**CS446: Machine Learning** **Spring 2017**

# Problem Set 1

Haoran Wang *Handed In: February 3, 2017*

1. Answer to problem 1

   (a) The algorithm is designed to learn conjunctions in the hypothesis space, so we should begin with the complete conjunctions of all elements:

   $$G = x_1 \wedge (\neg x_1) \wedge x_2 \wedge (\neg x_2) \wedge ... \wedge x_n \wedge (\neg x_n) \tag{1}$$

   So the result of initial conjunction $G$ is always negative. Then for every positive training example, we need to eliminate $x_i$ or $(\neg x_i)$ based on the value of $x_i$ in the training example. If $x_i = 1$, then we eliminate $(\neg x_i)$ in $G$; If $x_i = 0$, then we eliminate $(x_i)$ in $G$. After going through all positive training examples, we could get a final conjunction function $G$, if $G$ is empty, then we halt. If it is not empty, we could verify the modified conjunction function $G$ by passing all training examples, comparing the labels it will produce with the actual label of the example. If there is any inconsistent example, we halt. If we halt at any time in the mentioned process, we can say that there is no consistent hypotheses.

**Algorithm 1** LearnConjunctions(T[1,2,3,...,m])

```
G ← x_1 ∧ (¬x_1) ∧ x_2 ∧ (¬x_2) ∧ ... ∧ x_n ∧ (¬x_n)
for i ← 1 to m do
    if T[i] is labeled as '+' then
        if T[i]_j = 1 then
            Remove (¬x_j) from G
        end if
        if T[i]_j = 0 then
            Remove x_j from G
        end if
    end if
end for
if G is empty then
    halt, and return "No Consistent Hypothesis"
end if
for i ← 1 to m do
    if Label of T[i] is not the same with output of G then
        halt, and return "No Consistent Hypothesis"
    end if
end for
return G
```

(b) Proof:
There are two possible results from the algorithm: first, we get a conjunction function which is consistent with the hypothesis. In this circumstance, there is one step in the algorithm that it will verify the found conjunction function $G$ by going through all training examples. So if we could get a conjunction function from the algorithm, we could say that it is consistent with all the examples we have seen. So this part is verified.

Second, we may get "No consistent hypothesis" from the algorithm. 1) If we halt at the first loop after going through all positive examples, we could know that $G$ is empty now, which means we do not find a consistent conjunction function with all positive examples, so it is impossible to find another function consistent with all positive training examples because the initial form of $G$ is already the most general case. 2) If we halt at the second loop when going through all training examples(Both positive and negative), the function is inconsistent with at least one negative training example. Because we already eliminate the $x_i$ or $(\neg x_i)$ or both due to it is inconsistent with the positive examples, so we cannot add back to the conjunction function $G$. So it is a contradiction to the previous loop, which means there is no consistent hypothesis existed.
Finish

(c) As we can see from the algorithm pseudo-code, the first loop will compare the value of elements contained in the positive training example. So in worst case, we need to compare $n$ elements if all of them are in training example. The time is $O(n)$ if we ignore the time of removing elements from $G$. As we have two loops here, so we need go through all $m$ training examples, and we could infer that the time complexity is $O(mn)$.

(d) If the conjunction function $G$ is consistent with all training examples we have seen, we could say that we are confident it could correctly predict positive examples. In other words, if the output of $G$ is positive, the label of new example must be positive. If the output is negative, we cannot say that our output is one hundred percent accurate. It could be wrong. Because we have eliminated the inconsistent elements in the process to find consistent function $G$ as initialized in equation (1). The initial conjunction will output negative by default. When going through positive examples, we remove some elements in $G$ to make it consistent with positive cases, but it is always the most general case of all positive cases. So based on this process, we are certain that if we predict a positive label, it must be positive. But when we predict a negative label, maybe it is wrong because we are not guaranteed to have seen all positive examples, so maybe we will misclassify some extreme positive cases.

In a nutshell, when the function output is positive, the label must be positive. If the output is negative, the label maybe negative or positive.

2. Answer to problem 2

   (a) Analytical solution: suppose $x_1$ is a point on the hyperplane $\vec{w}^T\vec{x} + \theta = 0$, so we have the following equation:

   $$\vec{w}^T\vec{x_1} + \theta = 0 \tag{2}$$

   The distance between $x_0$ and $x_1$ is:

   $$d = \|\vec{x_0} - \vec{x_1}\| \tag{3}$$

   We know that the normal vector of the hyperplane is $\vec{w}$, so in order to find the distance between the point and the hyperplane, we only need to calculate the projection of d on the direction of $\vec{w}$, which is calculated as:

   $$h = d \cdot cos\theta, \ \textit{where } \theta \textit{ is the angle between } \vec{w} \textit{ and } \vec{x_0} - \vec{x_1} \tag{4}$$

   Thus, the distance $h$ could be calculated as:

   $$\begin{aligned} h = d \cdot cos\theta &= \frac{\vec{w} \cdot (\vec{x_0} - \vec{x_1})}{\|\vec{w}\| \cdot \|\vec{x_0} - \vec{x_1}\|} \cdot \|\vec{x_0} - \vec{x_1}\| \\ &= \frac{\|\vec{w} \cdot \vec{x_0} - \vec{w} \cdot \vec{x_1}\|}{\|\vec{w}\|} \\ &= \frac{|\vec{w} \cdot \vec{x_0} + \theta|}{\|\vec{w}\|} \end{aligned} \tag{5}$$

   (b) Suppose $\vec{x_1}$ is a point on the hyperplane $\vec{w}^T\vec{x} + \theta_1 = 0$, similarly, $\vec{x_2}$ is a point on the other hyperplan $\vec{w}^T\vec{x} + \theta_2 = 0$. Suppose the distance between $\vec{x_1}$ and $\vec{x_2}$ could be represented as the distance between two hyperplanes.

   We know the normal vector $\vec{w}$ is perpendicular to the two hyperplanes, so the unit vector of $\vec{w}$ could be used as:

   $$\vec{u} = \frac{\vec{w}}{\|\vec{w}\|}$$

   So the vector between $\vec{x1}$ and $\vec{x_2}$ could be represented as:

   $$\vec{x_1} - \vec{x_2} = d \cdot \vec{u} \tag{6}$$

   where d is the distance (scalar) between two hyperplanes. So we can get the representation of d by dot product $\vec{w}$ on both sides in order to use the equations $\vec{w}^T\vec{x} + \theta_1 = 0$ and $\vec{w}^T\vec{x} + \theta_2 = 0$, the equation is:

   $$\begin{aligned} \vec{w} \cdot \vec{x_1} - \vec{w} \cdot \vec{x_2} &= d \cdot \frac{\vec{w}}{\|\vec{w}\|} \cdot \vec{w} \\ d &= \frac{|\vec{w} \cdot \vec{x_1} - \vec{w} \cdot \vec{x_2}|}{\|\vec{w}\|} \\ d &= \frac{|\theta_1 - \theta_2|}{\|\vec{w}\|} \end{aligned} \tag{7}$$

   So the distance of two hyperplanes is $d = \frac{|\theta_1 - \theta_2|}{\|\vec{w}\|}$

3. Answer to problem 3

(a) i. First we need to prove from LHS to RHS, suppose we have data set $D$ is linearly separable, we need to prove that $\delta = 0$ in this situation.

We know that there exists a hyperplane $\vec{w}^T\vec{x} + \theta = 0$ that could separate the data set, we choose the point that is closet to the hyperplane from the positive side. And it is the same that we need select a point from negative side. We suppose $V_1$ is the point from positive side, we have:

$$V_1 = \vec{w}^T\vec{x_1} + \theta \geq 0$$

Suppose $V_2$ is the point from negative side, we have:

$$V_2 = \vec{w}^T\vec{x_2} + \theta < 0$$

Suppose there exists a small $\alpha$ that we could get a new hyperplane that could also separate the data set if we move the current hyeperplane by $\alpha$. We have $V_1 - \alpha \geq 0 > V_2 - \alpha$, we could get the value of $\alpha$ analytically. Since we move the hyperplane on both sides, which means two hyperplanes are symmetric along with a central hyperplane that has the biggest margin between the two points. And the distance between the central hyperplane and either of the two hyperplanes is the same, we can get the equation of this distance by using $\alpha$:

$$|V_1 - \alpha| = |V_2 - \alpha|$$
$$V_1 - \alpha = -V_2 + \alpha$$
$$\alpha = \frac{V_1 + V_2}{2}$$

To get the format of $y_i(\vec{w}^T\vec{x_i} + \theta) \geq 1 - \delta, \qquad \forall(\vec{x_i}, y_i) \in D$, we have:

$$y_i(\vec{w}^T\vec{x_i} + \theta - \alpha) = \frac{V_1 - V_2}{2}$$

Since $V_1 > V_2$, we have :

$$y_i(\frac{2}{V1 - V2} \cdot \vec{w}^T + \frac{2}{V1 - V2} \cdot (\theta - \alpha)) = 1$$

$$y_i(\vec{w'}^T + \theta') \geq 1 - \delta$$

So we could see that $\vec{w'} = \frac{2}{V1-V2} \cdot \vec{w}$, $\theta' = \frac{2}{V1-V2} \cdot (\theta - \alpha)$, and $1 = 1 - \delta$, which implies $\delta = 0$

Second, we also need to prove from RHS to LHS, when we have $\delta = 0$, we need to prove that

$$y_i = \begin{cases} 1 & \text{if } \vec{w}^T\vec{x_i} + \theta \geq 0 \\ -1 & \text{if } \vec{w}^T\vec{x_i} + \theta < 0. \end{cases}$$

So, $\delta = 0$, we have

$$y_i(\vec{w}^T\vec{x} + \theta) \geq 1 - \delta$$
$$y_i(\vec{w}^T\vec{x} + \theta) = 1,\ where\ |y_i| = 1$$
$$\vec{w}^T\vec{x} + \theta = 1 > 0$$
$$\vec{w}^T\vec{x} + \theta = -1 < 0$$

Therefore, it is obvious that the equations satisfy the condition in (1). So we finish the proof.

If we have $\delta > 0$, we have several conditions to consider. First, if $1 - \delta > 0$, we can prove that data set could be separated in the same way as mentioned for $\delta = 0$. But if $\delta \geq 1$, $y_i$ and $\vec{w}^T\vec{x_i} + \theta$ have different signs, so the data set is not separable.

ii. The trivial solution:
it is easy to see that if we set $\delta = 0$, then $\vec{w}, \theta$ are also 0. This solution could satisfy the conditions we get from [2] to [4]. And $\delta = 0$ could satisfy the condition [4].

We don't use the formulation because the trivial optimal solution is not a hyperplane to separate the data set.

iii. There are only two data points in the data set. It is obvious that we could separate the data set.

$$w_1 + w_2 + w_3 + ... + w_n + \theta \geq 1 \tag{8}$$

$$-(w_1 + w_2 + w_3 + ... + w_n - \theta) \leq -1 \tag{9}$$

So from (8) and (9), we have $w_1 + w_2 + w_3 + ... + w_n \geq 1 + |\theta|$, and $\delta = 0$. This is the optimal solution.

(b) i. The function findLinearDiscriminant is written as:

```
% 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
% representation of A, b, c
```

```
A = zeros(m+1, n+2);
b = zeros(m+1,1);
c = zeros(n+2,1);
c(n+2,1) = 1;
b(1:m,1) = 1;
% label of the training examples
y_i = data(:,np1);
for i = 1:m
    A(i,1:n) = y_i(i) * data(i,1:n);
end
A(1:m,n+1) = y_i(1:m);
A(1:m,n+2) = 1;
A(m+1,n+1) = 0;
A(m+1,n+2) = 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
```

ii. 1)The monotone conjunction data set is:

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

2)The plot2dSeparator function is:

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

function plot2dSeparator(w, theta)
x = [-2:0.01:2];
y = -(w(1)*x+theta)/w(2);
plot(x, y);
end
```

3)The picture of the line separator with data is shown as Figure 1:

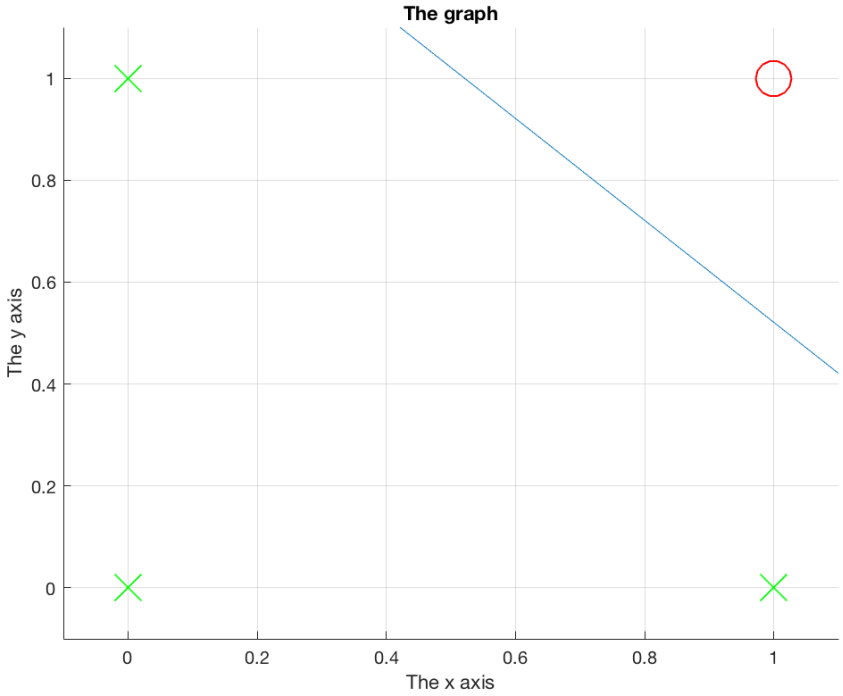


Figure 1: Separator with data points

After using the findLinearDiscriminant function, we get the hyperplane of 2d monotone conjunction function as:

$$156.2452x_1 + 156.2452x_2 - 237.6709 = 0$$

4)For learning hw1conjunction.txt, we get the output as shown at Figure 2:

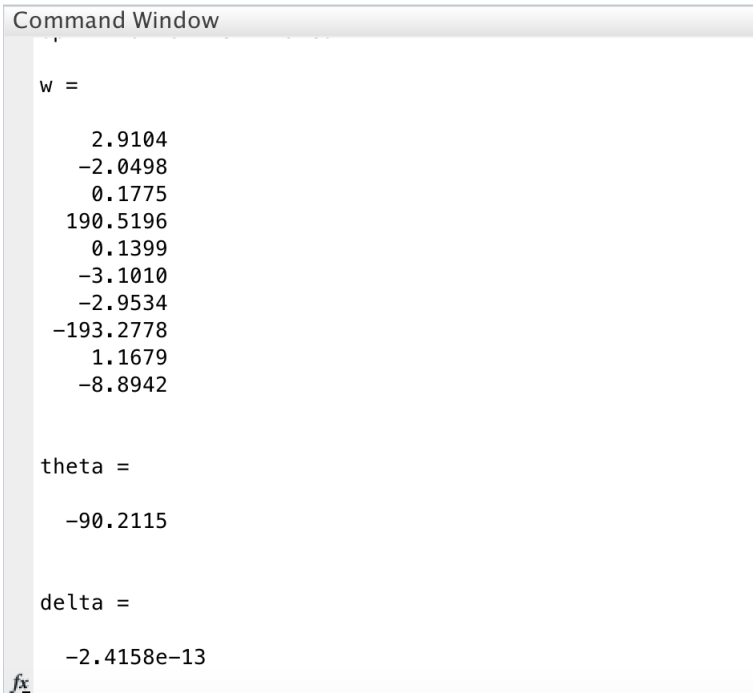


Figure 2: Output for hw1conjunction.txt

According to the result from Figure 6, the answer is: $\vec{w}^T x - \theta = 0$. $\theta = -90.2115$, $\delta = -2.4158 * 10^{-13}$ and $\delta \approx 0$, $\vec{w} is listed in the Figure 6$. Therefore, from this function, we could conclude that the data set is separable. $\delta$ is the target we want to minimize in this function. $\theta$ is the threshold for the function $\vec{w}^T \vec{x}$, which means it could be a range of values that could fit the hyperplane within the gap between the closest positive point and negative point. As for $\vec{w}$, we could observe from Figure 6 that the 4th and 8th element of $\vec{w}$ have larger weights compared with others, although the value of 8th weight is negative. To represent the weights in a conjunction function, we could denote it as:

$$G = x_4 \wedge \neg x_8 \tag{10}$$

We have $\neg x_8$ because the value is negative in the actual weights, so we use negation in this conjunction.

iii. 1)The computeLabel function is implemented as follows:

```
% 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)
temp = w.' * x + theta;
% w_dim = n x 1; where x_dim = n x 1;
disp(w)
if temp < 0
    y = -1;
```

```
12 else
13     y = 1;
14 end
15 end
```

2)The result of the script is shown at Figure 3:

```
delta =

   1.2619e-09


accuracyInTrain =

     1


accuracyInTest =

     1
```

Figure 3: Result of learnBadges

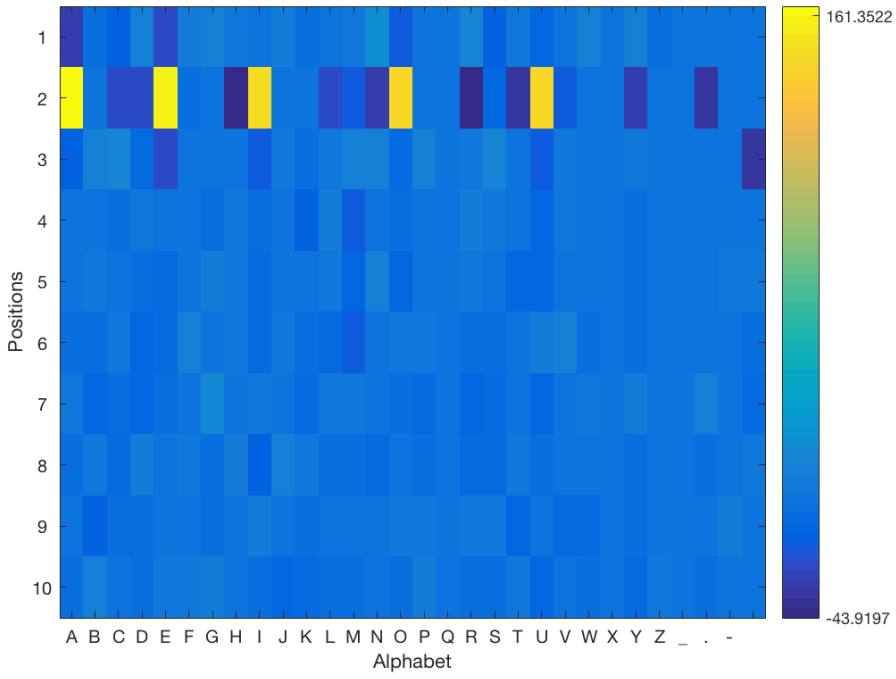


Figure 4: Original Alphabets and Positions

From the result, we could observe that $\delta \approx 0$, which means we found a linear separator for the data set. The train accuracy is 1, which also indicates that we successfully separate the data set. Besides, using this function also separates the test data set with no error, the accuracy is also 1. Figure 9 shows the weights for our features in the Alphabet vector. We could see that the letter 'A', 'E', 'I', 'O', 'U' have the largest weights compared with others at the 2nd position. Based on this feature, we could propose a hypothesis that the name with vowel at 2nd position will be classified as positive, otherwise, it will be negative.

4) To achieve 100% accuracy in training examples, but less than 100% on test data, I found two ways to achieve that.

First, we could remove letter 'U' at the Alphabets, then we could get result shown at Figure 5.

```
delta =

   1.6200e-12


accuracyInTrain =

     1


accuracyInTest =

    0.9574
```

Figure 5: Remove 'U' at Alphabets

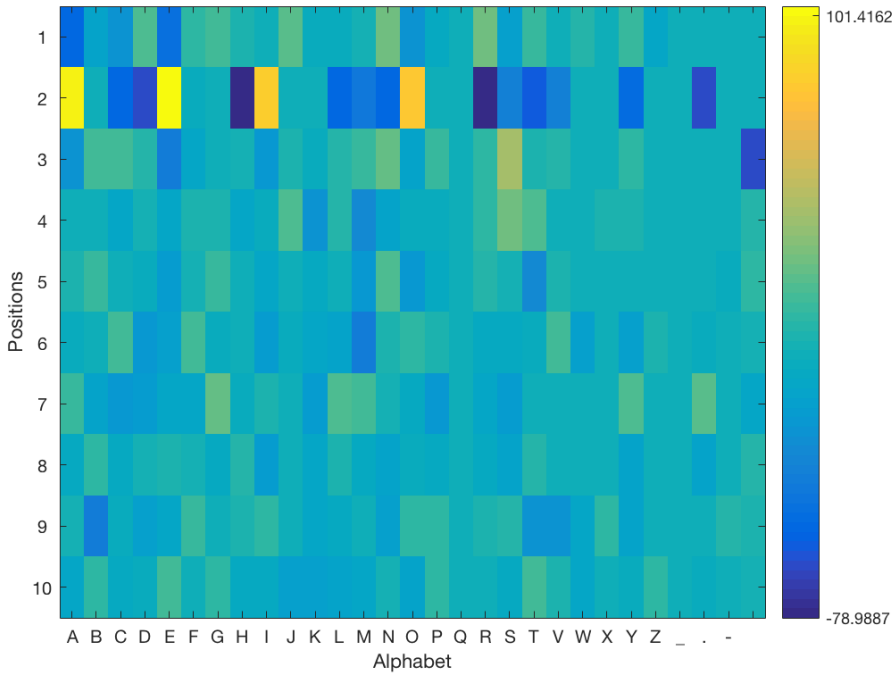


Figure 6: Weights graph when remove 'U' at Alphabets

We can get this result because we may have never seen the 'U' at 2nd position in our training data set, but we may have several cases at test data set, which explains why we cannot get 100% accuracy in the test data set.

Second, I found changing the positions could also generate the required result. Suppose I change the position from 1:10 to 3:10, then we can get:

```
delta =

   1.0232e-12


accuracyInTrain =

     1


accuracyInTest =

    0.7234
```

Figure 7: Change position to 3:10

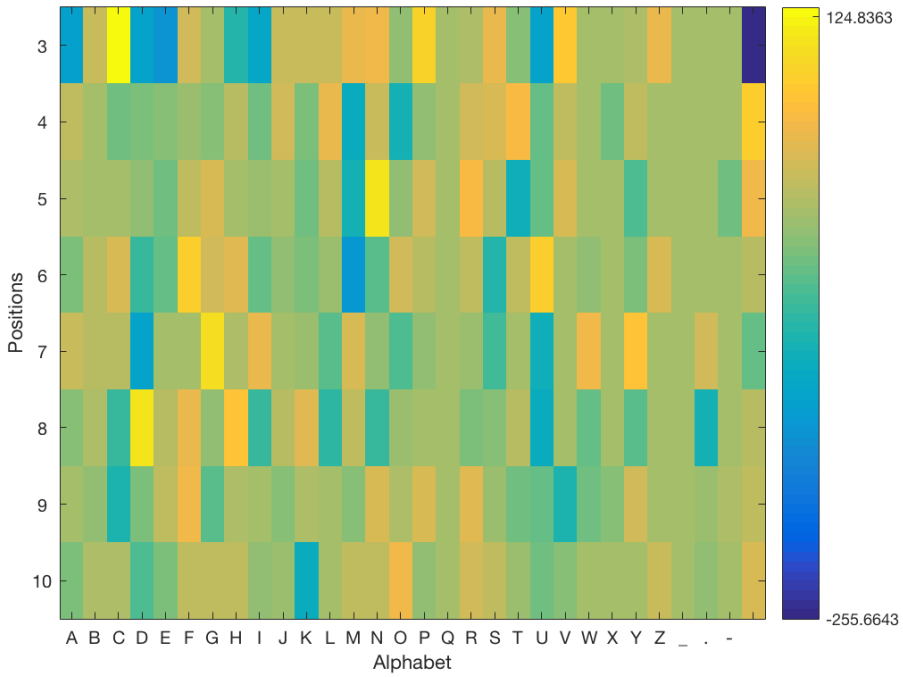


Figure 8: Weights graph when change position to 3:10

Therefore, from the Figure 8, we could conclude that the function finds a more complicated combination of the features to represent the desired linear

separator. We also have $\delta \approx 0$, which indicates it could separate the train data set. But the accuracy is lower at test data set, which indicates the separator could not completely separate the test data set, we may not choose the best separator.

iv. 1)The findLinearThreshold function is implemented as:

```
% 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
A = zeros(m+1, n+2);
b = zeros(m+1,1);
c = zeros(n+2,1);
c(n+2,1) = 1;
b(1:m,1) = 1;
% label of the training examples
y_i = data(:,np1);
for i = 1:m
    A(i,1:n) = y_i(i) * data(i,1:n);
end
A(1:m,n+1) = y_i(1:m);
A(1:m,n+2) = 1;
A(m+1,n+1) = 0;
A(m+1,n+2) = 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

theta = t(n+1);
delta = t(n+2);

end
```

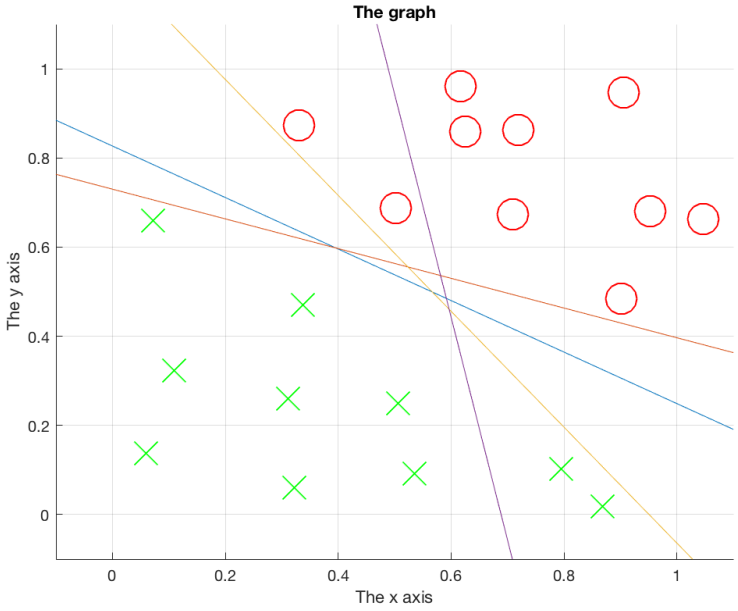


Figure 9: Line separators

2) From the output of learnMultipleHypothesis function, we get the Figure 9. We could see the blue one is the separator generated from the findLinearDiscriminant function, which is not given $\vec{w}$, so it has larger margin than others. So the other three lines(yellow, orange, purple) are generated by the given weights. From the graph, we could see the yellow, blue, orange lines could separate the data without errors. In other words, it could separate the data set exactly. But the purple line could not separate the data set. In addition, we could see that the yellow and orange lines are very close to positive or negative data points.

For yellow, blue and orange lines, the value of $\delta \approx 0$, which indicates it could separate the data set exactly.

As for which separator is better, basically speaking, the blue one maybe better because it has larger margin to the closest points on both sides. But we cannot say the other two separators(yellow and orange) are not good because we only have these data points, maybe they can do better job when it comes to lots of new training examples.

From the Figure 9, we found that the yellow and orange separators are also valid for separating the entire data set. So they might also be valid output from the process [2] to [4] in linear program. So we could infer that the solution is not unique.