CS446: Machine Learning **Spring 2017**

Problem Set 1

Bangqi Wang *Handed In: February 4, 2017*

1. Answer to problem 1

   a. The main idea of the algorithm is to construct an initial hypothesis and then update the hypothesis when iterating the training set. For instance, the algorithm constructs an initial complete hypothesis that is the conjunction of all variables and their negations. i.e. $\mathbf{H} = (x_1 \wedge x_2 \wedge ... \wedge x_n \wedge \neg x_1 \wedge \neg x_2 \wedge ... \wedge \neg x_n)$. Then, for each positive sequences in training set, the algorithm updates the hypothesis by removing the contradictions. i.e. $x_i = a$ in hypothesis, but $x_i = b$ in sequence. After finishing iterating positive sequences, using negative sequences to check the hypothesis. If there is any contradiction between negative sequence and hypothesis, the algorithm will indicate that there is no consistent hypothesis.

Algorithm 1 Pseudocode:

```
1: procedure MyProcedure
2:     result ← (x_1 = 1) ∧ (x_2 = 1) ∧ ... ∧ (x_n = 1) ∧ (x_1 = 0) ∧ (x_2 = 0) ∧ ... ∧ (x_n = 0)
3:     for each '+' sequence S in training set do
4:         if (x_i = v) ≠ S_i then
5:             remove (x_i = v) from result
6:         end if
7:     end for
8:     for each '−' sequence S in training set do
9:         if S contrdict with result then
10:            Indicate: no consistent hypothesis
11:            halt
12:        end if
13:    end for
14:    return result
15: end procedure
```

   b. Inorder to prove the correctness of this algorithm, I will prove that there are neither **false positive** nor **false negative**.

      b.1 If there is a **false positive**, there are at least one condition absent from *result*, say $(x_i = v)$. However, in the first loop, the algorithm eliminated $(x_i = v)$ from complete conjunction, because it contradicts with other positive examples. Those two assumptions are mutually conflicting.

      b.2 If there is a **false negative**, there are at least one incorrect condition in *result*, say $(x_i = v)$. However, in the first loop, the algorithm eliminated all conditions that contradict with other positive examples. The remaining

($x_i = v$) should not contradict with any positive sequence. Therefore, those two assumptions are mutually conflicting.

Therefore, the *result* contains neither **false positive** nor **false negative**. The algorithm is correct.

c. The algorithm traverses m training examples and checks each example with hypothesis *result* that has $2n$ conditions at worst. The runtime should be $\mathbf{O}(mn)$.

d. There might be **false negative** in this situation. The example labeled as positive must be positive. However, the example labeled as negative might be positive, because the algorithm updates the hypothesis according to positive examples in training examples. The hypothesis changed from general to specific. The hypothesis *result* is the superset of real hypothesis **H**.

2. Answer to problem 2

a. Assume a vector $\vec{w}$ that is perpendicular to the hyperplane. Point $x^* = (x_1^*, x_2^*, ..., x_n^*)$ stands for any point in hyperplane. Define $\vec{v}$ as vector from point $x^0 = (x_0^0, x_1^0, ..., x_n^0)$ to point x^*. *(I use x^* and x^0 to express the two points. * and ^{0} are just labels.)*

$$\vec{v} = \begin{bmatrix} x_0^* - x_0^0, & x_1^* - x_1^0, & ..., & x_n^* - x_n^0 \end{bmatrix}^T$$

Then, the length D of the projection of $\vec{v}$ onto $\vec{w}$ is the distance from the point to the plane.

$$\begin{aligned}
D &= |\operatorname{proj}_{\vec{w}} \vec{v}| && (1)\\
&= \frac{|\vec{w}^T \cdot \vec{v}|}{\|\vec{w}\|} && (2)\\
&= \frac{|w_0 \cdot (x_0^* - x_0^0) + w_1 \cdot (x_1^* - x_1^0) + ... + w_n \cdot (x_n^* - x_n^0)|}{\sqrt{w_0^2 + w_1^2 + ... + w_n^2}} && (3)\\
&= \frac{|w_0 \cdot x_0^* - w_0 \cdot x_0^0 + w_1 \cdot x_1^* - w_1 \cdot x_1^0 + ... + w_n \cdot x_n^* - w_n \cdot x_n^0|}{\|\vec{w}\|} && (4)\\
&= \frac{|\theta + w_0 \cdot x_0^* + w_1 \cdot x_1^* + ... + w_n \cdot x_n^*|}{\|\vec{w}\|} && (5)\\
&= \frac{|\vec{w}^T \cdot \vec{x^*} + \theta|}{\|\vec{w}\|} && (6)
\end{aligned}$$

b. The distance D between the hyperplanes:

$$\begin{aligned}
\vec{w}^T \cdot \vec{x} + \theta_2 &= 0 && (7)\\
\vec{w}^T \cdot \vec{x} &= -\theta_2 && (8)\\
D &= \frac{|\vec{w}^T \cdot \vec{x} + \theta_1|}{\|\vec{w}\|} && (9)\\
&= \frac{|\theta_1 - \theta_2|}{\|\vec{w}\|} && (10)
\end{aligned}$$

3. Answer to problem 3

   a. a.1 For this problem, I will prove that the data set D is linearly separable if *(a.1.1) there exists a hyperplane that satisfies condition* (3) *with* $\delta = 0$, and *(a.1.2) only if there exists the hyperplane.* Then I will consider the case (*a*.1.3) *that* $\delta > 0$.

   (a.1.1) if hyperplane with $\delta = 0$, then dataset D is linear separable:

   - For case $y_i = -1$, $\delta = 0$:

$$\begin{aligned} y_i(\vec{w}\vec{x_i} + \theta) &\geq 1 - \delta && (11)\\ -1 \cdot (\vec{w}\vec{x_i} + \theta) &\geq 1 - 0 && (12)\\ (\vec{w}\vec{x_i} + \theta) &\leq -1 && (13)\\ (\vec{w}\vec{x_i} + \theta) &\leq 0 && (14)\\ & && (15) \end{aligned}$$

   - For case $y_i = 1$, $\delta = 0$:

$$\begin{aligned} y_i(\vec{w}\vec{x_i} + \theta) &\geq 1 - \delta && (16)\\ 1 \cdot (\vec{w}\vec{x_i} + \theta) &\geq 1 - 0 && (17)\\ (\vec{w}\vec{x_i} + \theta) &\geq 1 && (18)\\ (\vec{w}\vec{x_i} + \theta) &\geq 0 && (19)\\ & && (20) \end{aligned}$$

   The result is consistent with condition (1). Therefore, there exists linear separable dataset D if there exists a hyperplane that satisfies condition (3) with $\delta = 0$.

   (a.1.2) if dataset D is linear separable, then there exists hyperplane with $\delta = 0$: We need to prove that:

$$\begin{aligned} (\vec{w}\vec{x_i} + \theta) \geq 0 &\quad \Rightarrow y_i = 1 && (21)\\ (\vec{w}\vec{x_i} + \theta) \leq 0 &\quad \Rightarrow y_i = -1 && (22) \end{aligned}$$

   We assume that there is a hyperplane that separates the dataset and touches a positive point. We define D as the distance between this hyperplane and the closest negative point and $\vec{v}$ as vector.

$$\begin{aligned} D &= \frac{|\vec{w}^T \cdot \vec{x_i} + \theta|}{\|\vec{w}\|} && (23)\\ (\vec{w}\vec{x_i} + \theta) &= 0 && (24) \end{aligned}$$

   We shift the hyperplane by $D/2$ along $\vec{v}$. The hyperlane is still linear separatble. After shifting $D/2$, $(\vec{w}\vec{x_i} + \theta) > 0$. Scaling the hyperplane by multiplying a factor in both sides. We get equation: $(\vec{w}\vec{x_i} + \theta) \geq 1$ and $(\vec{w}\vec{x_i} + \theta) \leq 1$. This is consistent with condition (3) with $\delta = 0$.

(a.1.3) For case $\delta > 0$:

- if $\delta < 1$: we can use similar trick to prove that the dataset is linear separable.
- if $\delta \geq 1$: the dataset is not separable.

a.2 The trivial solution is:

$$\vec{w} = [0_0, 0_1, ..., 0_n],\ \theta = 0,\ \delta = 0$$

This is the optimal solution. $\theta = 0$ is the smallest θ that minimizes the δ with condition $1 - \delta \geq 0$. To avoid this trivial solution, we add a positive constant to make sure there is a hyperplane.

a.3 By plugging in $\vec{x_1}$ and $\vec{x_2}$:

$$\sum_{i=1}^{n} \vec{w_i} + \theta \quad \geq 1 \tag{25}$$

$$\sum_{i=1}^{n} -\vec{w_i} + \theta \quad \leq 1 \tag{26}$$

b. b.1 **findLinearDiscriminant.m**

findLinearDiscriminant.m

```
% 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
y = data(:, np1);
features = data(:, 1:n);
sign = zeros(m, n);
for i = 1:m
    sign(i,:) = y(i) * features(i,:);
end
c = [zeros(np1,1); 1];
A = [sign, y, ones(m,1); zeros(1,np1), 1];
b = [ones(m,1); 0];
%% 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
```

b.2 **hw1sample2d.txt**

hw1sample2d.txt

```
0 0 -1
0 1 -1
1 0 -1
1 1 1

```

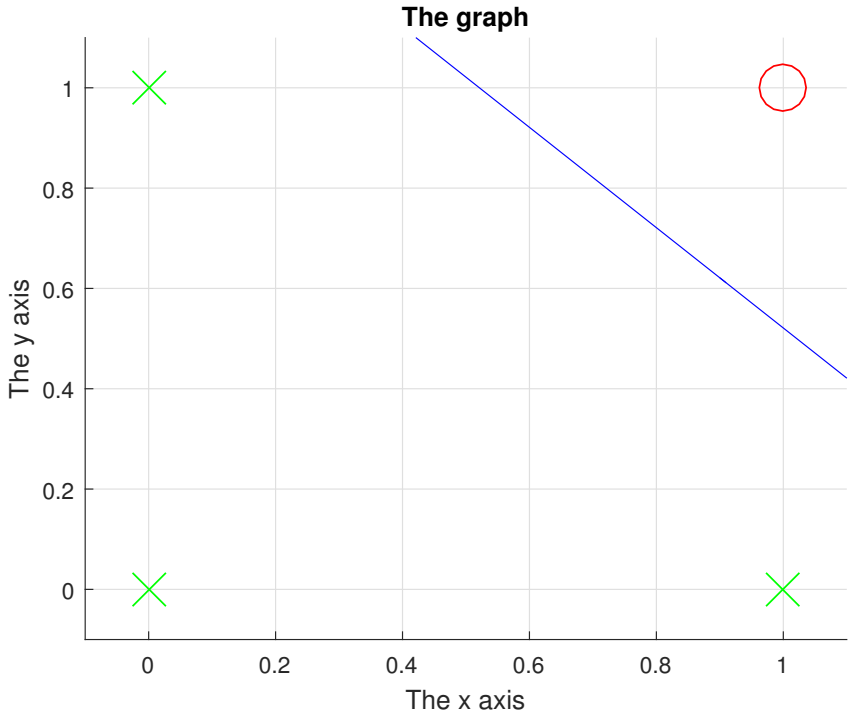
The dataset represents a monotone conjunction over two variables.

plot2dSeparator.m

plot2dSeparator.m

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

function plot2dSeparator(w, theta)
xdata = -10:0.01:10;
y = -w(1)/w(2) * xdata - theta/w(2);
plot(xdata, y, 'b');
end

```

two dimension separator:



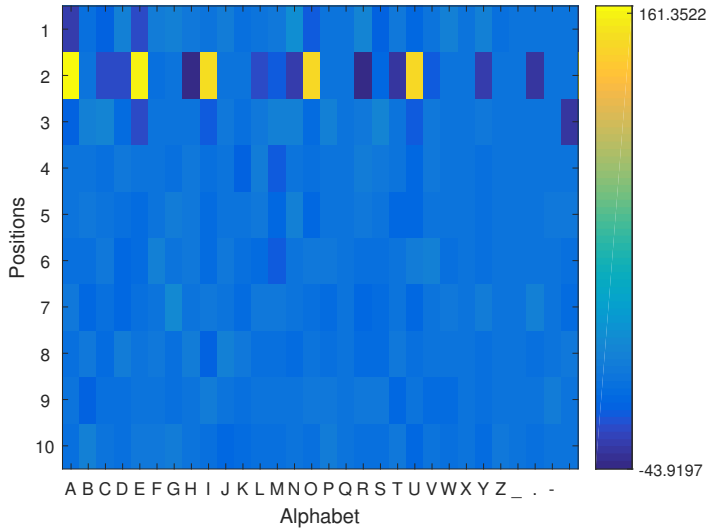
$\theta = -90.2115$, $\delta = -1.5632e - 13$
$\vec{w} = [2.910, -2.050, 0.178, 190.520, 0.140, -3.101, -2.953, -193.278, 1.168, -8.894]$

In this solution, $\delta \approx 0$ shows that the dataset is linearly separable. Since x_4 and x_8 are significantly larger than others, the label is mostly depent on x_4 and $\neg x_8$. The hyperplane should be $x_4 \wedge \neg x_8$. In this solution, $\theta << 0$ guarantees a negative ouput when $\vec{w}\vec{x}$ is small.

b.3 computeLabel.m

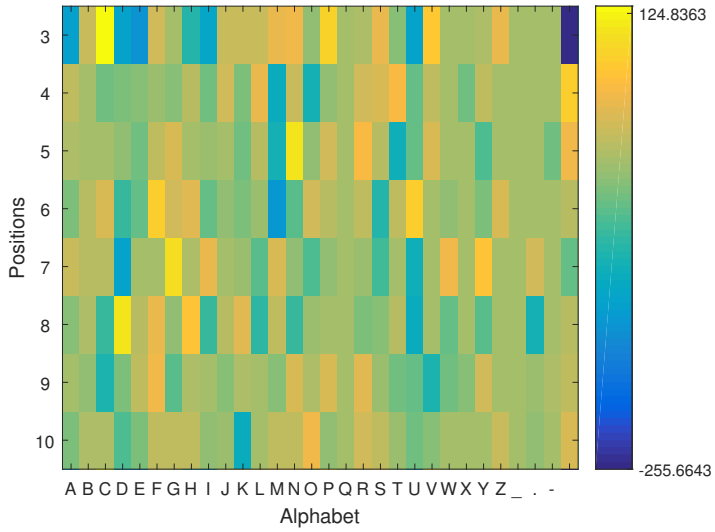
computeLabel.m

```
% 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 w'*x + theta >= 0
    y = 1;
else
    y = -1;
end
end

```



($\delta = 1.2619e-09$, $accuracyInTrain = 1$, $accuracyInTest = 1$)
Since $\delta \approx 0$, there exists a hyperplane that separates dataset. This set of features shows a perfect accuracy in both training dataset and test dataset.

change the position from 1:10 to 3:10:
This change will ignore the first two positions, so the obvious yellow cube in image above will be ignored.

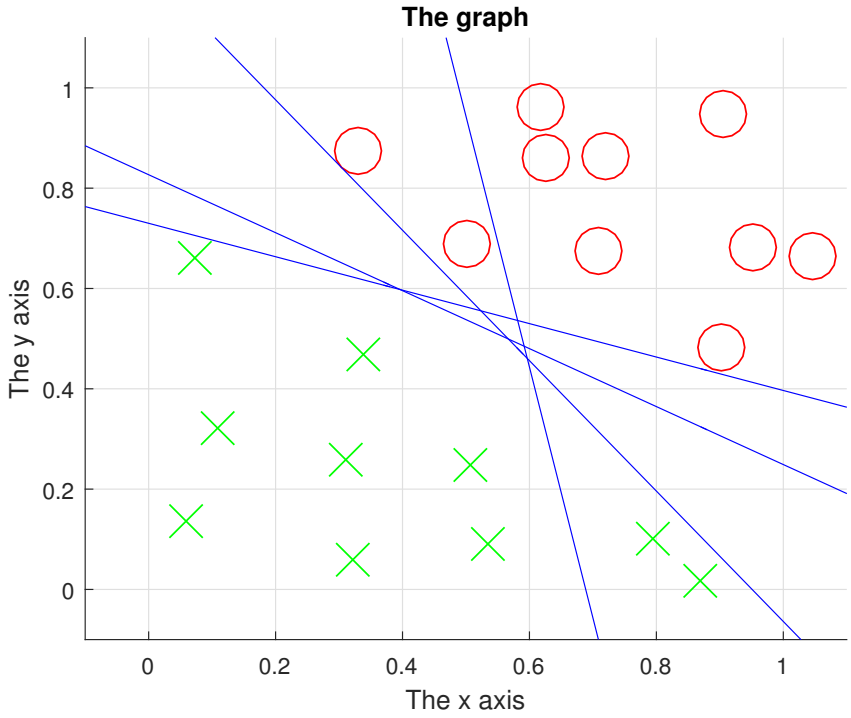


($\delta = 1.2179e-11$, $accuracyInTrain = 1$, $accuracyInTest = 0.7234$)
Since $\delta \approx 0$, the new hyperplane separates the dataset. The accuracy for training dataset is 1, but the accuracy for test is only 0.7. There is no obvious feature under this alphabet and position.

b.4 **findLinearThreshold.m**

findLinearThreshold.m

```
% 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
y = data(:, np1);
features = data(:, 1:n);
sign = zeros(m, n);
for i = 1:m
    sign(i,:) = y(i) * features(i,:);
end
c = [zeros(np1,1); 1];
A = [sign, y, ones(m,1); zeros(1,np1), 1];
b = [ones(m,1); 0];
%% 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
```



($\delta_1 = 1.0658e-12$, $\delta_2 = -2.2737e-13$, $\delta_3 = -2.0009e-11$, $\delta_4 = 92.9650$)
Among those four hypothesis, only three hypothesis sucessfully separates the dataset, ($\delta_4 >> 0$). The lines which left end located between 0.8 and 1 separates the dataset exactly. Since $\delta_4 >> 0$, the most vertical line fails to separates the dataset. Other two lines are closer to positive or negative examples. We can infer that there are multiply lines that can separate the dataset.