Note 1

01 Introduction & Abstractions

by Alvin Wan

Have you ever tailored your exam preparation, to optimize exam performance? It seems likely that you would, and in fact, those preparations may range from "taking one practice exam" to "wearing lucky underwear". The question we're implicitly answering is: which of these actually betters exam performance? This is where machine learning comes in. In machine learning, we strive to find patterns in data, then make predictions using those patterns, just as the average student would do to better exam scores.

1 Introduction

1.1 Terminology

We will call each exam a **sample**, where each exam score is the **label**. To predict our exam score, we consider various factors that may affect our score, such as homework or practice exam scores. We call these **features** of each sample, which are usually quantitative factors that are believed to influence our label. We then train a **model** that attempts to predict labels, using only the features of each sample.

By convention, each sample point x_i has d features. We write $x_i \in \mathbb{R}^d$ to mean that each x_i is a $d \times 1$ *column* vector with real-valued entries. Confusingly, the matrix X containing all of our samples is written differently, as an $n \times d$ matrix, treating each sample as a *row* vector. Here, n is the number of samples. We denote the model using $w \in \mathbb{R}^d$, meaning w is a $d \times 1$ column vector.

One of machine learning's most dreaded evils is **overfitting**, where a model is too closely tailored to the training data. In the limit, the most overfit model will memorize the data. This might mean that if one does well on exam A, one repeats every detail for exam B - down to the duration of an inter-exam restroom trip and whether or not one used the urinal. A related but less common evil is **underfitting**, where the model

is not sufficiently expressive to capture important information in the data. This could mean that one looks only at homework scores to predict exam scores, ignoring the effects of reading notes, completing practice exams, and more. Our goal is to build a model that generalizes to new examples while making the appropriate distinctions.

Given these two evils, there are a variety of approaches to fighting both. One is tuning **hyperparameters**, which may govern aspects of training such as "speed" or "momentum". We will formalize these notions later on. Another is ensuring that we train, validate, and test properly. This means separating our data accordingly and not peeking prematurely at any set of data. In general, our data is split into three portions:

1. **Training set**: This is the dataset you train your model on.
2. **Validation set**: This is the dataset you evaluate your model on, to determine hyperparameters.
3. **Test set**: This is the dataset you evaluate on for accuracy, once at the very end. Running on the test set prematurely could mean your model overfits to the test set as well, so run only once.

1.2 Overview

In this course, we will explore both **supervised learning**, where our algorithm has access to labeled data, and **unsupervised learning**, where our algorithm does not have access to labels. In supervised learning, we explore the following two classes of problems:

1. **classification**: Determine which of k classes $\{C_1, C_2, \dots C_k\}$ to which each sample point x_i belongs. Ex. "Which respiratory disease is this?"
2. **regression**: Determine a real-valued output, which are often probabilities. Ex. "What is the probability this patient has neuroblastoma? (eye cancer)"

In unsupervised learning, we explore the following classes of problems:

1. **clustering**: Cluster sample points x_i into k clusters. We do not have a label for the resulting clusters. "Which DNA sequences are most similar?"

2. **dimensionality reduction**: Reduce the number of "unique" (linearly independent) features we consider. "What are common features of faces?"

In either case, the quality of our data determines the quality of our model and more importantly, we need vast amounts of data. This is often on the order of tens of thousands of sample points. Some algorithms, such as k-nearest neighbors, may even see cutting edge accuracy with sufficiently large amounts of data, on the order of hundreds of thousands.

1.3 Abstractions

Consider the following abstractions for machine learning. It is important to compartmentalize, so that you understand what math goes where. The following is quoted verbatim from Professor Jonathan Shewchuk's notes:

1. **Data**: Is the data labeled or not? If so, are the labels categorical (classification) or quantitative (regression)? If not, are we interested in similarity (clustering) or positioning (dimensionality reduction)?
2. **Model**: Determine the following.
   (a) Decision functions: linear, polynomial, logistic, neural net ...
   (b) Nearest neighbors, decision trees
   (c) Features
   (d) Low vs. High Capacity (affects overfitting, underfitting, inference)
3. **Optimization Problem**: Variables, objective functions, constraints (least squares, PCA)
4. **Optimization Algorithm**: Gradient descent, simplex, SVD