CS446: Machine Learning **Spring 2017**

Problem Set 1

Hsiang-Wei Hwang *Handed In: February 3, 2017*

1. (a) I will use *Decision Tree* to deal with this problem. To choose a suitable attribute splitting the data set, entropy and information gain are introduced to find the highest purity of data after partitioning by the chosen attribute:

$$Entropy(S) = -p_+ log(p_+) - p_- log(p_-) \tag{1}$$

where p+ is the proportion of positive data in S (label = +1) and p- is the negative data (label = -1).

$$Gain(S, a) = Entropy(S) - \sum_{v \in values(a)} \frac{|S_v|}{|S|} Entropy(S_v) \tag{2}$$

where Sv is the subset of S for which attribute a has value v, and the entropy of partitioning the data is calculated by weighing the entropy of each partition by its size relative to the original set. In this case, **a** should be one of x_1, x_2 ..., x_n and the value v should be 0 or 1. The data could be partitioned by finding the largest information gain from x_1, x_2 ..., x_n. The algorithm works as following steps:

1. Calculate the entropy, e, of the input data set. If entropy is equal to 0, stop generating new nodes and set current node as leaf node.

```
for each example $d_i$ in S
  if( L($d_i$) == 0) then L1++; else L0++;
e = entropy(L1, L0);
if (e == 0) then thisLabel = (L0 == 0)? 1:0, return this;
```

2. Collect information gain of different attributes and find the attribute with max information gain.

```
for each attribute $x_j$
  for each example $d_i$ in S
    if ($x_i$ == 1) then List1.add($d_i$); else List0.add($d_i$) ;
  if ( $infoGain(e, List1, List0) > maxIG$ )
    maxX = $x_j$, $maxIG = infoGain(List1, List0)$;
```

3. Splitting the data set by the attribute with max information gain and generate new nodes for each labels

```
node n0 = new node(List0), node n1 = new node(List1);
thisLabel = -1, return this; // thisLabel = -1 $\rightarrow$ not leaf node
```

(b) Because we only consider a hypothesis space consisting of all *Conjunctions* over all attributes, decision tree would split the data set by using the attributes and digging into all combinations that the attributes can make. After digging all combinations of $x_1, x_2, ..., x_n$, the chosen attributes in all nodes of the decision tree can combine a logic equal to the expanded result of the target conjunction function. For example, the target conjunction is

$$(x1 = 1 \land x5 = 0 \land x7 = 1) \equiv (x1 \land \neg x5 \land x7)$$

Now, building decision tree by finding the highest information gain, we could easily find that the examples with label = +1 should all have attributes x1 = 1, x5 = 0, and x7 = 1. We could easily separate data by building the 3-layer decision tree using x1, x5, and x7 as the partitioning criteria.

(c) For data set with n attributes and m training examples, time complexity for each node is O(m') + O$(m'n)$ + O(2) where m' is the number of examples in the sub data set. The maximum number of examples in a layer should be m, therefore, the time complexity for each layer is O(mn). Besides, the max depth of a decision tree is n. Thus, the upper bound of the total time complexity is O(mn^2).

(d) Use the example in (b). If the information in the training data set is insufficient, like all x1 in the examples are 1s, the algorithm will be unable to generate the function as the target function. Furthermore, if we get a noisy input example, with the same attributes but different label from the training data set, the decision tree will be unable to generate the right label.

2. (a) $\vec{w}$ is the vector orthogonal to the hyperplane. Assume that the projection vector from the point, $\vec{x_0}$, to the hyper plane is $a\vec{w}$ which is the shortest path between the point and the hyperplane. Thus, the projection on the hyperplane is $\vec{x}_{project} = \vec{x_0} + a\vec{w}$. The distance is $|a| \times \|\vec{w}\|$. Substitute original formula to obtain the answer:

$$\vec{w}^T \vec{x}_{project} + \theta = 0 \Rightarrow \vec{w}^T(\vec{x_0} + a\vec{w}) + \theta = 0 \Rightarrow \vec{w}^T(\vec{x_0} + a\vec{w}) + \theta = 0 \Rightarrow \vec{w}^T\vec{x_0} + a\|\vec{w}\|^2 + \theta = 0$$

So,

$$|a| \times \|\vec{w}\| = \frac{|\vec{w}^T\vec{x_0} + \theta|}{\|\vec{w}\|}$$

(b) According to (a), with the point on the plane 1, $\vec{x_1}$, the distance between $\vec{x_1}$ and the plane 2 is $\frac{|\vec{w}^T\vec{x_1} + \theta_2|}{\|\vec{w}\|}$. Substituting the distance with the formula, $\vec{w}^T\vec{x_1} + \theta_1 = 0$, we get

$$distance = \frac{|-\theta_1 + \theta_2|}{\|\vec{w}\|}$$

.

3. (a.1) Firstly we prove following statement:

$$\text{If } y_i = \begin{cases} 1 & \text{if } \vec{w}^T\vec{x_i}+\theta \geq 0 \\ -1 & \text{if } \vec{w}^T\vec{x_i}+\theta < 0 \end{cases} \text{ then } (\vec{w'},\theta') \text{ exists that } y_i(\vec{w'}^T\vec{x_i}+\theta') \geq 1, \forall(\vec{x_i},y_i) \in D$$

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

$$\text{If } \theta < 0, \begin{cases} (\frac{1}{-\theta}\vec{w}^T)\vec{x_i} \geq 1 \Rightarrow y_i((\frac{1}{-\theta}\vec{w}^T)\vec{x_i}) \geq 1 \\ (\frac{1}{-\theta}\vec{w}^T)\vec{x_i} < 1 \Rightarrow y_i((\frac{1}{-\theta}\vec{w}^T)\vec{x_i}) \geq 1 \end{cases}$$

$$\text{If } \theta > 0, \begin{cases} (\frac{1}{\theta}\vec{w}^T)\vec{x_i} \geq -1 \Rightarrow (\frac{1}{\theta}\vec{w}^T)\vec{x_i}+2 \geq 1 \Rightarrow y_i((\frac{1}{\theta}\vec{w}^T)\vec{x_i}+2) \geq 1 \\ (\frac{1}{\theta}\vec{w}^T)\vec{x_i} < -1 \Rightarrow (\frac{1}{\theta}\vec{w}^T)\vec{x_i}+2 < 1 \Rightarrow y_i((\frac{1}{\theta}\vec{w}^T)\vec{x_i}+2) \geq 1 \end{cases}$$

$$\text{If } \theta = 0, \begin{cases} \vec{w}^T\vec{x_i} \geq 0 \Rightarrow \vec{w}^T\vec{x_i}+1 \geq 1 \Rightarrow y_i(\vec{w}^T\vec{x_i}+1) \geq 1 \\ \vec{w}^T\vec{x_i} < 0 \Rightarrow \vec{w}^T\vec{x_i}+1 < 1 \Rightarrow y_i(\vec{w}^T\vec{x_i}+1) \geq 1 \end{cases}$$

$\Rightarrow$ When $\theta < 0$, $(\vec{w'} = \frac{1}{-\theta}\vec{w}, \theta' = 0)$ exists that $y_i(\vec{w'}^T\vec{x_i}+\theta') \geq 1, \forall(\vec{x_i},y_i) \in D$.
When $\theta > 0$, $(\vec{w'} = \frac{1}{\theta}\vec{w}, \theta' = 2)$ exists that $y_i(\vec{w'}^T\vec{x_i}+\theta') \geq 1, \forall(\vec{x_i},y_i) \in D$.
When $\theta = 0$, $(\vec{w'} = \vec{w}, \theta' = 1)$ exists that $y_i(\vec{w'}^T\vec{x_i}+\theta') \geq 1, \forall(\vec{x_i},y_i) \in D$.
Thus,$\forall\theta \in R$ the first statement is proved.

Secondly we prove following statement:

$$\text{If } y_i(\vec{w'}^T\vec{x_i}+\theta') \geq 1, \text{ then } (\vec{w},\theta) \text{ exists that } y_i = \begin{cases} 1 & \text{if } \vec{w}^T\vec{x_i}+\theta \geq 0 \\ -1 & \text{if } \vec{w}^T\vec{x_i}+\theta < 0 \end{cases} \forall(\vec{x_i},y_i) \in D$$

$$y_i(\vec{w'}^T\vec{x_i}+\theta') \geq 1,\ y_i \in \{-1,1\} \Rightarrow$$

$$\begin{cases} y_i \geq \frac{1}{\vec{w'}^T\vec{x_i}+\theta'} > 0 & \text{if } \vec{w'}^T\vec{x_i}+\theta' > 0 \\ y_i \leq \frac{1}{\vec{w'}^T\vec{x_i}+\theta'} < 0 & \text{if } \vec{w'}^T\vec{x_i}+\theta' < 0 \\ \vec{w'}^T\vec{x_i}+\theta' \geq \frac{1}{y_i} & \text{if } y_i > 0 \\ \vec{w'}^T\vec{x_i}+\theta' \leq \frac{1}{y_i} & \text{if } y_i < 0 \end{cases} \Rightarrow$$

$$\because y_i \in \{-1,1\} \therefore \begin{cases} y_i = 1 & \text{if } \vec{w'}^T\vec{x_i}+\theta' > 0 \\ y_i = -1 & \text{if } \vec{w'}^T\vec{x_i}+\theta' < 0 \\ \vec{w'}^T\vec{x_i}+\theta' \geq 1 & \text{if } y_i = 1 \\ \vec{w'}^T\vec{x_i}+\theta' \leq -1 & \text{if } y_i = -1 \end{cases}$$

$$\Rightarrow \begin{cases} y_i = 1 & \text{if } \vec{w'}^T \vec{x_i} + \theta' \geq 0 \\ y_i = -1 & \text{if } \vec{w'}^T \vec{x_i} + \theta' < 0 \end{cases}$$

We prove the second statement. Combining first and second statements proved, the iff statement is proved.
For the hyperplanes with $\delta > 0$, we can say that there are must be an answer for $\delta = 0$ by scaling and shifting $\vec{w}$ and θ.

(a.2)

$$y_i(\vec{w}^T\vec{x_i} + \theta) \geq 1 - \delta_1,\ \delta_1 \geq 0 \Rightarrow y_i(\vec{w}^T\vec{x_i} + \theta) \geq -\delta_1,\ \delta_1 \geq 0$$

We already proved that $\delta = 0$ is the $min\delta$ solution for [(2)(3)(4)] which means $y_i(\vec{w}^T\vec{x_i} + \theta) \geq 1 \ \Rightarrow y_i(\vec{w}^T\vec{x_i} + \theta) \geq 0$. Thus, the solution to (a.2) is $min\delta = 0$. The reason we do not choose this as our optimal problem is that there is a deadlock at $(\vec{w} = 0, \theta = 0, \delta = 0)$ which could be an answer to any input data set. Thus, we choose $[(2)to(4)]$ to gather useful information.

(a.3)

$$\vec{w} = \begin{pmatrix} w_0 \\ w_1 \\ . \\ . \\ . \\ w_n \end{pmatrix} \Rightarrow \text{Apply two examples to (3)} \begin{cases} y_1(\sum_{i=0}^{n} w_i + \theta) \geq 1 - \delta \\ y_2(-\sum_{i=0}^{n} w_i + \theta) \geq 1 - \delta \end{cases} , \delta \geq 0$$

$$\Rightarrow \begin{cases} \sum_{i=0}^{n} w_i + \theta \geq 1 - \delta \\ \sum_{i=0}^{n} w_i - \theta \geq 1 - \delta \end{cases} , \delta \geq 0 \overset{add\ two\ inequalities}{\longrightarrow} 2 \times \sum_{i=0}^{n} w_i \geq 2 - 2\delta,\ \delta \geq 0$$

$$\Rightarrow \sum_{i=0}^{n} w_i + \delta \geq 1, \delta \geq 0 \Rightarrow \text{To achieve } min\ \delta = 0, \text{ we get } \sum_{i=0}^{n} w_i \geq 1$$

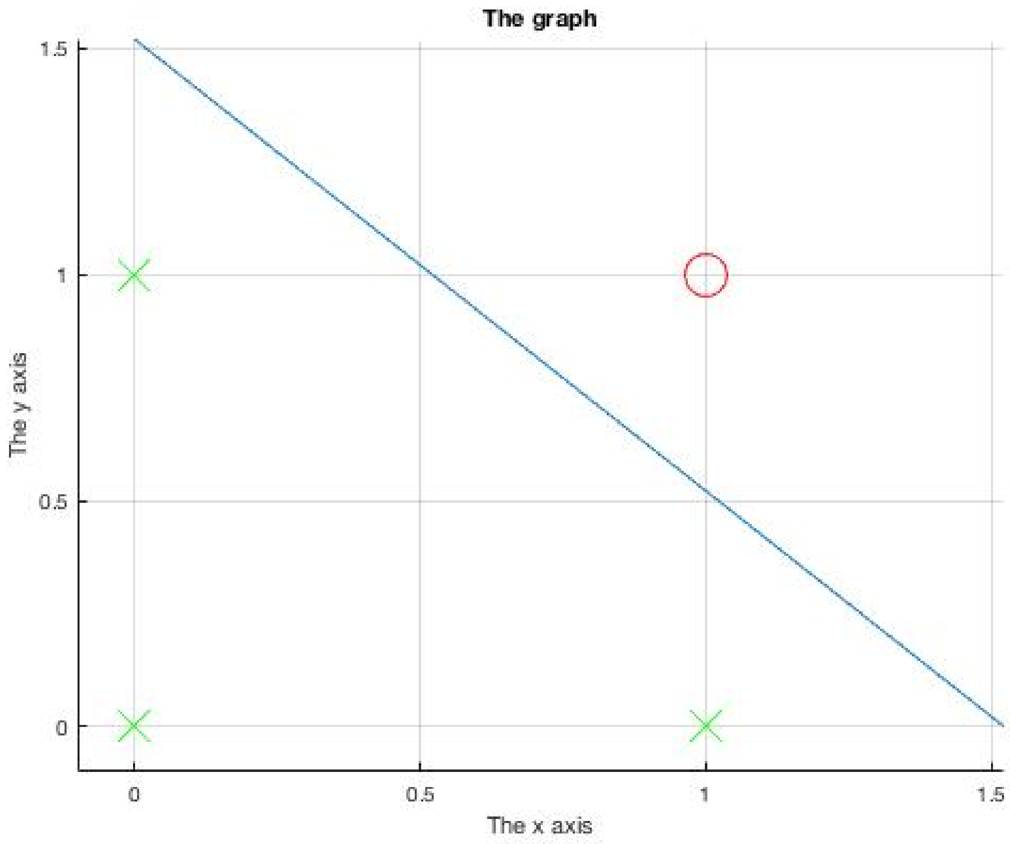
(b.1) We get:

$$\vec{c} = \begin{pmatrix} 0 \\ 0 \\ . \\ . \\ . \\ 0 \\ 0 \\ 1 \end{pmatrix}, A = \begin{pmatrix} y_1x_{11} & y_ix_{12} & . & . & . & y_ix_{1m} & y_1 & 1 \\ y_2x_{21} & y_2x_{22} & . & . & . & y_2x_{2m} & y_2 & 1 \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ y_mx_{m1} & y_mx_{m2} & . & . & . & y_mx_{mm} & y_m & 1 \\ 0 & 0 & . & . & . & 0 & 0 & 1 \end{pmatrix}, \vec{t} = \begin{pmatrix} w_1 \\ w_2 \\ . \\ . \\ . \\ w_m \\ \theta \\ \delta \end{pmatrix}, \vec{b} = \begin{pmatrix} 1 \\ 1 \\ . \\ . \\ . \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

(b.2) My data set is :

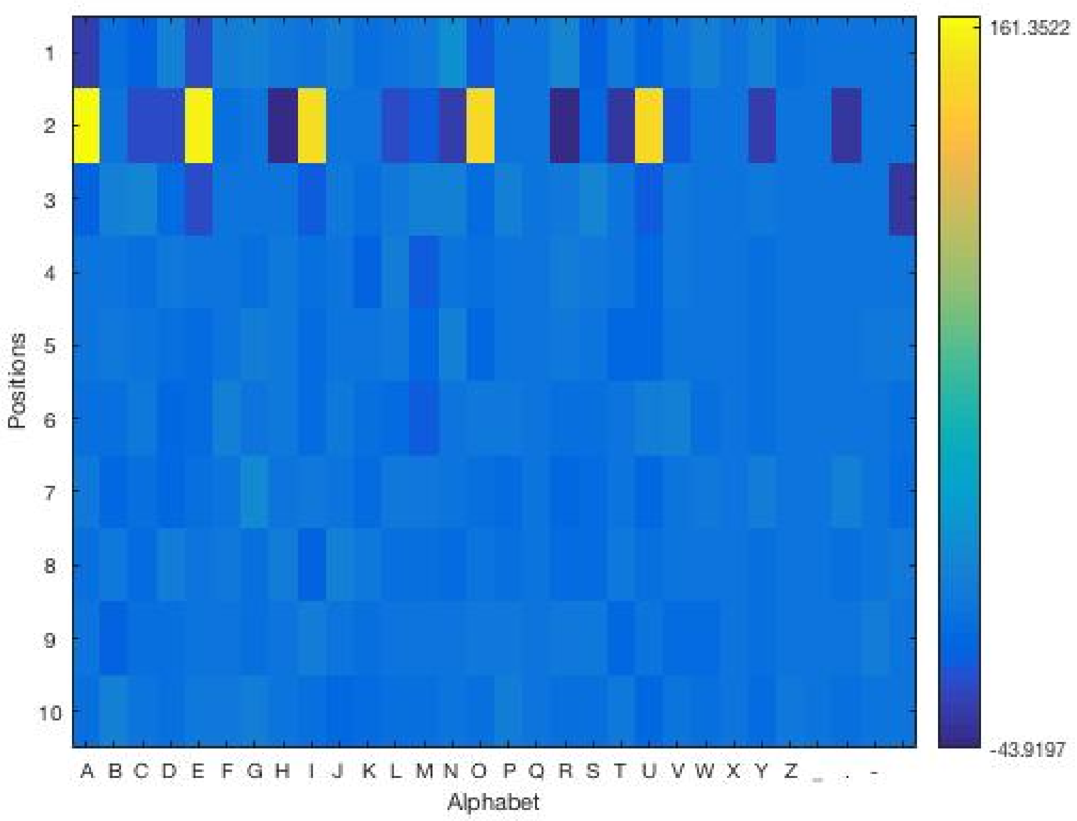
x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

Result is $\vec{w^T} = (156.2452, 156.2452),\ \theta = -237.6709,\ \delta = 8.3610e - 10.$

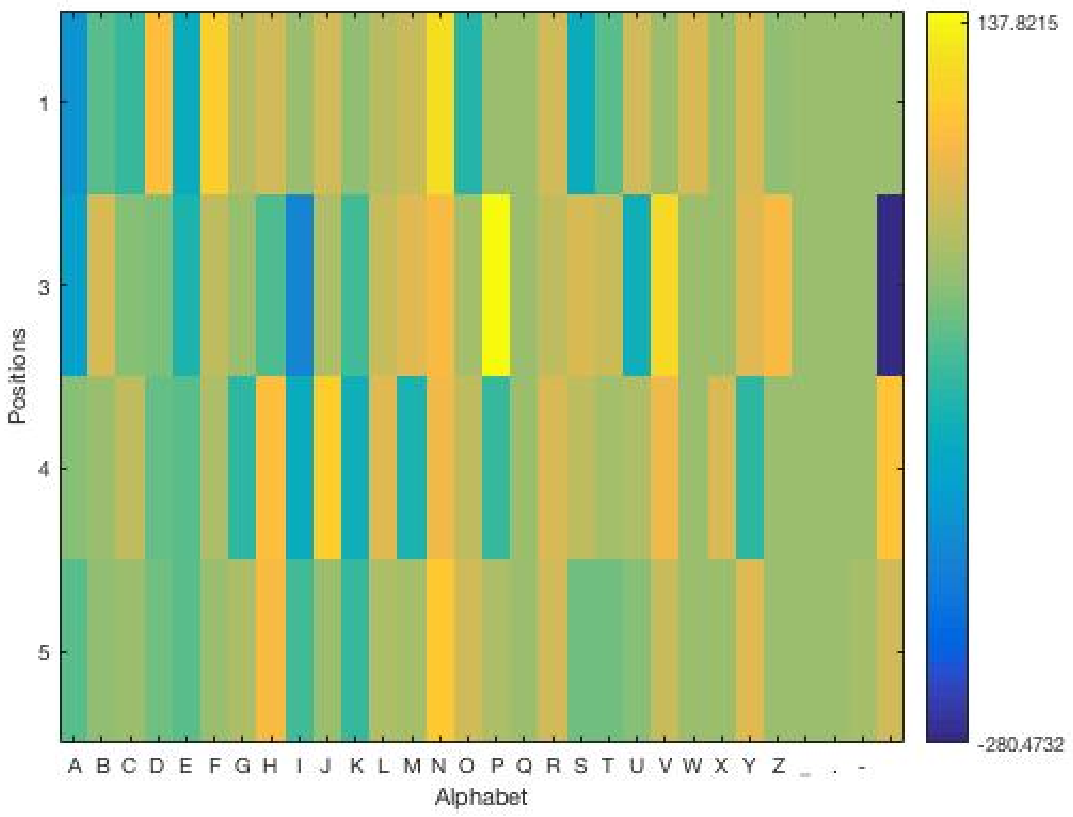


Result of `hw1conjunction.txt`:
$\vec{w^T} = (2.9104, -2.0498, 0.1775, 190.5196, 0.1399, -3.1010, -2.9534, -193.2778, 1.1679, -8.8942)$
θ = -90.2115
δ =-2.4158e-13 ≈ 0
According to the $\vec{w^T}$, we can infer the conjunction as $x_3 \wedge \neg x_7$ because w_3 is a large positive coefficient and w_7 is a small negative coefficient. Other w_i are nonzero but relatively small comparing to θ. Even we add all of $|w_i|$ except w_3 and w_7, the result is not large enough to change our decision. Besides, δ is almost equal to 0 which is our target of this linear programming problem. All the evidences show that the program works as we expect and can give us a right estimated answer.

(b.3) The result:
$\delta = 1.2619e - 09 \approx 0$, accuracyInTrain = 1, accuracyInTest =1
Here, we get δ is almost equal to 0 which means the linear programming problem was solved well. The accuracies are perfect which means the training data set is not noisy and the classifier we trained is perfect to the test data set.
The figure below shows the weighs we got. We can immediately find that the weighs of vowels at d = 2 are high in yellow and the others are mostly in blue, negative numbers.



Adjusting `positions = [1 3 4 5]`, we can get a perfect accuracy in the training data set, but an imperfect accuracy in the testing data set (accuracyInTrain = 1, accuracyInTest = 0.8404). Apparently, the figure looks random comparing to original one and there are still some patterns of high weighs which means that the algorithm finds a more complex way to fit our training data and to predict the test data.



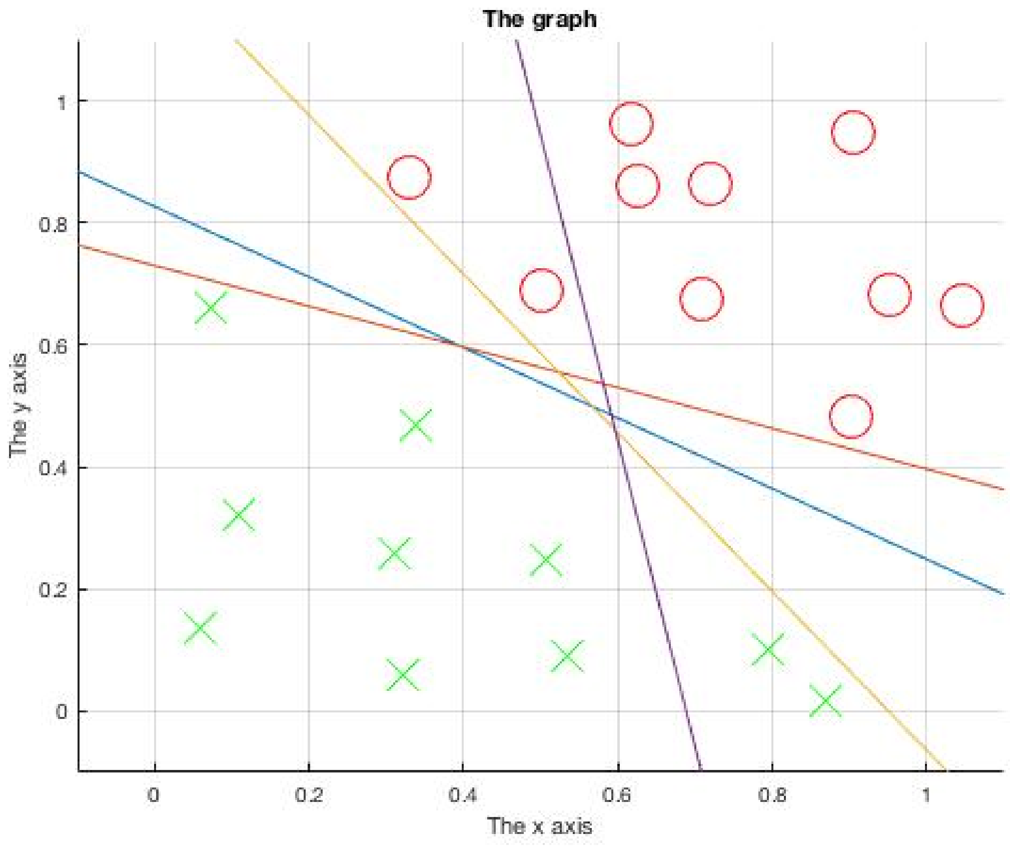
(b.4) There are two ways to achieve θ finding:

1. Setting a new linear program problem

$$\vec{c} = \begin{pmatrix} 0 \\ 1 \end{pmatrix},\ A = \begin{pmatrix} y_1 & 1 \\ y_2 & 1 \\ . & . \\ . & . \\ . & . \\ y_m & 1 \\ 0 & 1 \end{pmatrix}, \vec{t} = \begin{pmatrix} \theta \\ \delta \end{pmatrix},\ \vec{b} = \begin{pmatrix} 1 - y_1 \vec{w}^T \vec{x_1} \\ 1 - y_2 \vec{w}^T \vec{x_2} \\ . \\ . \\ . \\ 1 - y_m \vec{w}^T \vec{x_m} \\ 0 \end{pmatrix}$$

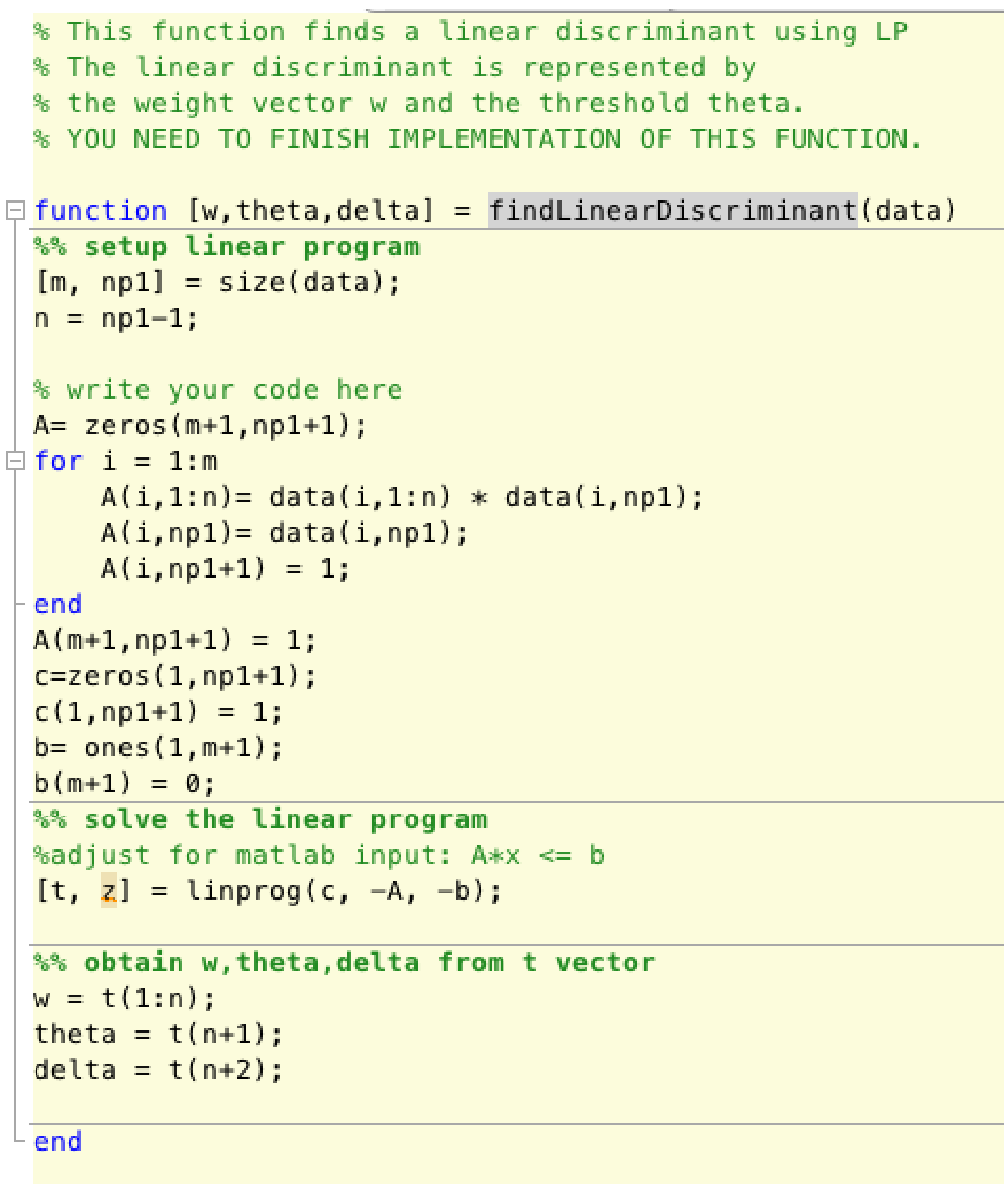
2. Giving constraints to Matlab function to make sure $\vec{w}$ will not change.

Here, I choose 2 because it is much more faster. Then, we get following figure.



In this script, we fix $\vec{w}$ and find θ to solve the linear program problem. We get 4 pair of (θ, δ), (-243.2630,-1.2506e-12), (-218.9946,-5.1159e-13), (-123.6116,4.2917e-11), (-344.3350,92.9650). First 3 hyperplanes successfully separate the data set and the δs are close to 0. However, The last hyperplane is apparently failed and the δ is not close to 0. Although the blue line with more margin between data points in different labels seems the best hyperplane separating the training data set, it is possible that there is an example in the testing data set that blue line cannot separate it right. Also, we can learn from this problem that there are several hyperplanes can separate our training data set. The solution to a linear program problem will not be unique.

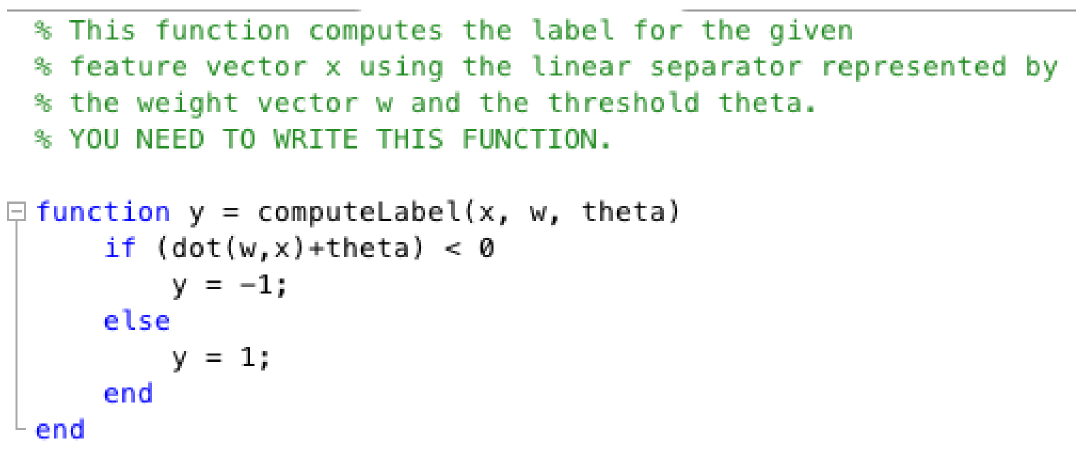
4. **Code snippets**
`findLinearDiscriminant.m`:



```
% 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
A= zeros(m+1,np1+1);
for i = 1:m
    A(i,1:n)= data(i,1:n) * data(i,np1);
    A(i,np1)= data(i,np1);
    A(i,np1+1) = 1;
end
A(m+1,np1+1) = 1;
c=zeros(1,np1+1);
c(1,np1+1) = 1;
b= ones(1,m+1);
b(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
```

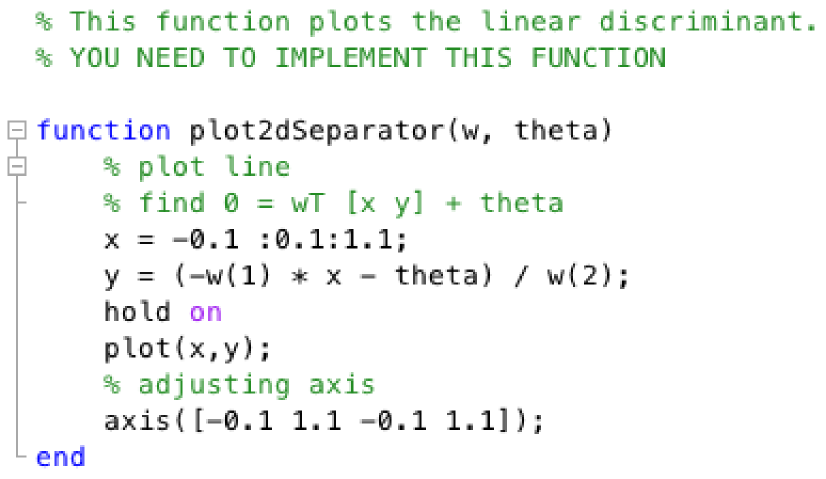
b

`computeLabel.m`:



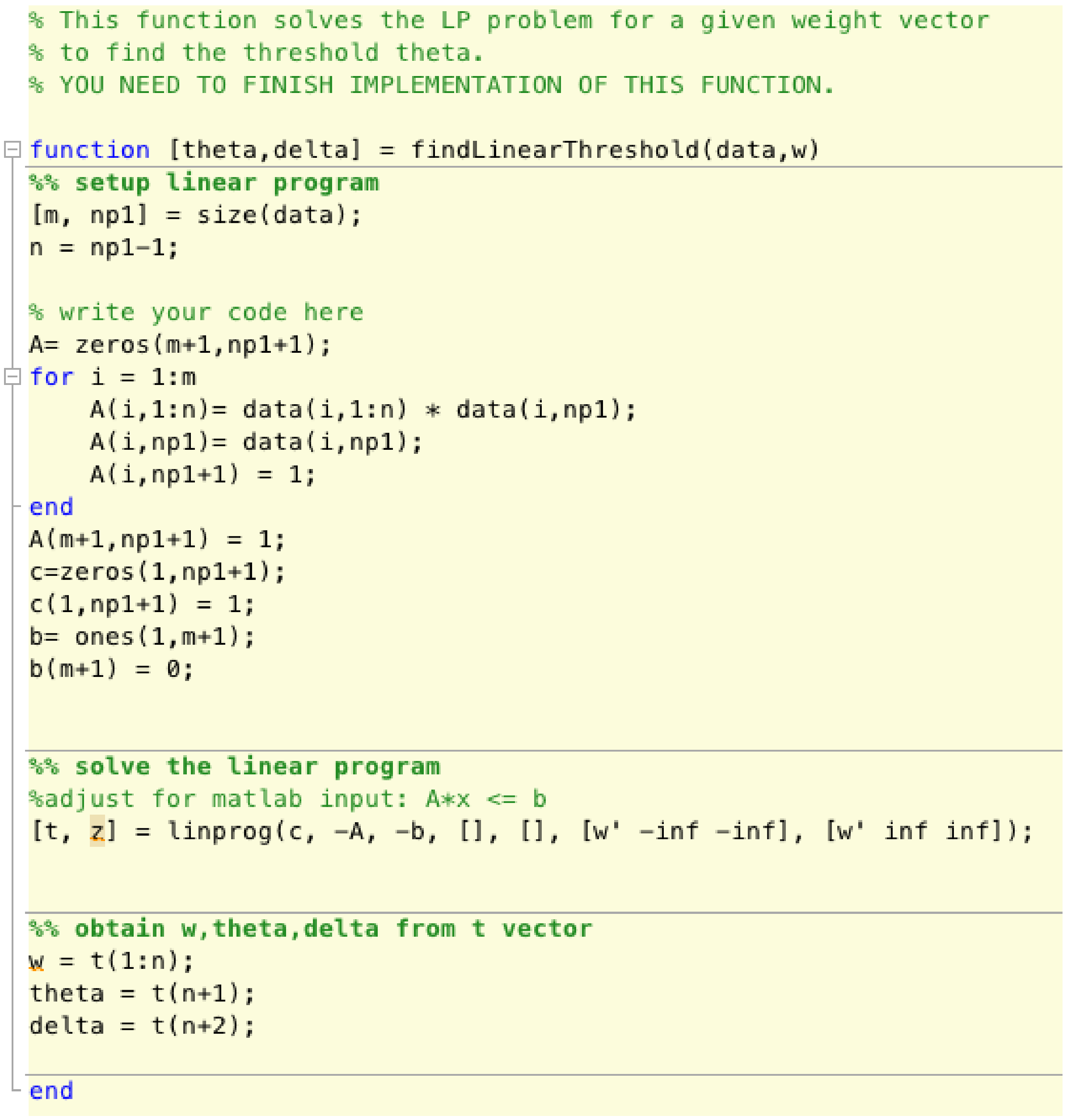
```
% This function computes the label for the given
% feature vector x using the linear separator represented by
% the weight vector w and the threshold theta.
% YOU NEED TO WRITE THIS FUNCTION.

function y = computeLabel(x, w, theta)
    if (dot(w,x)+theta) < 0
        y = -1;
    else
        y = 1;
    end
end
```

`plot2dSeparator.m`:



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

function plot2dSeparator(w, theta)
    % plot line
    % find 0 = wT [x y] + theta
    x = -0.1 :0.1:1.1;
    y = (-w(1) * x - theta) / w(2);
    hold on
    plot(x,y);
    % adjusting axis
    axis([-0.1 1.1 -0.1 1.1]);
end
```

`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
A= zeros(m+1,np1+1);
for i = 1:m
    A(i,1:n)= data(i,1:n) * data(i,np1);
    A(i,np1)= data(i,np1);
    A(i,np1+1) = 1;
end
A(m+1,np1+1) = 1;
c=zeros(1,np1+1);
c(1,np1+1) = 1;
b= ones(1,m+1);
b(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
```