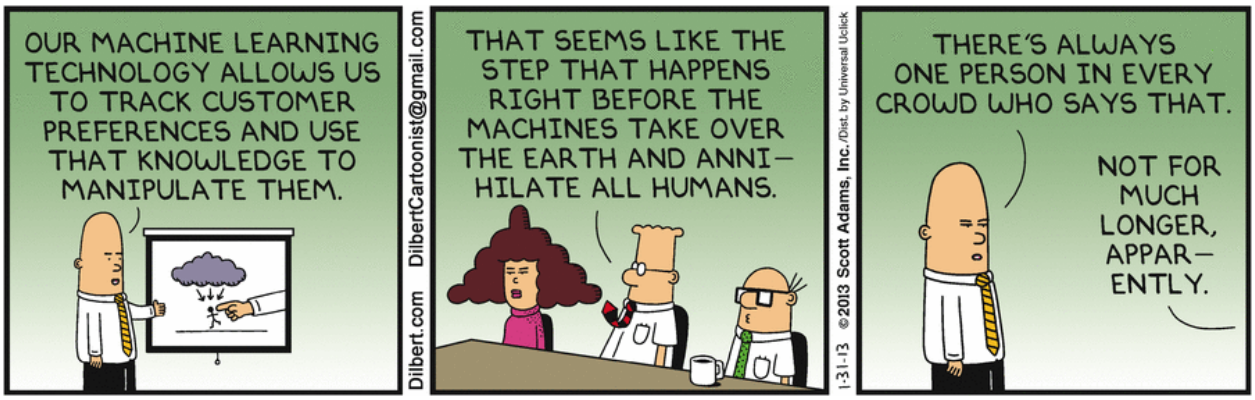
Chapter 1: The Learning Problem



1 Overview

The first chapter of the book introduces

1. basic notation, and
2. a learning model called the perceptron.

We analyze the runtime properties of the perceptron in this chapter, and we'll analyze the statistical properties of the algorithm in Chapter 2. Then in Chapters 3-4, we'll look at more complicated learning algorithms.

The author provides lecture videos that go along with the book, which you may find helpful. They are located at `https://www.youtube.com/watch?v=mbyG85GZ0PI`.

2 Section 1.1: Problem Setup

Problem 1. You are a bank and need an algorithm for determining whether to approve a credit loan application. Describe the learning framework notation for this problem. (See Section 1.1.1 in the textbook.)

NOTE: You should be able to do this for other applications as well. See Exercise 1.1 for a list of alternative applications to practice with.

Problem 2. This problem explores the perceptron hypothesis class.

1. Define the Perceptron hypothesis class. (See Section 1.1.2 in the textbook.)

2. Describe which features for the bank loan problem are likely to have high weights, and which features are likely to have low weights on a "good" perceptron hypothesis.

   NOTE: This is closely related to Exercise 1.2 in the book, which you should be able to complete on your own.

3. Show "good" and "bad" examples of hypotheses in the Perceptron hypothesis class in $d = 2$ dimensions. (See the discussion in Figure 1.3 and 3.1 in the textbook.)

Problem 3. The Perceptron Learning Algorithm (PLA) is a procedure for selecting a particular hypothesis from the Perceptron hypothesis class.

1. State the PLA update rule, as found in Eq (1.3).

2. Exercise 1.3 in the textbook is designed to help you get some intuition for why the PLA update rule is a good rule. It has the following parts:

   (a) Show that $y(t)\mathbf{w}^T(t)\mathbf{x}(t) < 0$.

(b) Show that $y(t)\mathbf{w}^T(t+1)\mathbf{x}(t) > y(t)\mathbf{w}^T(t)\mathbf{x}(t)$.

(c) As far as classifying $\mathbf{x}(t)$ is concerned, argue that the move from $\mathbf{w}(t)$ to $\mathbf{w}(t+1)$ is a move 'in the right direction'.

NOTE: See the Figure on the bottom of page 7 for a visual description of these results.

3. Problem 1.3 (at the end of the chapter) proves that the total number of iterations t required for the PLA to converge to an optimal solution is bounded by

$$t \leq \frac{R^2 \|\mathbf{w}^*\|_2^2}{\rho^2}, \tag{1}$$

where $\mathbf{w}^*$ is an optimal hyperplane,

$$\rho = \min_{1 \leq n \leq N} y_n(\mathbf{w}^{*T}\mathbf{x}_n) \qquad \text{and} \qquad R = \max_{1 \leq n \leq N} \|\mathbf{x}_n\|_2. \tag{2}$$

In this class, we won't complete that problem and prove this bound; instead, we will focus on understanding the consequences of this bound.

(a) Illustrate the quantities above.

(b) What is the overall runtime of running the PLA to convergence?

4. Describe some of the differences between our runtime analysis for the PLA and the Pagerank power method algorithm.

Problem 4. Based on the analysis above, draw two data sets that we expect PLA to have very different runtimes for, and explain which data set will be faster and which will be slower.

Problem 5. In order to speed up the runtime of PLA, Alice is proposing several different variants of the algorithm. For each variant below, identify the conditions (if any) under which it will have a faster runtime.

1. Alice proposes "shrinking" the data points by a factor of α so that R will be smaller. That is, she first creates new data points defined by
$$\mathbf{x}_i' = \frac{\mathbf{x}_i}{\alpha} \tag{3}$$
and then runs the PLA on this new data set.

2. Alice proposes "centering" the data points so that R will be smaller. That is, she first creates new data points defined by

$$\mathbf{x}_i' = \mathbf{x}_i' - \bar{\mathbf{x}} \tag{4}$$

where

$$\bar{\mathbf{x}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i, \tag{5}$$

and then runs the PLA on this new data set.

3. Alice notices that there is no unique value for $\mathbf{w}^*$ that defines the optimal separating hyperplane. In particular, for every $\mathbf{w}^*$ that defines a separating hyperplane and every positive real number α, $\alpha\mathbf{w}^*$ also defines a separating hyperplane. By setting α to be small and using $\alpha\mathbf{w}^*$ as the separating hyperplane in the runtime formulas, Alice hopes to achieve a faster runtime. Why won't this work?