CS446: Machine Learning **Fall 2016**

Problem Set 1

Yihao Zhang *Handed In: September 13, 2016*

1. Answer to problem 1
   a. **Pseudo Code**:

Algorithm 1 Hypothesis H

procedure (main())
    Define $H_n(x_1, x_2, \ldots, x_n)$ as a variable that has n attributes
    X is labeled either positive(1) or negative(-1)
    x_n represents 1, and $\neg x_n$ represents -1 as attributes
    First create a concatenation of all possible attributes: $x_1 \vee \neg x_1 \vee x_2 \vee \neg x_2 \vee \ldots \vee x_n \vee \neg x_n$
    Create a dictionary D for all the possible attributes
    for For each example H_n in negative examples: **do**
        for For each x_i in H_n(negative) **do**
            pop the symbol from D
    Reconstruct answer A from dictionary D by doing disjunction of all remaining symbols
    for every positive example H_n(positive): **do**
        if $A(H_n)$ is negative: **then**
            return "No consistent hypothesis"
            HALT
    Return A as answer

b.
In order to fit every negative examples, we cannot have any symbol in negative example because otherwise the whole expression will end up True according to the attribute of OR operation. For all the positive examples, we don't have to modify the answer to fit them. However, we do have to check if all the positive examples are consistent with the negative examples. That means, there's no contradiction in the example pool. Because there could be mistakes like a same expression is marked as both positive and negative.
c.
Runtime-wise, the procedure for creating initial answer and deleting symbols will be O(n). However we do have to check each symbols of m examples. So it will finally give us O(mn) time complexity.
d.
Using my algorithm, the outcome for new examples will probably fail due to over-fitting. The reason for over-fitting is: given limited examples, we can only delete limited symbols. That will leave the final expression longer than the "real" expression that we do not know. For example, we might get $x_1 \vee \neg x_1 \vee x_2$, but the answer is x_1

$\vee$ x_2. So provided that we do not have the full example pool. The answer we have will always over-fit.

2. Answer to problem 2
   a.
   So we are trying to find the min of $\| \vec{x}_0 - \vec{x} \|$, where $\vec{x}$ is a arbitrary vector in space $\vec{w}^T\vec{x} + \theta = 0$.
   We have this formula Distance:

$$\begin{aligned} D &= \frac{\|\vec{w}^T(\vec{x} - \vec{x}_0)\|}{\|\vec{w}\|} \\ &= \frac{\|\vec{w}^T\vec{x} - \vec{w}^T\vec{x}_0\|}{\|\vec{w}\|} \\ &= \frac{\| - \theta - \vec{w}^T\vec{x}_0\|}{\|\vec{w}\|} \\ &= \frac{\|\theta + \vec{w}^T\vec{x}_0\|}{\|\vec{w}\|} \end{aligned}$$

   b.
   The two hyper-planes have the same $\vec{w}$. This means they are parallel to each other. So the distance would just be $\frac{\|\vec{w}^T(\vec{x}_1 - \vec{x}_2)\|}{\|\vec{w}\|} = \frac{\|\theta_1 - \theta_2\|}{\vec{w}}$

3. Answer to problem 3
   a1.
   First, from δ is 0 to linearly separable:
   If δ is 0, we have $y(\vec{w}^T\vec{x}) > 1$. This is obviously ≥ 0.
   Second, from linearly separable to δ is 0:
   If a data set is linearly separable, $\min(\vec{w}^T\vec{x}+\theta)$(when y=1) $\geq 0 \geq \max(\vec{w}^T\vec{x}+\theta)$(when y=-1) Set the $\min(\vec{w}^T\vec{x} + \theta)$(when y=1) to be dpos, set $\max(\vec{w}^T\vec{x} + \theta)$(when y=-1) to be dneg. A plane have equal distance to both $\min(\vec{w}^T\vec{x} + \theta)$(when y=1) and $\max(\vec{w}^T\vec{x} + \theta)$(when y=-1) will have D = (dpos+dneg)/2 distance to both points. So we have the equation for a new hyperplane $(\vec{w}^T\vec{x} + \theta - D = 0)$. But since now $\min(\vec{w}^T\vec{x}+\theta)$ and $\max(\vec{w}^T\vec{x}+\theta)$ all have distance D to the new hyper plane. We have a new equation that $y(\vec{w}^T\vec{x} + \theta - \frac{dpos + dneg}{2}) > \frac{dpos - dneg}{2}$. Now we just have to divide both sides by $\frac{dpos - dneg}{2}$ and that will give us the right side as 1, thus delta as 0.
   a2.
   The trivial optimal solution is $\vec{w}$, θ and δ are all 0. Because firstly, delta is 0, the minimized value. Also, this answer satisfies all the constraints. Because we want a hyper-plane instead of trivial answer for our linear program. We choose formulate the

linear program as from 2 to 4.
a3.
Because x and y will be both 1 or -1, we have either $(w_1 + w_2 + ... + w_n + \theta) \geq 1$ or $-(-w_1 - w_2 - ... - w_n + \theta) \geq 1$. So we have $w_1 + w_2 + ... + w_n \geq 1 + \|\theta\|$ and δ is 0.

b1.
The code snippet:

```
% This function finds a linear discriminant using LP
% The linear discriminant is represented by
% the weight vector w and the threshold theta.
% YOU NEED TO FINISH IMPLEMENTATION OF THIS FUNCTION.

function [w,theta,delta] = findLinearDiscriminant(data)
%% setup linear program
[m, np1] = size(data);
n = np1-1;

% write your code here
c=[zeros(n+1,1);1];
b=[ones(m,1);0];
y =data(:,np1);
x = data(:,1:n);
yxt = zeros(m,n);
for i = 1:m
    yxt(i,1:n) = y(i)*x(i,1:n)
end
A=[yxt data(:,np1) ones(m,1); zeros(1,np1) 1];
%% solve the linear program
%adjust for matlab input: A*x <= b
[t, z] = linprog(c, -A, -b);

%% obtain w,theta,delta from t vector
w = t(1:n);
theta = t(n+1);
delta = t(n+2);

end
```

b2.
Code snippet:

```
% This function plots the linear discriminant.
% YOU NEED TO IMPLEMENT THIS FUNCTION
```

```
function plot2dSeparator(w, theta)

x = linspace(0,1);
y = -theta/w(2)-w(1)/w(2)*x;
plot(x,y);

end
```

Result and plot is attached in the appendix.
For Conjunction:
theta = -90.2115
delta =-2.4158e-13
And w4 = 190 and w8 = -193
delta is nearly 0. It has a negative sign because of the machine precision. So the data is linear separable.
We can make prediction of $x_4 \wedge \neg x_8$

b3.

Code snippet:

```
% This function computes the label for the given
% feature vector x using the linear separator represented by
% the weight vector w and the threshold theta.
% YOU NEED TO WRITE THIS FUNCTION.

function y = computeLabel(x, w, theta)
    if (transpose(w)*x+theta) >= 0
        y = 1;
    else
        y = -1;
    end
end
```

Result and plot is in the appendix.
For this problem, the accuracy is 1, meaning we have 100 percent success rate. And delta is 1.2619e-09, meaning that data is linearly separable. The figure means that there a high relationship between the badge and letter A, E and those having a bright color.
I changed the E to A in the alphabet. Resulting a 0.9787 success rate. With new delta = -4.6896e-13, we can say that the training data is still linearly separable. But we didn't achieve 100 percent on test. The new figure seems to have more important positions and letters. But the correlation seems all pretty low (below half of the original max).

b4.
Code snippet:

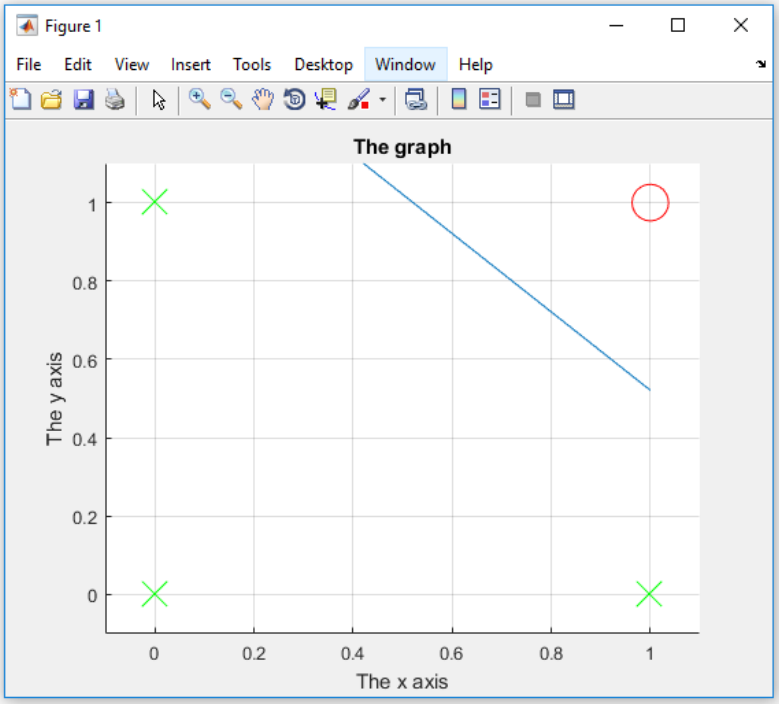
```
% This function solves the LP problem for a given weight vector
% to find the threshold theta.
% YOU NEED TO FINISH IMPLEMENTATION OF THIS FUNCTION.

function [theta,delta] = findLinearThreshold(data,w)
%% setup linear program
[m, np1] = size(data);
n = np1-1;

% write your code here
c=[zeros(n+1,1);1];
b=[ones(m,1);0];
y =data(:,np1);
x = data(:,1:n);
yxt = zeros(m,n);
for i = 1:m
    yxt(i,1:n) = y(i)*x(i,1:n)
end
A=[yxt data(:,np1) ones(m,1); zeros(1,np1) 1];
%% solve the linear program
%adjust for matlab input: A*x <= b
[t, z] = linprog(c, -A, -b, [], [], [w' -inf -inf], [w' inf inf]);


%% obtain w,theta,delta from t vector
w = t(1:n);
theta = t(n+1);
delta = t(n+2);

end
```

There's three lines separating the data pretty well with θ to be -243.262997,-218.994588 and -123.611614; and delta to be 0, which tells us we achieved perfect linear separation. In this case they are all the same because they all separates the data. We can infer that the solution is not unique.

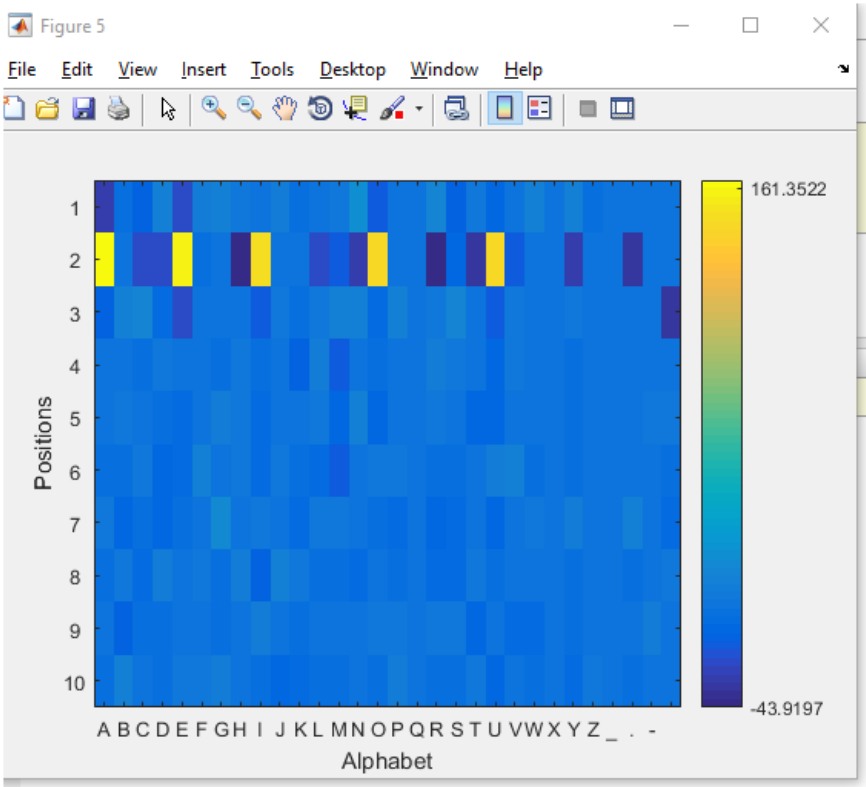
Results:

learnConjunction:

```
>> learnConjunctions
Optimization terminated.

w =

  156.2452
  156.2452


theta =

 -237.6709


delta =

   8.3610e-10

Optimization terminated.

w =

    2.9104
   -2.0498
    0.1775
  190.5196
    0.1399
   -3.1010
   -2.9534
 -193.2778
    1.1679
   -8.8942


theta =

  -90.2115


delta =

  -2.4158e-13

``
```

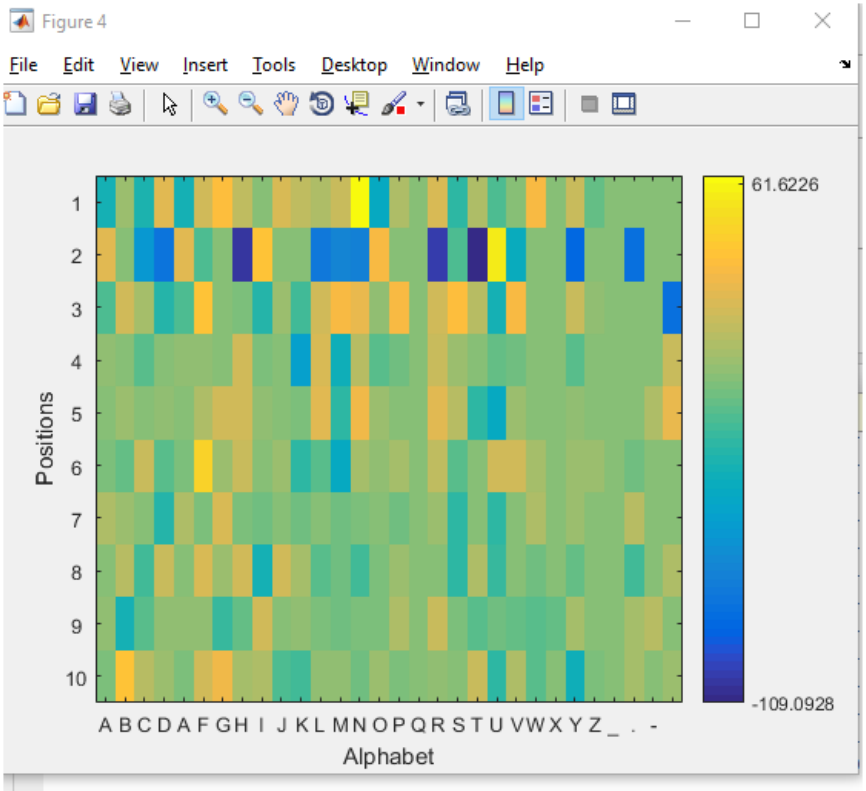


LearnBadges1:



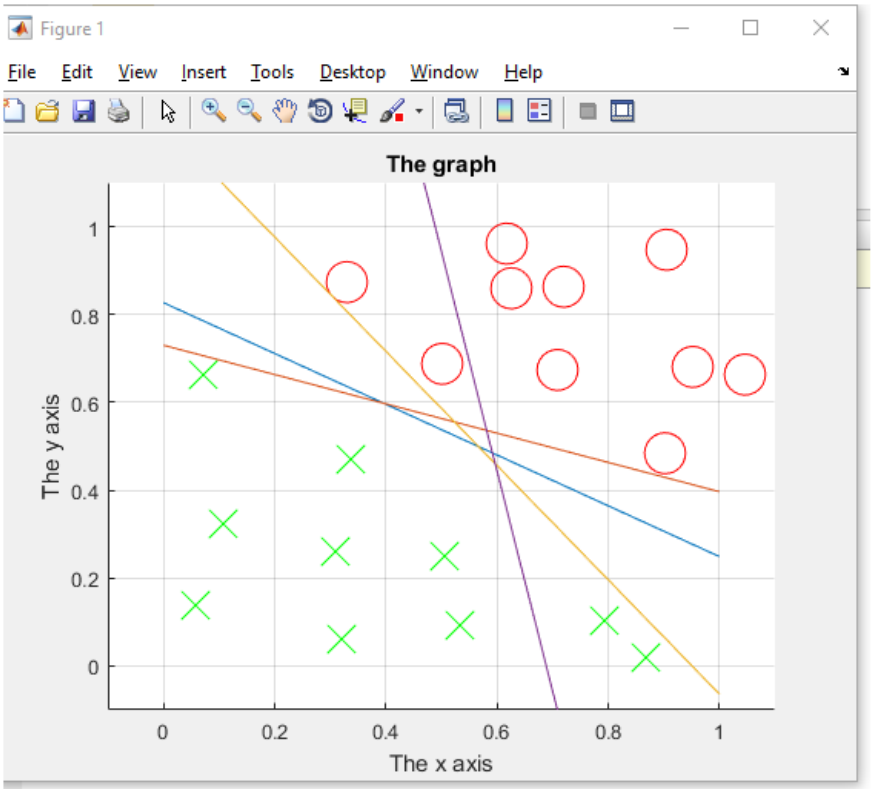

```
Optimization terminated.

delta =

   1.2619e-09


accuracyInTrain =

     1


accuracyInTest =

     1
```

LearnBadges2:




```
Optimization terminated.

delta =

  -4.6896e-13


accuracyInTrain =

     1


accuracyInTest =

    0.9787
```

LearnMultipleHyperplane:




```
Optimization terminated.

theta =

 -344.3350


delta =

   92.9650
```