Theoretical Foundations for Machine Learning

An Introduction to PAC Learning

Ameya Daigavane

Introduction

Introduction

What will we be covering?

- The PAC Learning Model
- Bias-Complexity Tradeoff
- The No-Free-Lunch Theorem
- VC-Dimension

Textbook: *Understanding Machine Learning* by Shai-Shalev Shwartz and Shai Ben-David.

A lot of the material here is from the textbook, chapters 2 to 6. We will also be solving problems from the end of each chapter of the textbook.

Introduction

Before we dive in, it is important to ask the following questions:

- What does it mean to learn?
- What can we learn?
- How can we learn?

Introduction

Before we dive in, it is important to ask the following questions:

- What does it mean to learn?
- What can we learn?
- How can we learn?

We will see answers to these, one-by-one.

The Probably Approximately Correct Learning Model

The Probably Approximately Correct Model of Learning was formalized by Leslie Valiant at Harvard University in 1984.

The PAC Model was groundbreaking - providing a formal mathematical framework for machine learning.

Leslie Valiant won the ACM Turing Award in 2010 for his contributions to theoretical computer science.

The Probably Approximately Correct Learning Model

The Probably Approximately Correct Model of Learning was formalized by Leslie Valiant at Harvard University in 1984.

The PAC Model was groundbreaking - providing a formal mathematical framework for machine learning.

Leslie Valiant won the ACM Turing Award in 2010 for his contributions to theoretical computer science.

Before we go into the details of this model, we need to revise some concepts.

Machine Learning Terminology

Definitions: Learner Input

To explain what the learner does, we first describe what is accessible to the learner:

- **The Domain/Instance Set**
  The domain set X is the set of all objects we can label. These objects are usually represented by a vector of features.
- **The Label Set**
  The label set Y is the set of labels we assign to objects from X.
- **Training Data**
  The training set S is a sequence of m pairs from $X \times Y$. Note that repetitions are allowed!

$$S = \{(x_1, y_1), \cdots (x_m, y_m)\}$$

Definitions: Learner Output

Using the inputs given, the learner produces a prediction rule, a function h mapping X to Y.

This function is also called a *classifier*, a *predictor*, or a *hypothesis*.

Definition: Measures of Success

The measure of success of a classifier is how accurate its outputted prediction rule h is.

Definition: Measures of Success

The measure of success of a classifier is how accurate its outputted prediction rule h is.

The true error of a classifier is the probability of predicting the wrong label, according to some distribution D over X. Formally, the error of a prediction rule, with respect to some distribution D, and true labelling function f, is

$$L_{(D,f)}(h) = P(h(x) \neq f(x))$$

Definition: Measures of Success

The measure of success of a classifier is how accurate its outputted prediction rule h is.

The true error of a classifier is the probability of predicting the wrong label, according to some distribution D over X. Formally, the error of a prediction rule, with respect to some distribution D, and true labelling function f, is

$$L_{(D,f)}(h) = P(h(x) \neq f(x))$$

Clearly, the true error then depends on both the distribution D, as well as the true labelling function f. Remember, the learner has no access to either of these.

Empirical Risk Minimization

The learner cannot compute the true error. However, it can compute the training error over the training set S as follows:

$$L_S(h) = \frac{|\{i \in \{1, 2, \cdots, m\} : h(x_i) \neq f(x_i)\}|}{m}$$

Empirical Risk Minimization

The learner cannot compute the true error. However, it can compute the training error over the training set S as follows:

$$L_S(h) = \frac{|\{i \in \{1, 2, \cdots, m\} : h(x_i) \neq f(x_i)\}|}{m}$$

The learning paradigm that seeks to minimize the training error is called **Empirical Risk Minimization (ERM)**.

Overfitting

The ERM rule can lead to overfitting! A learner using the ERM rule might yield excellent performance on the training set, but might still have large true error. This indicates that the learner has not generalized well.

Inductive Bias

How can we fix this? One solution is to use a restricted hypothesis space.

Inductive Bias

How can we fix this? One solution is to use a restricted hypothesis space.

Before receiving any input, the learner must fix a set of hypothesis, called the hypothesis class $\mathcal{H}$. Every function in $\mathcal{H}$ is a function from X to Y.

The $ERM_{\mathcal{H}}$ learner seeks a hypothesis h from $\mathcal{H}$ with the lowest training error. Note that there may be many such hypotheses.

$$ERM_{\mathcal{H}}(S) \in \operatorname*{argmin}_{h \in \mathcal{H}} L_S(h)$$

Inductive Bias

How can we fix this? One solution is to use a restricted hypothesis space.

Before receiving any input, the learner must fix a set of hypothesis, called the hypothesis class $\mathcal{H}$. Every function in $\mathcal{H}$ is a function from X to Y.

The $ERM_{\mathcal{H}}$ learner seeks a hypothesis h from $\mathcal{H}$ with the lowest training error. Note that there may be many such hypotheses.

$$ERM_{\mathcal{H}}(S) \in \operatorname*{argmin}_{h \in \mathcal{H}} L_S(h)$$

This is an example of an inductive bias - we are biasing the learner towards a particular set of hypothesis. Later, we will see that inductive bias is actually necessary, in the context of PAC learning.

Finite Hypothesis Classes

The question now is, what choices of $\mathcal{H}$ will ensure that $ERM_{\mathcal{H}}$ will not overfit?

Suppose we consider finite hypothesis classes. We now analyse the performance of the $ERM_{\mathcal{H}}$ learner under some additional assumptions.

Realizability: There exists h^* in $\mathcal{H}$ such that $L_{(D,f)}(h^*) = 0$.

Independent and Identically Distributed (iid) Samples: The samples in the training set are independently and identically distributed according to D. We write, $S \sim D^m$.

Learning Parameters

There is always the possibility of receiving a non-representative training set S. Thus, we cannot always guarantee that our learner will output a satisfactory hypothesis.

Instead, we quantify the performance of the learner by two parameters:

- **Confidence Parameter** $1 - \delta$
  The probability of getting a non-representative training set is given by δ. Then, $1 - \delta$ refers to the confidence parameter.
- **Accuracy Parameter** ϵ
  The learner might not be completely accurate when predicting labels. We want to bound the probability of failure, which is the event $L_{D,f}(h) > \epsilon$.

Finite Hypothesis Classes: Proof

Define $\mathcal{H}_B$ as the set of bad hypotheses from $\mathcal{H}$.

$$\mathcal{H}_B = \{h \in \mathcal{H} : L_{(D,f)}(h) > \epsilon\}$$

Finite Hypothesis Classes: Proof

Define $\mathcal{H}_B$ as the set of bad hypotheses from $\mathcal{H}$.

$$\mathcal{H}_B = \{h \in \mathcal{H} : L_{(D,f)}(h) > \epsilon\}$$

Define M as the set of potentially misleading training sets.

$$M = \{S_x : \text{There exists } h \in \mathcal{H}_B, L_S(h) = 0\} = \bigcup_{h \in \mathcal{H}_B} \{S_x : L_S(h) = 0\}$$

Finite Hypothesis Classes: Proof

Define $\mathcal{H}_B$ as the set of bad hypotheses from $\mathcal{H}$.

$$\mathcal{H}_B = \{h \in \mathcal{H} : L_{(D,f)}(h) > \epsilon\}$$

Define M as the set of potentially misleading training sets.

$$M = \{S_x : \text{There exists } h \in \mathcal{H}_B, L_S(h) = 0\} = \bigcup_{h \in \mathcal{H}_B} \{S_x : L_S(h) = 0\}$$

Further, let h_S be the learner's output hypothesis, when given the training set S. Now, the realizability assumption gives us:

$$\{S_x : L_{D,f}(h_S) > \epsilon\} \subseteq M$$

Finite Hypothesis Classes: Proof

Thus,

$$\begin{aligned}
D^m(\{S_x : L_{(D,f)}(h_S) > \epsilon\}) &\leq D^m(M) \\
&\leq \sum_{h \in \mathcal{H}_B} D^m(\{S_x : L_S(h) = 0\}) \\
&= \sum_{h \in \mathcal{H}_B} (D(\{x_i : h(x_i) = f(x_i)\}))^m \\
&\leq \sum_{h \in \mathcal{H}_B} (1-\epsilon)^m \\
&\leq \sum_{h \in \mathcal{H}_B} e^{-\epsilon m} \\
&\leq |\mathcal{H}_B| e^{-\epsilon m} \\
&\leq |\mathcal{H}| e^{-\epsilon m}.
\end{aligned}$$

Finite Hypothesis Classes: Proof

Thus,

$$D^m\{S_x : L_{(D,f)}(h_S) > \epsilon\} \leq |\mathcal{H}|e^{-\epsilon m}.$$

If we take m such that,

$$m \geq \frac{\log\left(|\mathcal{H}|/\delta\right)}{\epsilon}$$

Then, no matter which labelling function f and distribution D (assuming realizability), with probability of at least $1-\delta$ over the choice of m training samples, for every ERM hypothesis h_S, we have,

$$L_{(D,f)}(h_S) \leq \epsilon$$

Finite Hypothesis Classes: Proof

Thus,

$$D^m\{S_x : L_{(D,f)}(h_S) > \epsilon\} \leq |\mathcal{H}|e^{-\epsilon m}.$$

If we take m such that,

$$m \geq \frac{\log(|\mathcal{H}|/\delta)}{\epsilon}$$

Then, no matter which labelling function f and distribution D (assuming realizability), with probability of at least $1-\delta$ over the choice of m training samples, for every ERM hypothesis h_S, we have,

$$L_{(D,f)}(h_S) \leq \epsilon$$

Now, we can state what it means for a hypothesis class to be PAC learnable.

PAC Learnability

A hypothesis class $\mathcal{H}$ is said to be PAC learnable if there exists a function $m_{\mathcal{H}} : (0, 1)^2 \to \mathbb{N}$ and a learning algorithm with the following property:

For every $\epsilon, \delta \in (0, 1)$, for every distribution D over X, and for every labelling function $f : X \to \{0, 1\}$, if the realizability assumption holds, then, when running the learning algorithm on a training set S of $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ iid samples generated from D, and labelled by f, the algorithm returns a hypothesis h_S, such that, with probability at least $1 - \delta$,

$$L_{(D,f)}(h_S) \leq \epsilon$$

.

Sample Complexity

From the previous definition, finite hypothesis classes are PAC learnable. Specifically, the ERM learning algorithm can learn any finite hypothesis class. For the number of samples, $\frac{\log(|\mathcal{H}|/\delta)}{\epsilon}$ worked.

Sample Complexity

The sample complexity indicates how many samples we need in the training set to achieve some given accuracy ϵ with some given confidence $1 - \delta$.

Formally, given some ϵ, δ, the sample complexity $m_{\mathcal{H}}(\epsilon, \delta)$ is defined as the minimal number of samples required to meet the requirements of PAC learnability for $\mathcal{H}$.

Thus, every finite hypothesis class $\mathcal{H}$ is PAC learnable with sample complexity,

$$m_{\mathcal{H}}(\epsilon, \delta) \leq \frac{\log(|\mathcal{H}|/\delta)}{\epsilon}$$

What next?

Apart from finite hypothesis classes, what other hypothesis classes are PAC learnable? Can the realizability assumption be relaxed? It seems fairly restrictive.

We will look at these questions - as well as some surprising results - in the next lectures .

Solution to Class Problem

The problem asked in class was, on fixing *D* and *f*, show that,

$$\mathop{\mathbb{E}}_{S_x \; D^m}[L_S(h)] = L_{D,f}(h)$$

The calculation given in class was misleading - apologies. Here, I'm giving the intended solution - which works for all cases, not just when *X* is finite.

$$L_S(h) = \frac{1}{m}\sum_{i=1}^{m} \mathbb{I}_{h(x_i)\neq f(x_i)}$$

Note that, the definition of an indicator random variable and the iid assumption gives us:

$$\mathbb{E}[\mathbb{I}_{h(x_i)\neq f(x_i)}] = P(h(x_i) \neq f(x_i)) = P(h(x) \neq f(x))$$

Then, linearity of expectation implies:

$$\mathbb{E}[L_S(h)] = \frac{1}{m}\sum_{i=1}^{m} P(h(x) \neq f(x)) = P(h(x) \neq f(x)) = L_{D,f}(h)$$

as required.