10-715 Advanced Introduction to Machine Learning: Homework 1
Decision Trees and Perceptron Algorithm

Released: Wednesday, August 29, 2018
Due: 11:59 p.m. Wednesday, September 5, 2018

Instructions

- **Late homework policy:** Homework is worth full credit if submitted before the due date, half credit during the next 48 hours, and zero credit after that.
- **Collaboration policy:** Collaboration on solving the homework is allowed. Discussions are encouraged but you should think about the problems on your own. When you do collaborate, you should list your collaborators! Also cite your resources, in case you got some inspiration from other resources (books, websites, papers). If you do collaborate with someone or use a book or website, you are expected to write up your solution independently. That is, close the book and all of your notes before starting to write up your solution. We will be assuming that, as participants in a graduate course, you will be taking the responsibility to make sure you personally understand the solution to any work arising from such collaboration.
- **Online submission:** You must submit your solutions online on Autolab (link: `https://autolab.andrew.cmu.edu/courses/10715-f18/assessments`). Please use LaTeX to typeset your solutions, and submit a single pdf called **hw1.pdf**.

Problem 1: Splitting Heuristic for Decision Trees [30 Points]

Recall that the ID3 algorithm iteratively grows a decision tree from the root downwards. On each iteration, the algorithm replaces one leaf node with an internal node that splits the data based on one decision attribute (or feature). In particular, the ID3 algorithm chooses the split that reduces the entropy the most, but there are other choices. For example, since our goal in the end is to have the lowest error, why not instead choose the split that reduces error the most? In this problem we will explore one reason why reducing entropy is a better criterion.

Consider the following simple setting. Let us suppose each example is described by n boolean features: $X = \langle X_1, \dots X_n \rangle$, where $X_i \in \{0, 1\}$, and where $n \geq 4$. Furthermore, the target function to be learned is $f : X \to Y$, where $Y = X_1 \vee X_2 \vee X_3$. That is, $Y = 1$ if $X_1 = 1$ or $X_2 = 1$ or $X_3 = 1$, and $Y = 0$ otherwise. Suppose that your training data contains all of the 2^n possible examples, each labeled by f. For example, when $n = 4$, the data set would be

X_1	X_2	X_3	X_4	Y
0	0	0	0	0
1	0	0	0	1
0	1	0	0	1
1	1	0	0	1
0	0	1	0	1
1	0	1	0	1
0	1	1	0	1
1	1	1	0	1

X_1	X_2	X_3	X_4	Y
0	0	0	1	0
1	0	0	1	1
0	1	0	1	1
1	1	0	1	1
0	0	1	1	1
1	0	1	1	1
0	1	1	1	1
1	1	1	1	1

(a) [**10 Points**] How many mistakes does the best 1-leaf decision tree make, over the 2^n training examples? (The 1-leaf decision tree does not split the data even once)

(b) [**10 Points**] Is there a split that reduces the number of mistakes by at least one? (I.e., is there a decision tree with 1 internal node with fewer mistakes than your answer to part (a)?) Why or why not?

(c) [**5 Points**] What is the entropy of the output label Y for the 1-leaf decision tree (no splits at all)?

(d) [**5 Points**] Is there a split that reduces the entropy of the output Y by a non-zero amount? If so, what is it, and what is the resulting conditional entropy of Y given this split?

Problem 2: Accuracy of Perceptron Algorithm [30 Points]

(a) [**15 Points**] Lets suppose we have d weather models $\{M_i\}_{i=1}^{d}$, each of which predicts if it is going to rain the next day. Each model outputs either -1 or $+1$, where -1 corresponds to no rain and $+1$ corresponds to rain. We do not know anything about these models (some of these could be malicious). We only know that there is some subset of K of the d models whose majority vote always gives the correct prediction (where K is an odd number). However we dont know the identity of these K models.

Our goal is to learn the best possible way to combine the outputs of these d models to reliably predict rain. To be more formal, we would like to learn a function $h : \{-1, +1\}^d \to \{-1, +1\}$, which takes the predictions of the d models as inputs and reliably predicts whether it is going to rain the next day.

Suppose we use the Perceptron algorithm to learn a linear predictor h. Each day, we make a prediction based on the current h and update h the next day if our prediction was wrong. Show that you will make at most Kd wrong predictions using this approach.

(b) [**15 Points**] In this sub problem we will consider a variant of the online Perceptron algorithm we saw in the class. In each iteration of the algorithm, given example X with label $y \in \{-1, +1\}$, we always update w as:

$$w \leftarrow w + yX$$

(Note that in the original perceptron algorithm, we update w only if the predicted label differs from the true label.)

Suppose the data has margin γ and all points lie inside a ball of radius R. Is there an upper bound on the number of mistakes the above algorithm can make? If yes, provide one. If not, provide an example where it can make infinitely many mistakes.

Problem 3: Margin Perceptron Algorithm [30 Points]

Suppose we have a set of examples S and we want to find a large-margin separator for them. One approach is to directly solve for the maximum-margin separator using convex programming (the SVM algorithm). However, if we only need to approximately maximize the margin, then another approach is to use a variation on the Perceptron algorithm called the *Margin Perceptron algorithm.*

The Margin Perceptron algorithm works just like the standard Perceptron algorithm, except instead of just updating on a mistake it also updates when the current hypothesis gets the example correct but by margin less than $\gamma/2$. Assuming that data lies in a ball of radius R and is separable by margin γ, then we can show that this will make at most $O((R/\gamma)^2)$ updates. Specifically, the algorithm is as follows:

The Margin Perceptron Algorithm (γ):

1. Initialize $t = 1$ and start with the all-zeroes weight vector w_1.
2. Given an example x, predict positive iff $w_t \cdot x \geq 0$.
3. Given the true label y for x, if x was classified incorrectly *or* classified correctly but by a margin less than $\gamma/2$ (that is, $y(w_t \cdot x)/||w_t|| < \gamma/2$), then update the weight vector:
   (a) Let $w_{t+1} = w_t + yx$. That is, if x is positive then let $w_{t+1} = w_t + x$ and if x is negative let $w_{t+1} = w_t - x$.
   (b) Let $t = t + 1$.

[**20 Points**] Prove the following theorem regarding the Margin Perceptron Algorithm:

Margin Perceptron Theorem: *Let S be a sequence of labeled examples consistent with a linear separator $w^* \cdot x = 0$ of margin γ. That is, $||w^*|| = 1$ and*

$$\gamma = \min_{(x,y)\in S} y(w^* \cdot x).$$

Then the number of weight updates made by Margin Perceptron(γ) on S is at most $12(R/\gamma)^2$.

[**10 Points**] Suppose you are given a set S of labeled examples up front that are linearly separable by margin γ. How would you use the Margin Perceptron algorithm to find a linear separator for S of margin at least $\gamma/2$? (In other words, how could you convert the online algorithm described above into an algorithm you could run on a batch of examples that would produce a separator with large margin over all the examples in the batch?)

Problem 4: Runtime of Perceptron Algorithm [10 Points]

[**10 Points**] Describe a set S of $O(n)$ examples over $\{0,1\}^n$ that are linearly separable by a hyperplane through the origin, but where the Perceptron algorithm takes exponential time for learning (i.e., time $2^{\Omega(n)}$). Specifically, we are imagining we repeatedly cycle the perceptron algorithm through the examples until we have $w \cdot x > 0$ for every positive example $x \in S$ and we have $w \cdot x < 0$ for every negative $x \in S$. For simplicity, use the version of the Perceptron algorithm that does not normalize examples (i.e., when a mistake is made on a positive example x, it sets $w = w + x$, rather than $w = w + x/\|x\|$, and similarly for mistakes on negatives). This won't really matter since $\|x\| \leq \sqrt{n}$ for $x \in \{0,1\}^n$, but it is easier to think about since the w_i will always be integral. Explain why your set of examples has the desired property.