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

# Problem Set 1

Kaizhao Liang *Handed In: January 31, 2017*

1. Learning Conjunctions

   a. My algorithm uses the property of the conjunction, updating the weights by elimination with the positive samples and then verifies them with the negative examples. First of all, my algorithm augments the feature space into 2n, adding the negations of the original features, which makes updating weights more convenient. By the property of conjunction, positive samples could not tolerate any negative feature. In order to maintain the correctness of the prediction, it needs to eliminate the features that are of value 0, i.e. , setting the corresponding weights to 0. In addition, the predictions produced by the remaining features have to be consistent with every label in the negative set. Otherwise, there is not a single conjunction that can satisfiy this data set.

   **Possible Pseudocode:**

   Augment the feature space to 2n, where the last n features represent the first n features' negation.
   Initialize a weight vector w of length 2n with all 1's.
   Separate the data set into two sets with one of only positive samples $S_+$ and other of only negative samples $S_-$.
   FOR EACH sample $s$ in $S_+$:
   FOR EACH feature f in s:
   IF f==0 THEN
   w[f.index]=0
   ENDIF
   ENDFOR
   ENDFOR
   $\theta$ = number of the nonzero elements in w
   FOR EACH sample s in $S_-$
   prediction= sign of $w^T s - \theta$
   IF prediction!=s.label THEN
   PRINT OUT "inconsistency message"
   BREAK
   ENDIF
   ENDFOR

   b. This algorithm either succeeds to find a conjunction or stops when there is no conjunction to satisfy the data set. If it outputs a conjunction, it must be correct, because the elimination, the first loop, guarantees that it will predict all the negative samples right and the verification, the second loop guarantees that it will predict all the positive samples right. If it fails to output a conjunction,

it means that there does not exist any conjunction that is consistent with the dataset. Since it's not possible to modify the weights in the verfication without violating the previous samples in the positive set, if the prediction violates the label in the positive set, it coud be concluded that it's not possible to generate a conjunction that fits all the data. Thereby, the correctness of the algorithm is proven.

c. Here m is the number of samples and n is the number of features in each sample. Augmenting the feature space takes O(mn). Separating the dataset into two sets takes O(m). Elimination loop takes O(mn). Finding the threshod $\theta$ takes O(n). Verfication loop also takes O(mn), assuming that the matrix multiplication takes O(n). They added up to O(mn+n). In conclusion, the Running time is O(mn), which is polynomial.

d. There might be multiple conjunctions that are consistent with the training data. Since this is an elimination algorithm, it always find the most complicated one, which might include irrelevant features that might lead to an inaccurate prediction for the new data. However, the algorithm should be able to converge, given enough positive samples. As the positive training data grows, the confidence for the hypothesis for the unseen data also grows, when more features are eliminated and the hypothese gets closer to the targeted conjunction.

2. Linear Algebra Review

a. **S**uppose there is a point $x_0$ on the hyperplane such that the $\|x_0 - x_1\|^2$ is minimized, which means either $x_0 = x_1$ or that the vector $(x_0 - x_1)$ is perpendicular to the hyperplane:

$$x_0 - x_1 = \lambda w$$
$$\|x_0 - x_1\| = \lambda \|w\|$$

where $\lambda$ is a constant.
Since $x_1$ is on the hyperplane:

$$w^T x_1 + \theta = 0$$

and

$$x_1 = x_0 - \lambda w$$

so

$$w^T(x_0 - \lambda w) + \theta = 0$$
$$\|w^T x_0 + \theta\| = \lambda \|w^T w\|$$

Combining the above equations we have:

$$d = \|x_0 - x_1\| = \lambda \|w\| = \frac{\|w^T x_0 + \theta\|}{\|w\|}$$

b. **F**irst of all, the two hyperplanes are parallel, because of they share the same $w$ and since they are parallel, distance between any point of one plane and the other plane is also the distance between the two planes.
Picking random point $x_0$ on plane $w^T x + \theta_1 = 0$, we have:

$$w^T x_0 + \theta_1 = 0$$

i.e.

$$w^T x_0 = -\theta_1$$

Applying the analytical solution to distance between $x_0$ and plane $w^T x + \theta_2 = 0$:

$$d = \frac{\|w^T x_0 + \theta_2\|}{\|w\|}$$

Substituting the $w^T x_0$ with $-\theta_1$:

$$d = \frac{\|\theta_2 - \theta_1\|}{\|w\|}$$

3. Finding a Linear Discriminant Function via Linear Programming

a.1. **Proof**: Suppose that there exists a plane with w and $\theta$, such that $\delta$=0. Consider the positive sample $x_+$ and the negative sample $x_-$ that are closest to the plane. Since $y_+(w^T x_+ + \theta) \geq 1$ and $y_-(w^T x_- + \theta) \geq 1$,

$$y_+(w^T x_+ + \theta) \geq 1$$

$$y_-(w^T x_- + \theta) \geq 1$$

By the definition of the linear separability, $y_i(w^T x_i + \theta) \geq 0$
Both samples satisfy the above inequalities, hereby that if $\theta = 0$, the data set is linearly separable is proven
Suppose that there exists a hyperplane $w^T x + \theta = 0$ satisfying the condition $y_i(w^T x_i + \theta) \geq 1 - \delta$:

$$y_i(w^T x_i + \theta) - 1 \geq -\delta$$

Then,

$$\frac{y_i(w^T x_i + \theta) - 1}{c} \geq \frac{-\delta}{c}$$

where c is a positive constant.
Since $w^T x_i + \theta - 1 = 0$ and $\frac{w^T x_i + \theta - 1}{c} = 0$ are exactly the same plane, and $-w^T x_i - \theta - 1 = 0$ and $\frac{-w^T x_i - \theta - 1}{c} = 0$ are also the same plane, two conditions, $y_i = 0$ and $y_i = 1$, are combined to give:

$$\frac{y_i(w^T x_i + \theta) - 1}{c} \geq -\delta$$

Since c could be any positive number and the assumption is that the data is linearly separable, which means the this hyperplane exists:

$$-\delta = -\frac{\delta}{c}$$

i.e. $\delta = 0$
Hereby that the data is linearly separable iff the $\delta$ could be optimized to zero is proven.

If there exists a hyperplane that satisfies the condition (3) with $\delta > 0$, the linear separability of the data is uncertain.
If $\delta \leq 1$ then the data set is definitely linearly separable and the $\delta$ can be minimized to 0.
If $\delta \geq 1$ then whether the data set is linear separable is not clear, since $\delta$ could be further minimized.

a.2. The trivial optimal solution is

$$w = zeros$$

$$\theta = 0$$

The above solution is correct but useless. To prevent the algorithm from getting this result, the formulation adds some constant c in front of $-\delta$. The $c-\theta$ is greater than zero and prevents the left side from converging to zero, thus avoiding the trivial solution. While the value of c does not make great differences as long as it's greater than 0, it only influences the margin.

a.3. Given $y_i(w^T x_i + \theta) \geq 1 - \delta$ and $\delta = 0$ where ( $x_1 = [1, 1, ..., 1]$,$y_1 = 1$ ) and ( $x_2 = [-1, -1, ..., -1]$,$y_2 = -1$ ),

$$y_1(w^T x_1 + \theta) \geq 1$$

$$y_2(w^T x_2 + \theta) \geq 1$$

i.e.

$$\sum w_i + \theta \geq 1$$

$$\sum w_i - \theta \geq 1$$

Combining the above inequality, it gives solutions:

$$\sum w_i \geq 1$$

and

$$\theta \geq 1 - \sum w_i$$

b.1. Rewritting the learning problem:
Given $y_i(w^T x_i + \theta) \geq 1 - \delta$, it gives

$$y_i(w^T x_i + \theta) + \delta \geq 1$$

Combining above inequality with $\delta \geq 0$:

$$A = [[y_i x_i, y_i, 1]; [0, 0, 0, ..., 1]]$$

$$t = [w; \theta; \delta]$$

$$b = [1, 1, 1, ..., 0]$$

where b is of dimension (m+1)x1, and [0,0,0,...,1] is of dimension (n+1)x1. So $z(t) = c^T t = \delta$, which is what needs to be minimized.

**Snippet of function findLinearDiscriminant:**

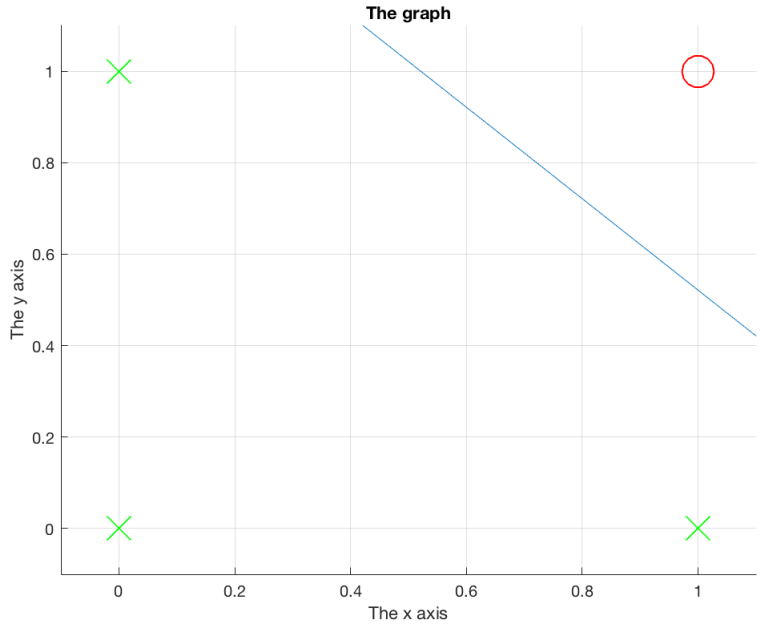
```
function [w,theta,delta] = findLinearDiscriminant(data)
%% setup linear program
[m, np1] = size(data);
n = np1-1;

% write your code here
% c'*t=delta,which is what we want to minimize,given t=[w1,w2,...wn,theta,delta]
c=zeros(n+2,1);
c(n+2,1)=1;
% y(w'x+theta)+delta>=1 ==> w1*y*x1+w2*y*x2+...+wn*y*xn+y*theta+delta>=1
% diag() creates a matrix that puts all the y on diagonal
A=[[diag(data(:,np1))*data(:,1:n),data(:,np1);zeros(1,np1)],ones(m+1,1)];
% b is just a vector of m ones, since y(w'x+theta)+delta>=1; the zero last
% element is for theta>=0
b=ones(m+1,1);
b(m+1,1)=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. **The data generated:**
1 1 1
1 0 -1
0 1 -1
0 0 -1
**The figure generated by this data set:**




**Snippet of function plot2dSeparator:**

```
function plot2dSeparator(w, theta)
    x1=0:0.1:1.5;
    x2= (-w(1)*x1-theta)/w(2);
    plot(x1,x2);
end
```

**Running linear program on hw1conjunction data set outputs:**

$$w = [2.9104, -2.0498, 0.1775, 190.5196, 0.1399, -3.1010, -2.9534, -193.2778, 1.1679, -8.8942]$$

$$\theta = -90.2115$$

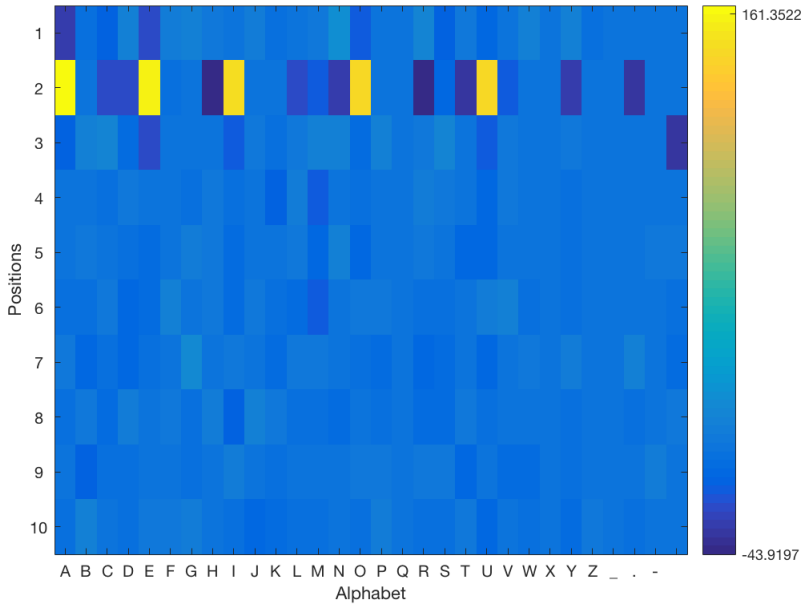
$$\delta = -2.4158 \times 10^{-13}$$

From the weight above, it's obvious that the conjunction is $x_4 \bigwedge \neg x_8$, since other weights are relatively less significant, while the weights of $x_4$ and $x_8$ are exceptionally large, and the sign of the weight determines whether it's negation or not. It's also obivous that $\delta \approx 0$, indicating that the linear program succeeded in separating the data. The threshod $\theta \approx \frac{\|w_4\|}{2} \approx \frac{\|x_8\|}{2}$ indicates that the program finds the 'best' hyperplane with largest margin possible.

b.3. **Snippet of function computeLabels:**

```
function y = computeLabel(x, w, theta)
    y=sign(x'*w+theta);
    y(y==0)=1;
end
```

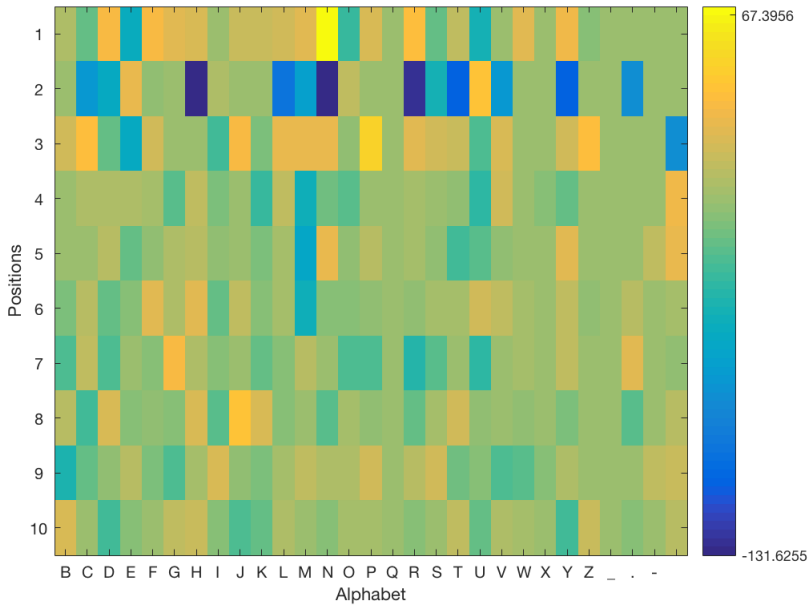
**The figure generated:**




$\delta = 1.2619 \times 10^{-9}$ and $accuarcyInTrain = 1, accuarcyInTest = 1$

From the figure above, it is clear that the A, E, I, O, U at position 2 are highlighted, which means that those are the signficant features. $\delta \approx 0$ indicates the success of the seperation of the data set. The accuarcyInTrain and accuarcyInTest are both 1, i.e. 100%, which implies high confidence in the hypothesis learned.

**After alphabat is changed to BCDEFGHIJKLMNOPQRSTUVWXYZ_˙- and position=1:10, figure generated:**



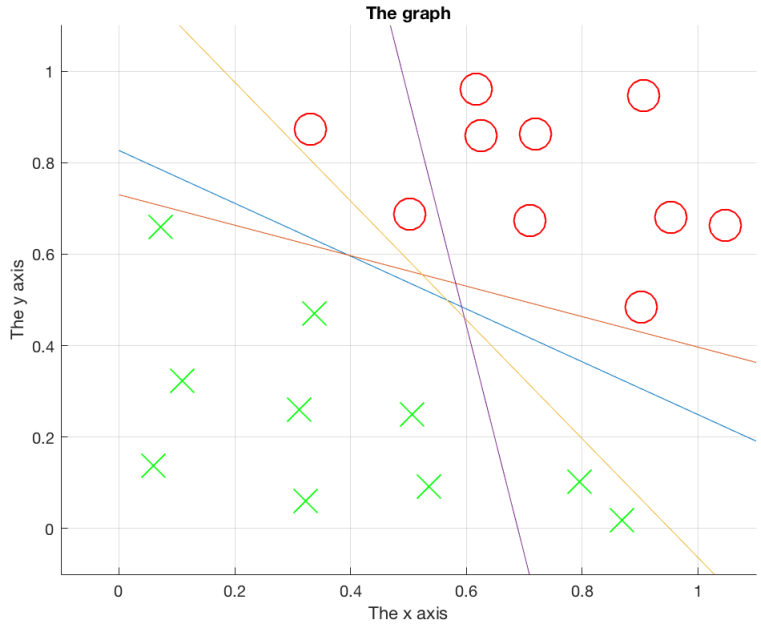

$\delta = 4.8317 \times 10^{-12}$ and $accuarcyInTrain = 1, accuarcyInTest = 0.9681$

The figure above is much noiser than the figure obtained before and the weights are more evenly distributed, which means that it's overfitting the training data. However, it still managed to identify two important features, E, U at position 2, and to retreive certain kind of structure that enables it do perform decently in test set.

b.4. **Snippet of function findLinearThreshold:**

```
function [theta,delta] = findLinearThreshold(data,w)
%% setup linear program
[m, np1] = size(data);
n = np1-1;
% write your code here
% c'*t=delta,which is what we want to minimize,given t=[w1,w2,...wn,theta,delta]
c=zeros(n+2,1);
c(n+2,1)=1;
% y(w'x+theta)+delta>=1 ==> w1*y*x1+w2*y*x2+...+wn*y*xn+y*theta+delta>=1
% diag() creates a matrix that puts all the y on diagonal
A=[[diag(data(:,np1))*data(:,1:n),data(:,np1);zeros(1,np1)],ones(m+1,1)];
% b is just a vector of m ones, since y(w'x+theta)+delta>=1; the zero last
% element is for theta>=0
b=ones(m+1,1);
b(m+1,1)=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
```

**The figure generated:**




$$\delta_1 = -1.2506 \times 10^{-12}, \theta_1 = -243.2630$$

$$\delta_2 = -5.1159 \times 10^{-13}, \theta_2 = -218.9946$$

$$\delta_3 = 4.2917 \times 10^{-11}, \theta_3 = -123.6116$$

$$\delta_4 = 92.9650, \theta_4 = -344.3350$$

From the above result, it's obvious that except for the purple plane, the other planes successfully separate the data. The $\delta_4$ belongs to the purple hyperplane that fails to separate the data, while the other hyperplanes successfully separate the data and minimize the $\delta$s to approximately zero. The $\delta$ value tells us whether there exist a threshod, such that the hyperplane could separate the data, given a weight. If $\delta$ could be minimize to 0, there exists a threshod that could enable the hyperplane to separate the data, otherwise, there is no solution.

Empirically, the blue hyperplane is the best, since it has the largest margin, which presumably is mroe generalized and is more likely to predict correctly when encountering a new unseen sample, while the others have too strict a condition and very likely to misclassify.

From the above examples, it's evident that the solution of the LP is not unique. Given different weights, the $\delta$ could all be minimizd to approximately 0. So all the weights that satisfy this condition could be solution.