dataset
[~,N]=size(data);               % {  if the values (+1,-1) of the 3Âř column
N=N-1;                          %    are mixed, sort their corresponding
so=sortrows(data,3);            %    rows (having ordered -1 and +1) }
[r,~]=find(so(:,N+1)==1);       % {  find the value where we have the
```

```
r=r(1);                             %    the first row with value +1 in 3Âř col. }
blacks1=so(1:r-1,1);                % assign the x1's of -1 rows
blacks2=so(1:r-1,2);                % assign the x2's of -1 rows
whites1=so(r:end,1);                % assign the x1's of +1 rows
whites2=so(r:end,2);                % assign the x2's of +1 rows
figure                              %
scatter(whites1,whites2,'r','filled')       % { plot the white
hold on                                     %   and black points
scatter(blacks1,blacks2,'b','filled')       %   (+1 and -1 points)
xlabel('x_{1}')                             %
ylabel('x_{2}')                             %
legend('+1','-1','Location','southwest')    %  }
title('Perceptrons inital line of division')%
x=[0 1];                                    % { plot the initial line
plot(x,(-b-w(1)*x)/w(2),'k')                %  }
f=b+w(1)*data(:,1)+w(2)*data(:,2);          % define linear separating hyperplane
yhat=[];                          % { for i from 1 to the length of the data,
for i=1:length(data)              %   calculate the value in the hyperplane
    if f(i)>=0                    %   equation. If f(i)>0, assign +1 in the
        yhat(i)=1;                %   vector of yhat, otherwise -1
    else                          %
        yhat(i)=-1;               %
    end                           %
end                               %  }
yhat=(yhat)';                     % transpose yhat
A=[yhat==data(:,3)];              % boolean check with 'data' of the +1 and -1 yhat values
k=find(A~=1);                     % location where they are different
mis1=length(k);                   % number of misclassified samples
```

Output:

```
mis1 =

     4
```

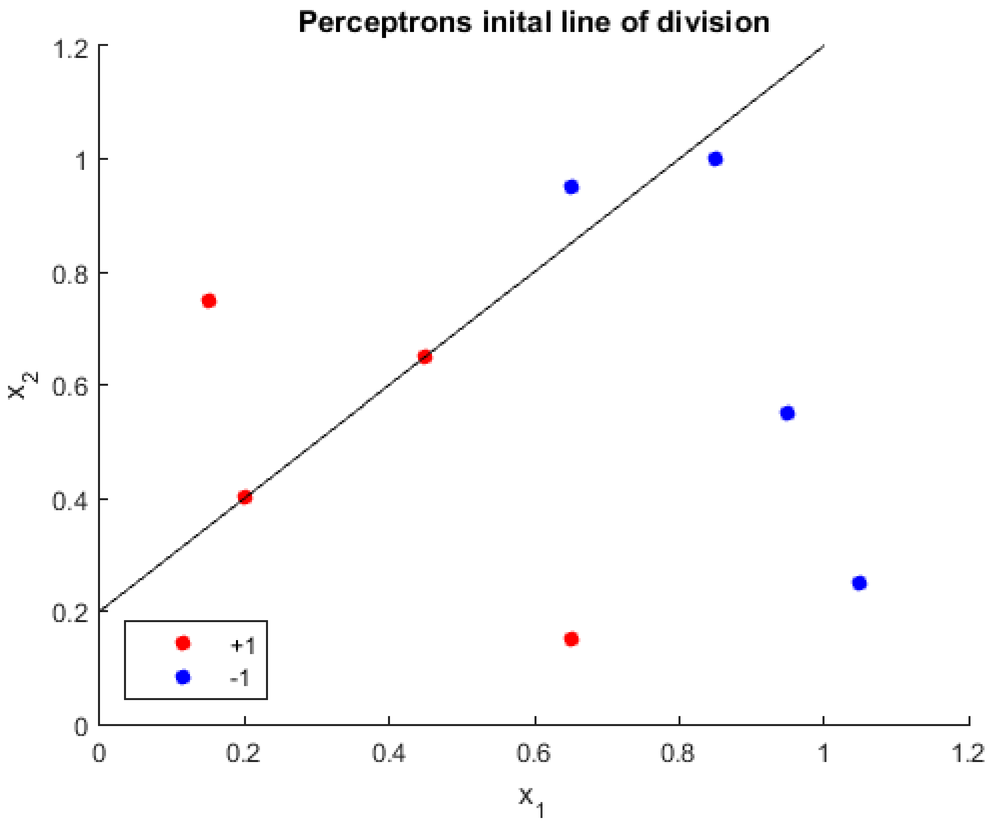

Figure 1: Initial line of division of q12.m

1.3 New line of division and new number of misclassified samples

In this case we make the first full iteration of the perceptron updating the weights values (under condition), obtaining a new line of division and new number of misclassified points.

Command window

```
>> file='sample.txt';
w=[1 -1];
b=0.2;
[mis2]=q13(file,b,w)            % output: new nÂř misclassified points
```

File q13.m

```
function [mis2]=q13(file,b,w)
data=importdata(file);                % {

[~,N]=size(data);
N=N-1;
so=sortrows(data,3);
[r,~]=find(so(:,N+1)==1);
r=r(1);
blacks1=so(1:r-1,1);                  %          See q12.m
blacks2=so(1:r-1,2);
whites1=so(r:end,1);
```

```
whites2=so(r:end,2);
figure
scatter(whites1,whites2,'r','filled')
hold on
scatter(blacks1,blacks2,'b','filled')
xlabel('x_{1}')
ylabel('x_{2}')
legend('+1','-1','Location','southwest')
title('New line of division')
x=[0 1];
plot(x,(-b-w(1)*x)/w(2),'k')
f=b+w(1)*data(:,1)+w(2)*data(:,2);       %  }

for i = 1:length(data)                   % { for 1:length(data),
    if (data(i,3)*f(i))<=0               %   if (data(i,3)*f(i))<=0,
        w(1)=w(1)+data(i,3)*data(i,1);   %   update the weights vector
        w(2)=w(2)+data(i,3)*data(i,2);   %   of w1 and w2 and the weigth
        b=b+data(i,3);                   %   value of b
    end                                  %
end                                      %  }
yhat=[];                                 % {
for i=1:length(data)                     %
if f(i)>=0                               %
   yhat(i)=1;                            %
else                                     %
   yhat(i)=-1;                           %
end                                      %
end                                      %   Plot the new line and
yhat=yhat';                              %   and returns the new number of
A=[];                                    %   of misclassified samples
A=[yhat==data(:,3)];                     %
k=find(A~=1);                            %
hold on                                  %
plot(x,(-b-w(1)*x)/w(2),'k')             %
mis2=length(k);                           % }
end
```

Output:

```
mis2 =

     4
```

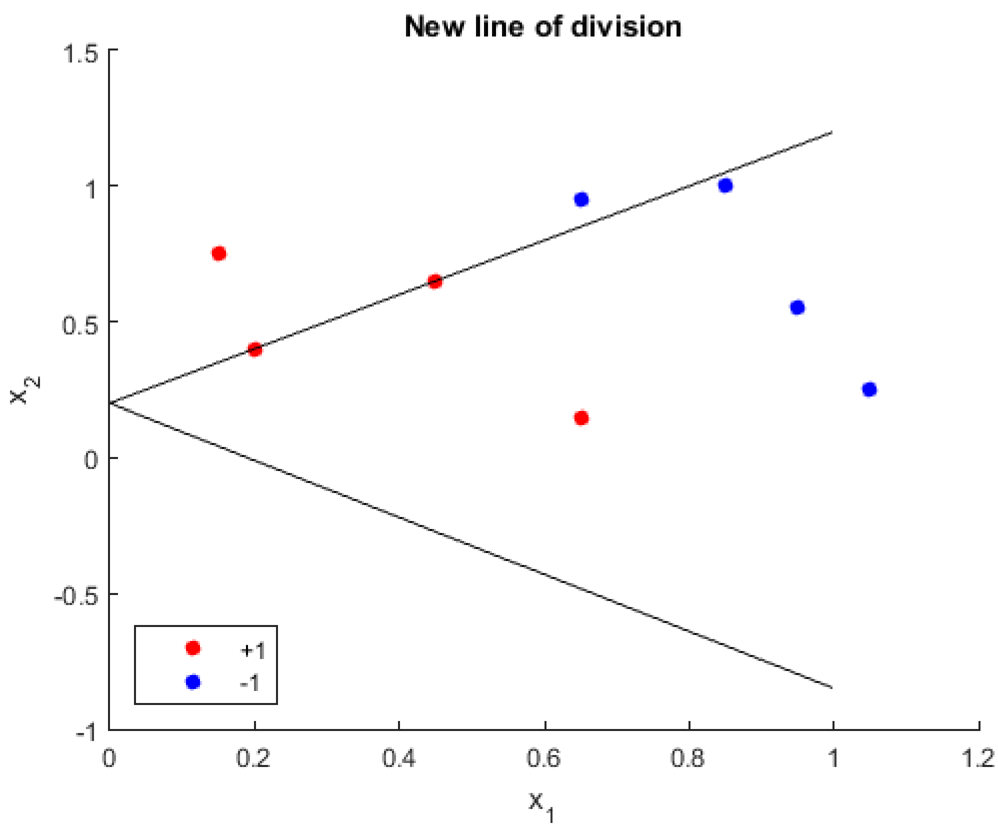

Figure 2: New line of division of q13.m

1.4 Repeat process until having six lines (achieves perfect classification)

We call the function q13.m and continue the iterations until we reach a total of six lines and the number of misclassified points is zero

Command window

```
>> file='sample.txt';
w=[1 -1];
b=0.2;
[mis3]=q14(file,b,w)             % output: final n misclassified points
```

```
function[mis3]=q14(file,b,w)                          %
[mis2]=q13(file,b,w);                                  % Recall function before in q13.m
data=importdata(file);                                 %    and add the new lines
cc=0;                                                  % initialize counter
l=length(data);                                        %
j=1;                                                   % initialize repeating process j
while cc < l || j==4                                   % { while cc < l or j==4
    for i = 1:length(data)                             %   for i = 1:l
        f(i)=b+w(1)*data(i,1)+w(2)*data(i,2);          %       if (data(i,3)*f)<=0
        if (data(i,3)*f(i))<=0                         %           update the weights vector
                w(1)=w(1)+data(i,3)*data(i,1);         %           of w1 and w2 and the weigth
                w(2)=w(2)+data(i,3)*data(i,2);         %           value of b
```

```
                b=b+data(i,3);                  %
        else                                    %       otherwise
            cc=cc+1;                            %           increase counter
        end                                     %
    end                                         %
    if cc<l                                     %       if cc<l
        cc=0;                                   %           set cc to 0
    end                                         %
    yhat=[];                                    %
    for i=1:8                                   % {
       if f(i)>=0
            yhat(i)=1;
        else
            yhat(i)=-1;
        end
    end                                         %       See q13.m
    yhat=yhat';
    A=[];
    A=[yhat==data(:,3)];
    k=find(A~=1);
    mis3=length(k);
    j=j+1;                                      % }
    hold on
    x=[0 1];                                    %       plot the new division line
    title('Perfect classification with 6 lines')
    plot(x,(-b-w(1)*x)/w(2),'k')                %       until there will be 6 lines }
end
end
```

Output:

```
mis3 =

     0
```

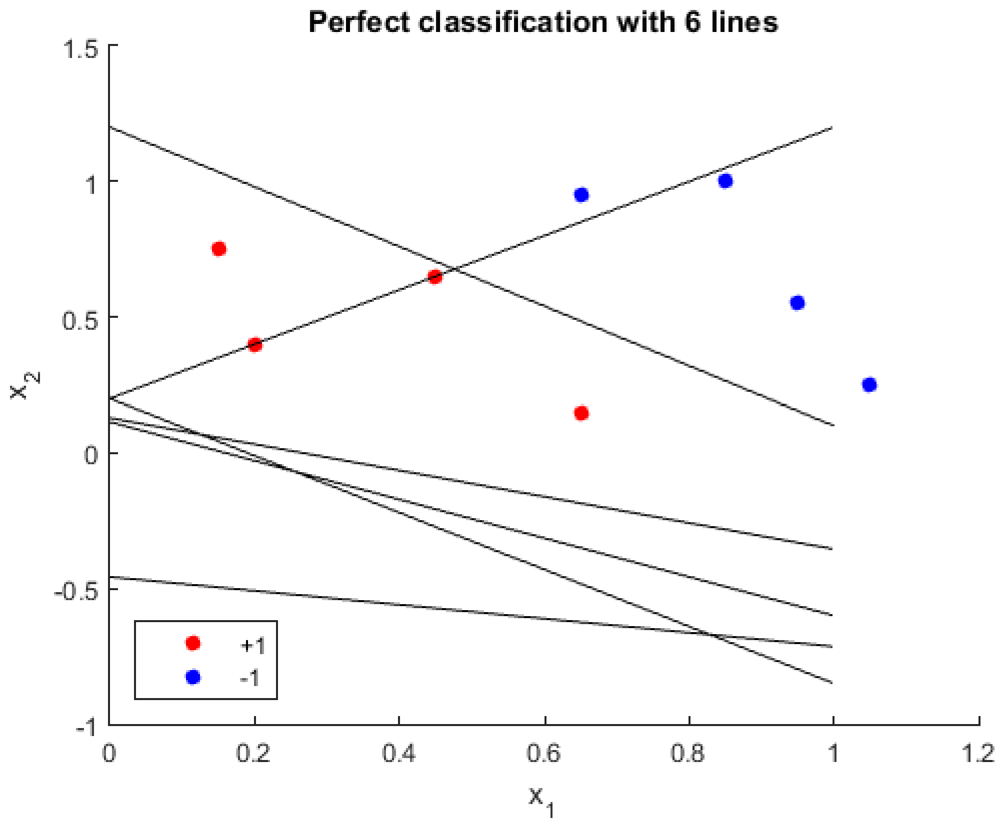

Figure 3: Six lines perceptrons of q14.m

Question 2

2.4 Complete implement of the simple perceptron and steps until perfect classification

The function is based on the exercises of question 1 (without a stopping condition) and, as we can expect, the number of steps is 5.

Command window

```
>> w=[1 -1];
b=0.2;
file='sample.txt';
[ steps ] = q24(file,b,w)        % output: steps to achive perfect classification
```

File q24.m

```
function [ steps ] = q24(file,b,w)            % Input: data, weights vector of
data=importdata(file);                        %    w1 and w2, and weight value of b
[~,N]=size(data);                             % Output: steps to let program
N=N-1;                                        %    achive perfect classification
so=sortrows(data,3);
[r,~]=find(so(:,N+1)==1);
```

```
r=r(1);
blacks1=so(1:r-1,1);
blacks2=so(1:r-1,2);
whites1=so(r:end,1);
whites2=so(r:end,2);                              %       See q12.m
figure
scatter(whites1,whites2,'r','filled')
hold on
scatter(blacks1,blacks2,'b','filled')
xlabel('x_{1}')
ylabel('x_{2}')
legend('+1','-1','Location','southwest')
x=[0 1];
plot(x,(-b-w(1)*x)/w(2),'k')
f=b+w(1)*data(:,1)+w(2)*data(:,2);                %

cc=0;                                             % {
l=length(data);
steps=1;
while cc < l
    for i = 1:l
        f(i)=b+w(1)*data(i,1)+w(2)*data(i,2);
        if (data(i,3)*f(i))<=0
                w(1)=w(1)+data(i,3)*data(i,1);
                w(2)=w(2)+data(i,3)*data(i,2); %       See q14.m
                b=b+data(i,3);
        else
            cc=cc+1;
        end
    end
    if cc<l
        cc=0;
    end                                           % }
    yhat=[];                                      % {
    for i=1:length(data)
       if f(i)>=0
            yhat(i)=1;
       else
            yhat(i)=-1;
       end
    end
    yhat=yhat';                                   %     See q13.m
    A=[yhat==data(:,3)];
    k=find(A~=1);
    n=length(k);
    %hold on
    %x=[0 1];
    %plot(x,(-b-w(1)*x)/w(2),'k')
    %axis([0 1.2 0 1.2])                          %  }
    if yhat==data(:,3)                            % { When the algorithm achives
        x=[0 1];                                  %   the right classification
        plot(x,(-b-w(1)*x)/w(2),'k')              %   plot the last division line
        axis([0 1.2 0 1.2])                       %
```

```
            break                                        %
        end                                              %  }
        steps=steps+1;                                   %  Increase steps number
end
end
```

Output:

```
steps =

     5
```

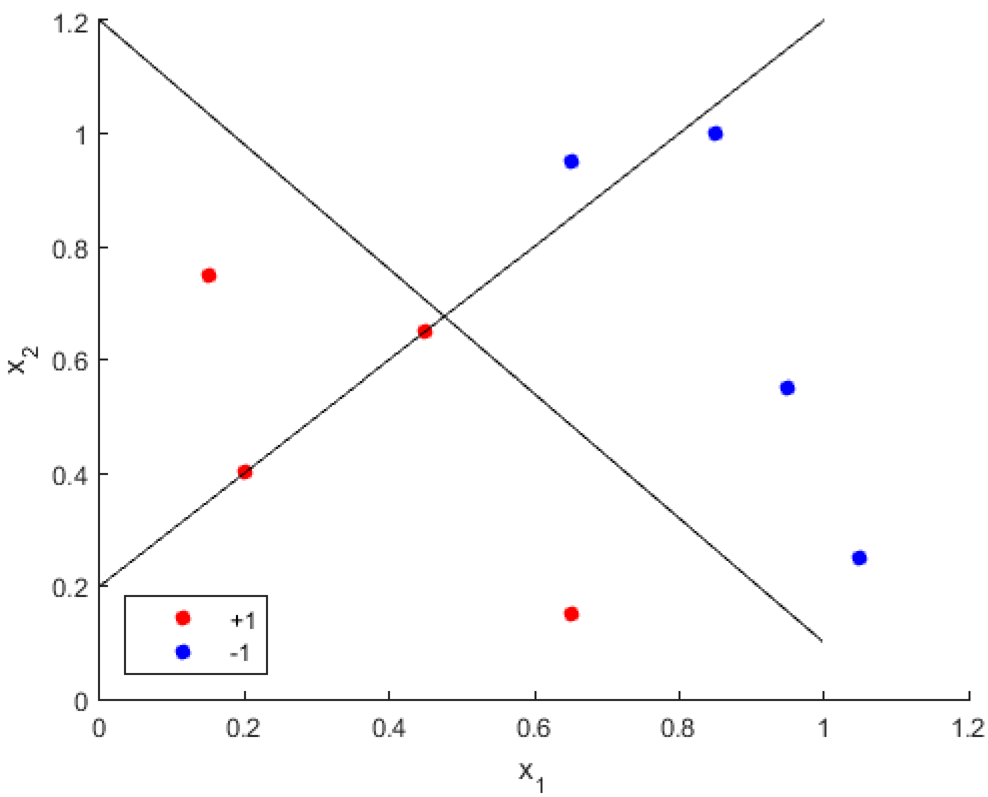

Figure 4: Final line division of q24.m

2.5 Averaged perceptron algorithm, iterations to get a separating hyperplane and algorithm performance

In the Averaged Perceptron we initiliaze also the average weights w_{avg} and a voting value v. The modified algorithm is shown below with final values of the w_{avg}, a) the number of iterations to reach convergence and b) the performance of the algorithm.

Command window

```
>> file='sample.txt';
w=[1 -1];
b=0.2;
```

```
maxiter=20;
wavg=[0 0];
bavg=0;
v=0;
its=0;
[wavg,bavg,iter]=q25(file,b,w,maxiter,wavg,bavg,v,its)
```

File q25.m

```
function [wavg,bavg,iter]=q25(file,b,w,maxiter,wavg,bavg,v,its) % {
data=importdata(file);
[~,N]=size(data);
N=N-1;
so=sortrows(data,3);
[r,~]=find(so(:,N+1)==1);
r=r(1);                                              %       See q12.m
blacks1=so(1:r-1,1);
blacks2=so(1:r-1,2);
whites1=so(r:end,1);
whites2=so(r:end,2);
figure
scatter(whites1,whites2,'r','filled')
hold on
scatter(blacks1,blacks2,'b','filled')
xlabel('x_{1}')
ylabel('x_{2}')
legend('+1','-1','Location','southwest')             % }

n=0;
y=0;
iter=0;
while its < maxiter                                     % { while its < maxiter
    for i = 1:length(data)                              %   for i = 1:length(data)
        f(i)=bavg+wavg(1)*data(i,1)+wavg(2)*data(i,2); %     New hyperplane equation
        if  (data(i,3)*f(i))<=0                         %     if  (data(i,3)*f(i))<=0
            wavg(1)=wavg(1)+v*w(1);                     %       update all the weights
            wavg(2)=wavg(2)+v*w(2);                     %       vectors wavg, bavg, w
            bavg=bavg+v*b;                              %       and b
            w(1)=w(1)+data(i,3)*data(i,1);              %
            w(2)=w(2)+data(i,3)*data(i,2);              %
            b=b+data(i,3);                              %
            v=1;                                        %  set value of v to 1
        else
            v=v+1;                                      %  increase value of v
        end                                             %
    end                                                 %  }
    h=bavg+wavg(1)*data(:,1)+wavg(2)*data(:,2);         %  vector of hyperplane equations
    yhat=[];                                          % {
    for i=1:length(data)
        if  h(i)>=0
            yhat(i)=1;
        else
            yhat(i)=-1;
        end
```

```
    end                                                  %        See q13.m
    yhat=yhat';
    A=[];
    A=[yhat==data(:,3)];
    k=find(A==1);
    n=length(k);                                         % }
    figure(2)
    scatter(its,n,'b','filled')                          % { Plot the perfomance having
    xlabel('n iteration')                                %   in the x axis the iteration
    ylabel('Correspondences')                            %   i and y axis the right
    title('Performance')                                 %   classified points
    %axis([0 10 0 10])                                   % }
    hold on
    its=its+1;                                           % increase iteration
    %if yhat==data(:,3)                                  % { if algorithm classified
    %    break                                           %   corretly all the points
    %end                                                 %   stop the loop }
    if yhat==data(:,3) & y==0                            % { if algorithm classified
        iter=its;                                        %   corretly all the points
        y=1;                                             %
    end                                                  %   stop the loop
end                                                      %  }
hold on
figure(1)
x=[0:0.1:1.2];
plot(x,(-bavg-wavg(1)*x)/wavg(2),'k')                    % plot the last division line
xlabel('x_{1}')
ylabel('x_{2}')
legend('+1','-1','Location','southwest')
title('Perceptrons line of division (averaged version)')
hold on
figure(2)                                                % plot the last perform. point
scatter(its,n,'b','filled')
xlabel('n iteration')
ylabel('Right classified points')
title('Algorithm performance')
wavg(1)=wavg(1)+v*w(1);                                  % update last time the average
wavg(2)=wavg(2)+v*w(2);                                  %    weights vectors
bavg=bavg+v*b;                                           %
if iter==0
    iter=its;
end
end
```

Output:

```
wavg =

   1.0e+03 *

   -1.0677   -0.7167



bavg =
```

```
 -305.0000


iter =

     9
```

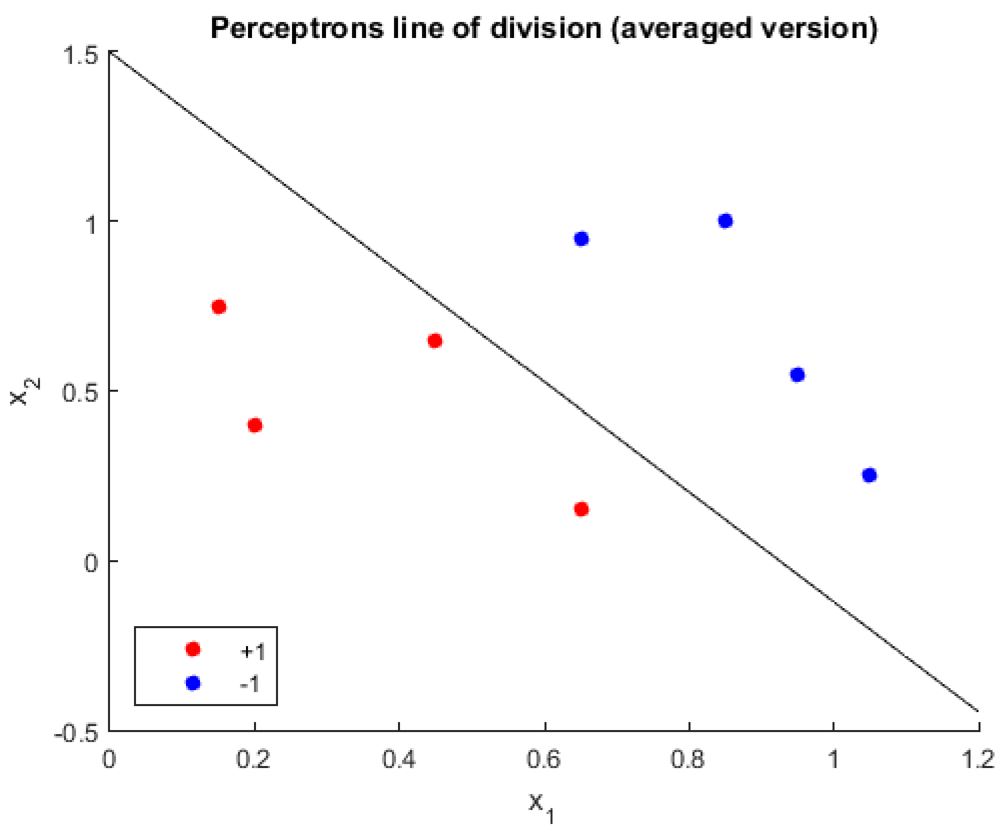

Figure 5: Average perceptron convergence of q25.m

Question 2 continued on next page. . .

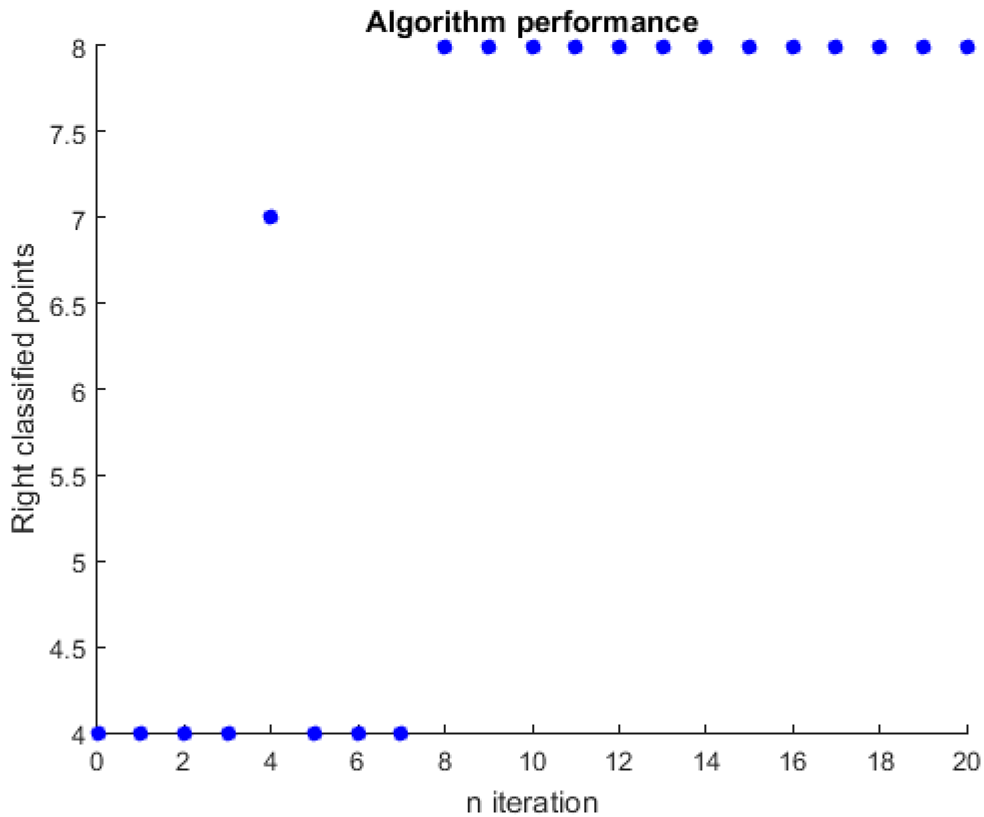

Figure 6: Algorithm performance of q25.m

2.6 Change the labels of some points. Average perceptron will never converge

If we change more label of the original dataset (here we have a new txt file called samplemod.txt), in a way that points can not be separated linearly, we'll never reach convergence.

Command window

```
>> file='samplemod.txt';            % txt file with the modified points
w=[1 -1];
b=0.2;
maxiter=50;
wavg=[0 0];
bavg=0;
v=0;
its=0;
[wavg,bavg,its]=q26(file,b,w,maxiter,wavg,bavg,v,its)
```

File q26.m

```
function [wavg,bavg,iter]=q26(file,b,w,maxiter,wavg,bavg,v,its)
[wavg,bavg,iter]=q25(file,b,w,maxiter,wavg,bavg,v,its)
end
```

Output:

```
wavg =
```

```

   1.0e+03 *

   -0.9269   -1.0263


bavg =

        1521


iter =

     50


wavg =

   1.0e+03 *

   -0.9269   -1.0263


bavg =

        1521


its =

     50
```

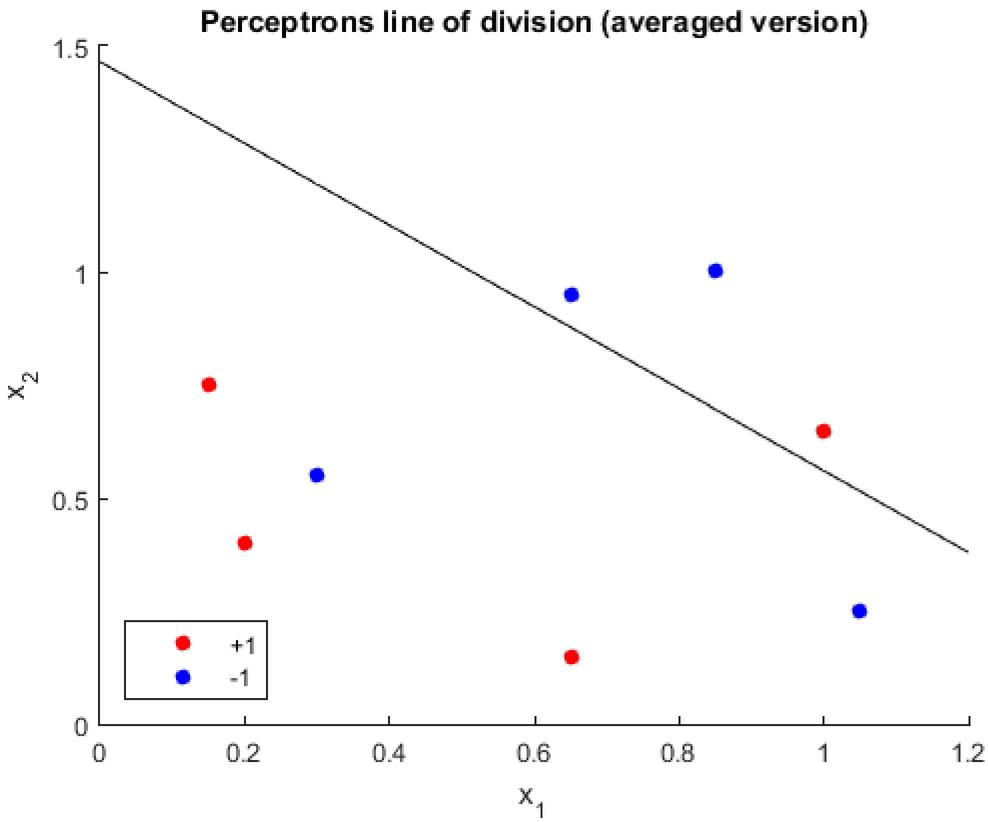

Figure 7: Average perceptron of q26.m

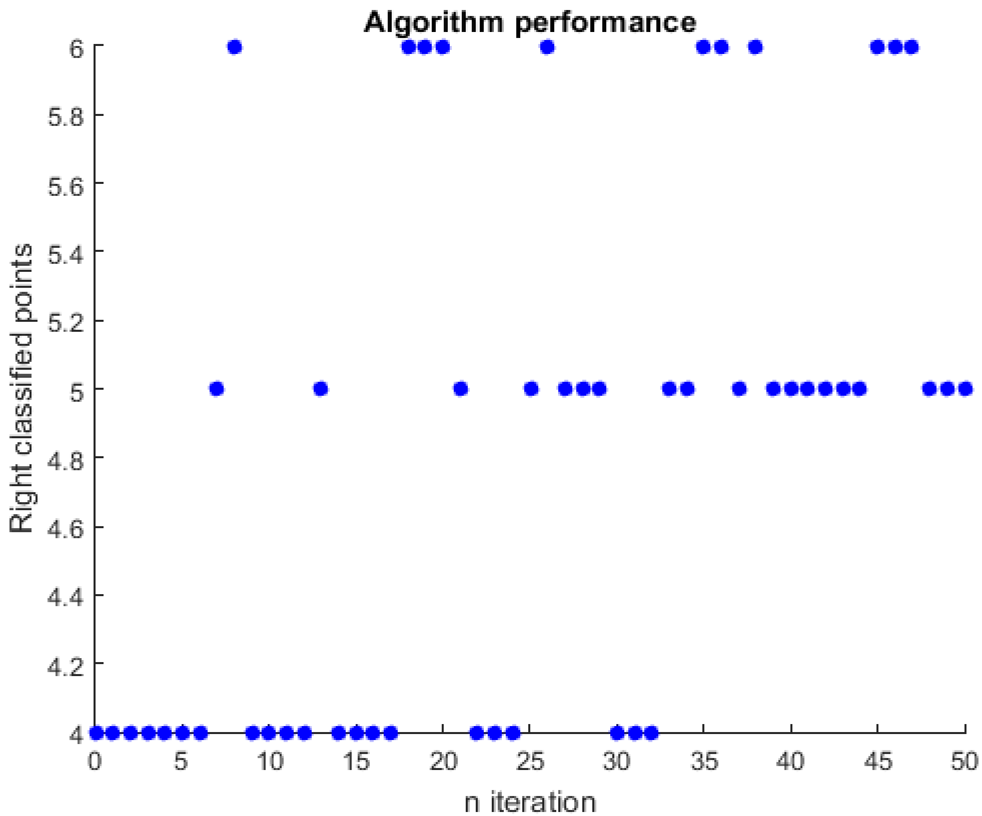

Figure 8: Algorithm performance of q26.m

Question 3

3.7 Average quadratic error and minimization of E(W,b)

The function receives in input a weight vector called w_all (with w_1,w_2 and b) and calculates the value of the error where in the sum we make the difference between the real values and the results in the hyperbolic tangent.

Command window

```
>> w_all=[0.2 1 -1];
[error] = q37a(w_all)
```

File q37a.m

```
function [error] = q37a(w_all)                         % Input: weights vector of
    file='sample.txt';                                 %        w1,w2 AND b
    data=importdata(file);                             %
    error=0;                                           % Set error value
    for i=1:length(data)                                               % { for i=1:length(data)
        f(i)=w_all(1)+w_all(2)*data(i,1)+w_all(3)*data(i,2);           %    update tbe Hyperbolic
        yhat(i)=(exp(f(i))-exp(-f(i)))/(exp(f(i))+exp(-f(i)));         %    Function, return and
        error=error+(data(i,3)-yhat(i))^2;                             %    sum the error between
```

```
    end                                                     %    real and estim. values }
    error=1/2*error;                                        %  Final value of error
end
```

Output:

```
error =

    5.7198
```

Now we initialize a starting guess x_0 and we'll minimize the error using the fminunc function of Matlab with a quasi-newton algorithm implemented. The results are the optimal values for s, w_1 and w_2

Command window:

```
x0 = [0,0,0]; % Starting guess
[b,w]=q37b(x0)
```

File q37b.m

```
function [b,w]=q37b(x0)
options = optimoptions(@fminunc,'Algorithm','quasi-newton'); % { Minimize with fminunc function
[opt_ws,fval,exitflag,output] = fminunc(@q37a,x0,options);   %   using quasi-newtown algorithm }
b=opt_ws(1);                     % optimal value for b
w=opt_ws(2:3);                   % optimal value for the weights vector
end
```

Output:

```
b =

   30.4343


w =

  -38.2258   -9.8615
```