CS446: Machine Learning **Spring 2017**

Problem Set 1

Handed Out: Jan 25th, 2017 *Due: Feb 3rd, 2017*

- Feel free to talk to other members of the class in doing the homework. I am more concerned that you learn how to solve the problem than that you demonstrate that you solved it entirely on your own. You should, however, write down your solution yourself. Please try to keep the solution brief and clear.
- Please use Piazza first if you have questions about the homework. Also feel free to send us e-mails and come to office hours.
- Please, no handwritten solutions. You will submit your solution manuscript as a single pdf file. Please use LaTeXto typeset your solutions. We have provided resources to get you started including homework templates. In addition, you also have to submit two output files for question 3.
- Please present your algorithms in both pseudocode and English. That is, give a precise formulation of your algorithm as pseudocode and *also* explain in one or two concise paragraphs what your algorithm does. Be aware that pseudocode is much simpler and more abstract than real code.
- The homework is due at 11:59 PM on the due date. We will be using Compass for collecting the homework assignments. Please submit your solution manuscript as a pdf file via Compass (`http://compass2g.illinois.edu`). Please do NOT hand in a hard copy of your write-up. Contact the TAs if you are having technical difficulties in submitting the assignment.

1. [Learning Conjunctions - 40 points] (Based on Mitchell, exercise 2.9)
   Consider a learning problem where each instance is a Boolean vector over n variables $(x_1, \ldots, x_n)$ and is labeled either positive (1) or negative (-1). Thus, a typical instance would be

   $$(1, 0, \ldots, 1) = (x_1 = 1, x_2 = 0, \ldots, x_n = 1)$$

   Now consider a hypothesis space **H** consisting of all *Conjunctions* over these variables. For example, one hypothesis could be

   $$(x_1 = 1 \wedge x_5 = 0 \wedge x_7 = 1) \equiv (x_1 \wedge \neg x_5 \wedge x_7)$$

   For example, an instance $(1, 0, 1, 0, 0, 0, 1, \ldots, 0)$ will be labeled as positive (1) and $(0, 0, 1, 0, 1, 1, 0, \ldots, 1)$ as negative (-1), according to the above hypothesis.

   a. Propose an algorithm that accepts a sequence of labeled training examples and outputs a hypothesis $h \in \mathbf{H}$ that is consistent with all the training examples, if one exists. If there are no consistent hypotheses, your algorithm should indicate as such and halt. [10 points]

   b. Prove the correctness of your algorithm. That is, argue that the hypothesis it produces labels all the training examples correctly. [10 points]

   c. Analyze the complexity of your algorithm as a function of the number of variables (n) and the number of training examples (m). Your algorithm should run in time polynomial in n and m (if it fails to, you should rethink your algorithm). [10 points]

d. Assume that there exists a conjunction in **H** that is consistent with all of the training examples. Now, you are given a new, unlabeled example that is not part of your training set. What can you say about the ability of the hypothesis derived by your algorithm in (*a.*) to produce the correct label for this example? [10 points]

2. [Linear Algebra Review - 10 points]
Assume that $\vec{w} \in \mathbb{R}^n$ and θ is a scalar. A hyper-plane in $\mathbb{R}^n$ is the set, $\{\vec{x} : \vec{x} \in \mathbb{R}^n, \vec{w}^T\vec{x} + \theta = 0\}$.

   a. **[5 points]** The distance between a point $\vec{x}_0 \in \mathbb{R}^n$ and the hyperplane $\vec{w}^T\vec{x}+\theta = 0$ can be described as the solution of the following optimization problem:

$$\begin{aligned} \min_{\vec{x}} \quad & \|\vec{x}_0 - \vec{x}\|^2 \\ \text{subject to} \quad & \vec{w}^T\vec{x} + \theta = 0 \end{aligned}$$

   However, there is an analytical solution (i.e. closed form solution) to find the distance between the point $\vec{x}_0$ and the hyperplane $\vec{w}^T\vec{x} + \theta = 0$. Derive the solution.

   b. **[5 points]** Assume that we have two hyper-planes, $\vec{w}^T\vec{x}+\theta_1 = 0$ and $\vec{w}^T\vec{x}+\theta_2 = 0$. What is the distance between these two hyper-planes?

3. [Finding a Linear Discriminant Function via Linear Programming - 50 points]
The goal of this problem is to develop a different formulation for the problem of learning linear discriminant functions and use it to solve the problem of learning conjunctions and the badges problem discussed in class. Some subproblems to question 3 may ask you to compose multiple programs or respond to multiple issues. Anything you are expected to **program, turn in, or write about in your report** will be in boldface.

   Definition 1 (Linear Program) *A* linear program *can be stated as follows:*

   Let A be an $m \times n$ real-valued matrix, $\vec{b} \in \mathbb{R}^m$, and $\vec{c} \in \mathbb{R}^n$. Find a $\vec{t} \in \mathbb{R}^n$, that minimizes the linear function

$$\begin{aligned} & z(\vec{t}) = \vec{c}^{\,T}\vec{t} \\ \textit{subject to} \quad & A\vec{t} \geq \vec{b} \end{aligned}$$

   In the linear programming terminology, $\vec{c}$ is often referred to as the *cost vector* and $z(\vec{t})$ is referred to as the *objective function.*[1] We can use this framework to define the problem of learning a linear discriminant function.[2]

[1]Note that you don't need to really know how to solve a linear program since we will use Matlab to obtain the solution (although you may wish to brush up on Linear Algebra). Also see the appendix for more information on linear programming.

[2]This discussion closely parallels the linear programming representation found in *Pattern Classification*, by Duda, Hart, and Stork.

The Learning Problem:[3] Let $\vec{x_1}, \vec{x_2}, \ldots, \vec{x_m}$ represent m samples, where each sample $\vec{x_i} \in \mathbb{R}^n$ is an n-dimensional vector, and $\vec{y} \in \{-1, 1\}^m$ is an $m \times 1$ vector representing the respective labels of each of the m samples. Let $\vec{w} \in \mathbb{R}^n$ be an $n \times 1$ vector representing the weights of the linear discriminant function, and θ be the threshold value.

We *predict* $\vec{x_i}$ to be a *positive* example if $\vec{w}^T\vec{x_i} + \theta \geq 0$. On the other hand, we *predict* $\vec{x_i}$ to be a *negative* example if $\vec{w}^T\vec{x_i} + \theta < 0$.

We hope that the learned linear function can separate the data set. That is, for the true labels we hope

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

In order to find a good linear separator, we propose the following linear program:

$$\min_{\vec{w},\theta,\delta} \quad \delta \tag{2}$$

$$\text{subject to} \quad y_i(\vec{w}^T\vec{x_i} + \theta) \geq 1 - \delta, \qquad \forall(\vec{x_i}, y_i) \in D \tag{3}$$

$$\delta \geq 0 \tag{4}$$

where D is the data set of all training examples.

a. Understanding linear separability *[15 points]*

a.1 *[8 points]* A data set $D = \{(\vec{x_i}, y_i)\}_{i=1}^m$ that satisfies condition (1) above is called *linearly separable*. **Show that the data set D is linearly separable if and only if there exists a hyperplane $(\vec{w'}, \theta')$ that satisfies condition** (3) **with** $\delta = 0$ (Need a hint?[4]). Conclude that D is linearly separable iff the optimal solution to the linear program [(2) to (4)] attains $\delta = 0$. **What can we say about the linear separability of the data set if there exists a hyperplane that satisfies condition** (3) **with** $\delta > 0$**?**

a.2 *[2 points]* **Show that there is a trivial optimal solution for the following linear program:**

$$\min_{\vec{w},\theta,\delta} \quad \delta$$

$$\text{subject to} \quad y_i(\vec{w}^T\vec{x_i} + \theta) \geq -\delta, \qquad \forall(x_i, y_i) \in D$$

$$\delta \geq 0$$

Show the optimal solution and use it to (very briefly) explain why we chose to formulate the linear program as [(2) **to** (4)] **instead.**

[3]Note that the notation used in the Learning Problem is **unrelated** to the one used in the Linear Program definition. You may want to figure out the correspondence.

[4]**Hint:** Assume that D is linearly separable. D is a finite set, so it has a positive example that is *closest* to the hyperplane among all positive examples. Similarly, there is a negative example that is *closest* to the hyperplane among negative examples. Consider their distances and use them to show that condition (3) holds. Then show the other direction.

a.3 *[5 points]* Let $\vec{x_1} \in \mathbb{R}^n$, $\vec{x_1}^T = \begin{bmatrix}1 & 1 & \dots & 1\end{bmatrix}$ and $y_1 = 1$. Let $\vec{x_2} \in \mathbb{R}^n$, $\vec{x_2}^T = \begin{bmatrix}-1 & -1 & \dots & -1\end{bmatrix}$ and $y_2 = -1$. The data set D is defined as

$$D = \{(\vec{x_1}, y_1), (\vec{x_2}, y_2)\}.$$

Consider the formulation in [(2) to (4)] applied to D. **Show the set of all possible optimal solutions (solve this problem by hand)**.

b. Solving linear programming problems *[35 points]*

This problem requires developing some Matlab code. We have provided you a collection of Matlab files. You will need to edit some of these files to answer the following questions. **In addition to your answers to the following questions, include as part of your solution manuscript the Matlab code you wrote in**

- `findLinearDiscriminant.m`
- `computeLabel.m`
- `plot2dSeparator.m`
- `findLinearThreshold.m`.

Please do NOT submit these files separately. A summary of what to submit is listed in the end of this problem.

b.1 *[Writing LP - 5 points]* **Rewrite the linear program** [(2) **to** (4)] **into a "standard form" linear program (the form used in Definition** 1**).**

Now, implement a Matlab function called `findLinearDiscriminant` **(by editing** `findLinearDiscriminant.m`**) to find a linear discriminant function that separates a given data set using the linear programming formulation described above** (you can refer to the appendix for a primer on solving linear programs using Matlab). Your Matlab function must take `data` as input, and produce the linear separator, i.e., the weights `w` ($\vec{w}$), the threshold `theta` (θ), and the value of the objective at the minimum, i.e., `delta` (δ). We will read our data from files. A data file consists of lines that contain a sequence of binary values followed by either 1, indicating a positive label, or by -1, indicating a negative label. A sample line might be "`0 1 -1`", which corresponds to $\vec{x} = [0, 1]^T, y = -1$. In order to read files in this format, we provided a Matlab function called `readFeatures` that takes a file name and the dimension of the feature vector $\vec{x}$ (denoted n) and returns an $m \times (n + 1)$ matrix called `data`, where m is the number of data points.

b.2 *[Learning Conjunctions as an LP - 10 points]* There are two parts in this question. **First, generate a data set that represents a monotone conjunction**[5] **over two variables manually. Write this data set into** `hw1sample2d.txt`. We provided a Matlab function called `plot2dData` to plot the labeled data points given by `data`. **Now, implement a Matlab**

[5] A conjunction with no negated variables. This file should be 4 lines with 3 numbers per line.

function called `plot2dSeparator` (by editing `plot2DSeparator.m`) that takes `w` and `theta` as input, and plots the corresponding separator line. For your dataset, find the linear separator and plot it together with your data, and include the figure in your solution.

In addition, we provided a data set in `hw1conjunction.txt` that is consistent with a conjunctive concept with $n = 10$. **Use your linear program to learn the target concept in `hw1conjunction.txt`. State the linear discriminant function returned and *succinctly* explain your results. For example, what conjunctive concept is represented by the hypothesis returned, and how can this be known from the resulting hyperplane from your function (or can this be known). What can be said about δ and θ? In addition to your explanations, you should also give the actual weights, δ and θ reported by your program.**

You can use the script `learnConjunctions` to run these two experiments and obtain the materials you need to submit in your answer. Note that this script outputs the model of the second experiment in a file named `p3b2-model.txt`. **You also need to submit `p3b2-model.txt`.**

b.3 *[Learning Badges - 10 points]* You will solve the badges problem using your linear program. The input files `badges.train`, and `badges.test` contain the names of the participants and the badge labels. We will consider features of the following form. Let Σ be the set of characters of interest. Let D be the set of character positions of interest. For each $(d, \sigma) \in D \times \Sigma$, we have a binary variable $x_{(d,\sigma)}$ that is set to 1 if the d-th character of the name is the character σ (ignoring case), or to 0 otherwise. The feature vector $\vec{x}$ then has dimensions $|\Sigma||D|$. Given Σ (`alphabet`) and D (`positions`), the Matlab function `writeBadgeFeatures` computes the feature vector and writes it to a file using the data file format.

In this part, you will use the Matlab script `learnBadges`. It defines an alphabet consisting of 30 characters, and a position list containing the first ten positions. This script uses your `findLinearDiscriminant` to obtain a linear separator. In order to evalute your result, we provided a code to compute the accuracy of this separator both in the training set and in the test set. The function `computeAccuracy` makes calls to the `computeLabel` function, which should predict the label for the given feature vector using the linear separator represented by w, and θ. **In order for accuracy computations to work, you need to implement the `computeLabel` function.**

In addition to accuracy, `learnBadges` creates a figure to illustrate the weight vector returned by your linear program. The $x-$axis shows the characters, the $y-$axis shows the positions, and the pixel colors at location (σ, d) denote the weight corresponding to the feature $x_{(d,\sigma)}$.

Now, run this script and explain δ, the accuracies, and the generated figure in your solution (Be sure to include the figure in your

solution).
Next, experiment with different feature vectors by changing `alphabet` and/or `positions` in `learnBadges` script. **Find one `alphabet` and `positions` such that the linear separator computed from the resulting feature vector has a perfect accuracy ($= 1$) in the training dataset, but has an imperfect accuracy (< 1) in the test dataset. Give the figure showing the weight vector for this linear separator. How does this figure compare to the figure you obtained before?**

b.4 *[Learning Multiple Hyperplanes - 10 points]* For this part, you will use the Matlab script `learnMultipleHyperplanes`. The script generates a data matrix that contains 20 two-dimensional real-valued examples with positive and negative labels. First, the script finds a linear separator using your linear program.
Now, consider that you are given the weights, but not the threshold, and the goal is to solve the linear program to find the threshold that minimizes δ. **Your job is to implement the function `findLinearThreshold` that takes the data and w, and returns θ, and δ.**
The script has three different weight vectors, and for each of them it calls your function and then plots the resulting linear function using `plot2dSeparator`. It also generates a text file `p3b4-model.txt` which stores the four different models. **Run this script, and explain the results (Be sure to include the generated figures in your solution and upload `p3b4-model.txt`). Which hyperplanes separated the data exactly? What δ values have you obtained, and what does they tell us? Is one hyperplane better than another for classification? From these examples, can we infer whether or not the solution to the linear program ([(2) to (4)]) is unique?**

What to Submit

- A pdf file which contains
  - your answers to each question,
  - the code snippets of `findLinearDiscriminant.m`, `computeLabel.m`, `plot2dSeparator.m`, and `findLinearThreshold.m`,
  - figures generated by `learnConjunctions.m`, `learnBadges.m`, and `learnMultipleHyperplanes.m`.
- `p3b2-model.txt` and `p3b4-model.txt`. You will generate these two files while executing `learnConjunctions.m` and `learnMultipleHyperplanes.m`.
- Please upload the above three files on Compass. (`http://compass2g.illinois.edu`)

Appendix: Linear Programming

In this appendix, we will walk through a simple linear programming example. If you want to read more on the topic, a good reference is *Linear Programming: Foundations and Extensions* by Vanderbei. Some classic texts include *Linear Programming* by Chvatal; and *Combinatorial Optimization: Algorithms and Complexity* by Papadimitrou and Steiglitz (in particular, the beginning of chapter 2 may be helpful). A widely available (albeit incomplete) reference is *Introduction to Algorithms* by Cormen, Leiserson, Rivest, and Stein.

Example: Consider the following problem.[6] You are given a choice of three foods, namely eggs at a cost of \$0.10 an egg, pasta at a cost of \$0.05 a bowl, and yogurt at a cost of \$0.25 a cup. An egg (t_1) provides 3 portions of protein, 1 portion of carbohydrates, and 2 portions of fat. A bowl of pasta (t_2) provides 1 portion of protein, 3 portions of carbohydrates, and no portions of fat. A cup of yogurt (t_3) provides 2 portions of protein, 2 portions of carbohydrates, and 1 portion of fat. You are required to consume at least 7 portions of protein and 9 portions of carbohydrates per day and are not allowed to consume more than 4 portions of fat. In addition, you obviously may not consume a negative amount of any food. The objective now is to find the cheapest combination of foods that still meet your daily nutritional requirements.

This can be written as the following linear program:

$$z = 0.1t_1 + 0.05t_2 + 0.25t_3 \to \min \tag{5}$$

$$3t_1 + t_2 + 2t_3 \geq 7 \tag{6}$$

$$t_1 + 3t_2 + 2t_3 \geq 9 \tag{7}$$

$$-2t_1 - t_3 \geq -4 \tag{8}$$

$$t_i \geq 0 \qquad \forall i \tag{9}$$

Note that inequality (8) of the LP is equivalent to $2t_1 + t_3 < 4$. This corresponds to:

$$\mathbf{A} = \begin{pmatrix} 3 & 1 & 2 \\ 1 & 3 & 2 \\ -2 & 0 & -1 \end{pmatrix} \quad \vec{b} = \begin{pmatrix} 7 \\ 9 \\ -4 \end{pmatrix} \quad \vec{c} = \begin{pmatrix} 0.1 \\ 0.05 \\ 0.25 \end{pmatrix} \quad \vec{t} = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix}$$

To solve this program using Matlab:

```
c = [0.1; 0.05; 0.25];
A = [3 1 2; 1 3 2; -2 0 -1];
b = [7; 9; -4];
lowerBound = zeros(3, 1); % this constrains t >= 0
[t, z] = linprog(c, -A, -b, [], [], lowerBound)
```

The results of this linear program show that to meet your nutritional requirements at a minimum cost, you should eat 1.5 eggs, 2.5 bowls of pasta, and no cups of yogurt, and the cost for such a diet is \$0.275.

[6] The problem closely resembles an instantiation of the *diet problem* by G. J. Stigler, *The Cost of Subsistence*, 1945.