Learning From Data Notes

Yuanbo Li

Contents

Chapter 1

The Learning Problem Notes

1.1 The Learning Problem

1.1.1 Problem Setup

Components of Learning

Notation Convensions:

- x: input
- f: $X \to Y$: unknown target function, where X is the input space, and Y is the output space.
- D: data set of input-output examples $(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_N}, y_N)$, where $y_n = f(\mathbf{x_n})$ for n = 1, ..., N.
- H: hypothesis set H. There is a learning algorithm that uses the data set D to pick a formula $g : X \to Y$ that approximates f. The algorithm chooses g from a set of candidate formulas under consideration, which we call the hypothesis set H.

A Simple Learning Model

- *Learning model*: The hypothesis set and learning algorithm we choose are refered to informmally as the learning model.

Learning versus Design

- The main differences between the learning approach and the design approach is the role that data plays. In the design approach, the problem is well-specified and one can analytically derive f without the need to see any data. In the learning approach, the problem is much less specified, and one needs data to pin down what f is.

1.1.2 Types of Learning

Supervised Learning

- Description: the training data contains explicitly examples of what the correct output should be for given inputs. Example: hand-written digit recognition problem.

- two types of supervised learning:
  - ***Active learning***: the data set is acquired through queries that we make. Thus, we get to choose a point $\mathbf{x}$ in the input space, and the supervisor reports to us the target value for $\mathbf{x}$. As you can see, this opens the possibility for strategic choice of the point $\mathbf{x}$ to maximize its information value, similar to asking a strategic question in a game of 20 questions.
  - ***Online learning***: The data set is given to the algorithm one example at a time. This happens when we have streaming data that the algorithm has to process 'on the run'. We should note that online learning can be used in different paradigms of learning, not just in supervised learning.

Reinforcement Learning

- The training data does not contain the correct output, but instead some possible output and a measure of how good a output is. The example does not specify how good other outputs would have been for this particular input.
- Reinforcement learning is very useful in some circumstances, for instance, learning how to play chess. It is hard to find the "correct" solution. However, in reinforcement learning setting, all you need is to take some action and report how well things goes, then you have a training example.

Unsupervised Learning

- The training data does not have any output information at all.
- The unsupervised learning can be viewed as the task of spontaneously finding patterns and structure in input data.

Other Views of Learning

- *Machine Learning*: the main fields that dedicated to the subject is called machine learning
- *Statistics*: shares the basic premise of learning from data. Statistics make restrict assumptions and analyze it in great detail.
- *Data Mining*: Data mining is a practical field that focuses on finding patterns, correlation, or anomalies in large relational databases. More emphasis on data analysis than on prediction. Because databases are usually huge, computational issues are often critical in data mining.

1.1.3 Is Learning Feasible?

Outside the Data Set

- Does the data set D tell us anything outside of D that we does not know before? If the answer is yes, we have learned *something.* If not, we can conclude that learning is not feasible.

Probability to the Rescue

- Example:
  - We pick a random sample of N independent marbles (with replacement) from the bin, and observe the fraction ν of red marbles within the sample. We denote the probability of picking a red marble as μ

- To quantify the relationship between ν and μ, we use a simple bound called the *Hoeffding Inequality.* It states that for any sample size N,

$$\mathbb{P}[|\nu - \mu| > \epsilon] \leq 2e^{-2\epsilon^2 N} \text{ for any } \epsilon > 0 \tag{1.1}$$

  Here, $\mathbb{P}$ denotes the probability of an event. In this case, it is exponentially unlikely that ν will deviate from μ by more than our "tolerance" ϵ.
- To connect the bin model to the pearning problem, the unknown μ here can be simply viewed as the function $f\colon X \to Y$.
- The error rate within the sample, which corresponds to ν in the bin model, will be called the *in-sample error*,

$$\begin{aligned} E_{in}(h) &= (\text{fraction of } D \text{ where } f \text{ and } h \text{ disagree}) \\ &= \frac{1}{N}\sum_{n=1}^{N}[[h(x_n) \neq f(x_n)]] \end{aligned} \tag{1.2}$$

  where [[statement = 1]] if the statement is true and = 0 if false.
- In the similar way, we can define the *out-of-sample error*

$$E_{out}(h) = \mathbb{P}[h(x) \neq f(x)], \tag{1.3}$$

  which corresponds to μ in the bin model. The probability is based on the distribution P over X which is used to sample the data point $\mathbf{x}$.
- Substituting the new notation E_{in} for ν and E_{out} for μ the Hoeffding Inequality can be rewritten as

$$\mathbb{P}[|E_{in}(h) - E_{out}(h)| > \epsilon] \leq 2e^{-2\epsilon^2 N} \text{ for any } \epsilon > 0 \ , \tag{1.4}$$

  where N is the number of training examples.
- we simply assume for now that H is a finite set. Let g denotes for the final hypothesis selected by learning algorithm.

$$\begin{aligned} \text{``}|E_{in}(g) - E_{out}(g)| > \epsilon\text{''} \implies \text{``}\quad & |E_{in}(h_1) - E_{out}(h_2)| > \epsilon \\ & \textbf{or } |E_{in}(h_2) - E_{out}(h_2)| > \epsilon \\ & \cdots \\ & \textbf{or } |E_{in}(h_M) - E_{out}(h_M)| > \epsilon\text{''}. \end{aligned} \tag{1.5}$$

  Then, we get

$$\begin{aligned} \mathbb{P}[|E_{in}(g) - E_{out}(g)| > \epsilon] \leq\ & \mathbb{P}[|E_{in}(h_1) - E_{out}(h_2)| > \epsilon \\ & \textbf{or } |E_{in}(h_2) - E_{out}(h_2)| > \epsilon \\ & \cdots \\ & \textbf{or } |E_{in}(h_M) - E_{out}(h_M)| > \epsilon]. \\ \leq & \sum_{m=1}^{M}\mathbb{P}[|E_{in}(h_m) - E_{out}(h_m)| > \epsilon]. \end{aligned} \tag{1.6}$$

  Applying the Hoeffding Inequality 1.1 to the M terms one at a time, we get

$$\mathbb{P}[|E_{in}(g) - E_{out}(g)| > \epsilon] \leq 2Me^{-2\epsilon^2 N} \tag{1.7}$$

- This is only meaningful with finite M. We will imporve on infinite case in Chapter 2.

Feasibility of Learning

- If we insist on a deterministic answer, which means that D tells us something certain about f outside of D, then the answer is no.
- If we accept a probabilistic answer, which means that D tells us something likely about f outside of D, then the answer if yes.
- Notice: There are two feasibility problem: first, can we make sure that $E_{out}(g)$ is close enough to $E_{in}(g)$. Then, can we make $E_{in}(g)$ small enough.
- ***The complexity of $\boldsymbol{H}$***. If the number of hypotheses M goes up, we run more risk that $E_{in}(g)$ will be a poor estimator of $E_{out}(g)$. M can be thought of as a measure of the 'complexity' of the hypothesis set H that we use.
- ***The complexity of $\boldsymbol{f}$***. Intuitively, a complex target function f should be harder to learn than a simple f.

1.1.4 Error and Noise

Error Measures

- $$Error = E(h, f) \tag{1.8}$$

  where $E(h, f)$ is based on the entirety of h and f, it is almost universally defined based on the errorss on individual input points $\mathbf{x}$.
- Different error measure may affects the outcome of the learning process. Different error measures may lead to different choices of the final hypothesis, even if the target and the data are the same, since the value of a particular error measure may be samll while the value of another error measure in the same situation is large.

Noisy Targets

- The function itself may be noisy, meaning that it is not deterministic. In this case, we may have a target *distribution* $P(y|\mathbf{x})$ instead of a target function $y = f(x)$.
- Another view of target function is that a noisy target is a deterministic target plus added noise.
- In noisy target setting, E_{in} itself will likely be worse so it is hard to fit the noise.

Chapter 2

Training Versus Testing

2.1 Training versus Testing

2.1.1 Theory of Generalization

- The out-of-sample error E_{out} measures how well our training on D has generalized to data that we have not seen before. It is based on the performance over the entire input space X. If we want to estimate it, we should use unseen "fresh" data points.
- The in sample error E_{in}, by contrast, is based on data points that have been used for training.
- ***Generalization error***: the discrepancy between E_{in} and E_{out}. Notice: sometime, 'generalization error' is also used for another name of E_{out}, but not for this text. We have already provides a way to characterize the generalization error with a probabilistic bound in chapter 1,
$$\mathbb{P}[|E_{in}(g) - E_{out}(g)| > \epsilon] \leq 2Me^{-2e^2N}, \tag{2.1}$$
for any $\epsilon > 0$
- The generalization error mentioned above can be rephrased as follows. Pick a tolerance level δ, for example, $\delta = 0.5$, we assert with probability at least $1-\delta$. Then, we can get $\mathbb{P}[|E_{in}(g)-E_{out}(g)| > \epsilon] \leq 2Me^{-2e^2N} \leq \delta$.
Finally we can reach
$$E_{out}(g) \leq E_{in}(g) + \sqrt{\frac{1}{2N} ln \frac{2M}{\delta}}. \tag{2.2}$$
We refer to the type of inequality in the form 2.2 as a ***generalization bound*** because it bounds E_{out} in terms of E_{in}
- (accuracy upper bound) It is worth noticing that the other side of $|E_{out} - E_{in}| \leq \epsilon$ also holds, that is, $E_{out} \geq E_{in} - \epsilon$. This is important for learning. The $E_{out} \geq E_{in} - \epsilon$ direction of the bound assures us that we couldn't do significantly better if we have a relatively high E_{in}.
- (limitation) The error bound $\sqrt{\frac{1}{2N} ln \frac{2M}{\delta}}$ in 2.2 depends on M, the size of the hypothesis set $\mathcal{H}$. If $\mathcal{H}$ is a infinite set, the bound will goes to infinity, thus the bound becomes meaningless. Unfortunately, almost all interesting hypothesis set $\mathcal{H}$ are infinite.
- (The cause of the limitation) We over-estimate the the probability using union bound. Let's denote $\mathcal{B}_m$ be ($\mathcal{B}$ad) event that "$|E_{in}(h_m) - E_{out}(h_m)| > \epsilon$", Then our assumption
$$\mathbb{P}[\mathcal{B}_1 \textbf{ or } \mathcal{B}_2 \textbf{ or } \cdots \textbf{ or } \mathcal{B}_M] \leq \mathbb{P}[\mathcal{B}_1] + \mathbb{P}[\mathcal{B}_2] + \cdots + \mathbb{P}[\mathcal{B}_M]. \tag{2.3}$$
The events $\mathbb{B}_1, \cdots, \mathbb{B}_M$ are more likely strongly overlapping, so our estimate is bad enough in this case.

2.1.2 Effective Number of Hypothesis

To overcome the limitation mentioned above, we introduce *growth function*, the quantity that will formalize the effective number of hypothesis. The growth function is what will replace M in the generalization bound 2.2. It is a combinatorics quantity that captures how different the hypothesis in $\mathcal{H}$ are, and hence how much overlap the different events have.

We will focus on binary target function for the purpose of this analysis.

First we have to introduce hichotomies.

Definition 2.1.1
Let $\mathbf{x}_1, \cdots, \mathbf{x}_N \in \mathcal{X}$. The dichotomies generated by $\mathcal{H}$ on these points are defined by

$$\mathcal{H}(\mathbf{x}_1, \cdots, \mathbf{x}_N) = \{(h(\mathbf{x}_1), \cdots, h(\mathbf{x}_N)) | h \in \mathcal{H}\}. \tag{2.4}$$

Based on hichotomies, we have our definition of growth function.

Definition 2.1.2
The growth function is defined for a hypothesis set $\mathcal{H}$ by

$$m_{\mathcal{H}}(N) = \max_{\mathbf{x}_1, \cdots, \mathbf{x}_N \in \mathcal{X}} |\mathcal{H}(\mathbf{x}_1, \cdots, \mathbf{x}_N)|, \tag{2.5}$$

where $|\cdot|$ denote the cardinality of a set.

By this way, we bound the size of the growth function by choosing N point from $\mathcal{X}$. which is

$$m_{\mathcal{H}}(N) \leq 2^N. \tag{2.6}$$

If $\mathcal{H}$ is capable of generating all possible dichotomies on $\mathbf{x}_1, \cdots, \mathbf{x}_N$, meaning that $\mathcal{H}(\mathbf{x}_1, \cdots, \mathbf{x}_N) = \{-1, +1\}^N$ and we say that $\mathcal{H}$ can ***shatter*** $\mathbf{x}_1, \cdots, \mathbf{x}_N$