Foundations for Machine Learning

L. Y. Stefanus

TU Dresden, June-July 2018

Reference

- Shai Shalev-Shwartz and Shai Ben-David. UNDERSTANDING MACHINE LEARNING: From Theory to Algorithms. Cambridge University Press, 2014.

A Formal Model of Learning

1. The learner's input: In the basic statistical learning setting, the learner has access to the following:
   - Domain set: An arbitrary set, X. This is the set of objects that we may wish to label. For example, in the papaya learning problem, the domain set will be the set of all papayas, represented by a vector of features (like the papaya's color and softness).

- Label set: For our current discussion, we will restrict the label set Y to be a two-element set, usually {0, 1}. For our papayas example, let 1 represents being tasty and 0 stands for being not-tasty.
- Training data: S = $((x_1, y_1), \ldots, (x_m, y_m))$ is a finite sequence of pairs in X x Y: that is, a sequence of labeled domain points. This is the input that the learner has access to (like a set of papayas that have been tasted, represented by their color, softness, and tastiness).

2. The learner's output: The learner is requested to output a prediction rule, $h : X \rightarrow Y$. This function is also called a predictor, a hypothesis, or a classifier. The predictor can be used to predict the label of new domain points. In our papayas example, it is a rule that our learner will employ to predict whether future papayas in the market are going to be tasty or not. We use the notation $A(S)$ to denote the hypothesis that a learning algorithm, A, returns upon receiving the training sequence S.

3. A data-generation model: How is the training data generated? First, we assume that the instances (the papayas) are generated by some probability distribution (in this case, representing the environment). Let us denote that probability distribution over X by D. It is important to note that we do not assume that the learner knows anything about this distribution. This could be any arbitrary probability distribution. As to the labels, in the current discussion we assume that there is some "correct" labeling function, $f : X \rightarrow Y$, and that $y_i = f(x_i)$ for all i. This assumption will be relaxed later. The labeling function is unknown to the learner. In fact, this is just what the learner is trying to figure out. In summary, each pair in the training data S is generated by first sampling a point x_i according to D and then labeling it by f.

4. Measures of success: The error of h is the probability to draw a random instance x, according to the distribution D, such that h(x) does not equal f(x). Given a domain subset, $A \subset X$, the probability distribution, D, assigns a number, D(A), which determines how likely it is to observe a point $x \in A$. We refer to A as an event and express it using a function $\pi : X \to \{0,1\}$, namely, $A = \{x \in X : \pi(x) = 1\}$. In that case, we also use the notation $P_{x\sim D}[\pi(x)]$ to express D(A). Now, we can define the error of a prediction rule, $h : X \to Y$, to be

$$L_{\mathcal{D},f}(h) \stackrel{\text{def}}{=} \mathop{\mathbb{P}}_{x\sim\mathcal{D}}[h(x) \neq f(x)] \stackrel{\text{def}}{=} \mathcal{D}(\{x : h(x) \neq f(x)\}).$$

$L_{D,f}(h)$ has several synonymous names such as the generalization error, the risk, or the true error of h.

- Notice that the learner is blind to the underlying distribution D over the world and to the labeling function f. The only way the learner can interact with the environment is through observing the training set.

Empirical Risk Minimization

- A learning algorithm receives as input a training set S, sampled from an unknown distribution D and labeled by some target function f, and should output a predictor $h_S : X \rightarrow Y$ (the subscript S emphasizes the fact that the output predictor depends on S). The goal of the algorithm is to find h_S that minimizes the error with respect to the unknown D and f.

Empirical Risk Minimization

- Since the learner does not know what D and f are, the true error is not directly available to the learner. A useful notion of error that can be calculated by the learner is the training error (the error the classifier incurs over the training sample):

$$L_S(h) \stackrel{\text{def}}{=} \frac{|\{i \in [m] : h(x_i) \neq y_i\}|}{m},$$

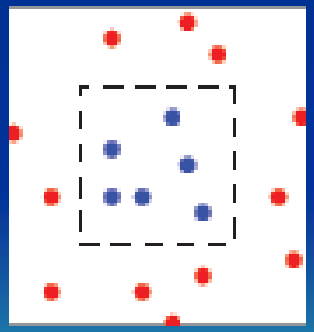
where [m] = {1, …, m}.

Empirical Risk Minimization

- The terms empirical error and empirical risk are often used interchangeably for training error.
- Since the training sample is the snapshot of the world that is available to the learner, it makes sense to search for a solution that works well on that data.
- This learning paradigm, which tries to come up with a predictor h that minimizes $L_S(h)$ is called Empirical Risk Minimization or ERM.

Overfitting

- Although the ERM rule seems very natural, without being careful, this approach may fail miserably.
- To demonstrate such a failure, let us go back to the problem of learning to predict the taste of a papaya on the basis of its softness and color.
- Consider a sample as depicted in the following:



Overfitting

- Assume that the probability distribution D is such that instances are distributed uniformly within the gray square and the labeling function, f, determines the label to be 1 if the instance is within the inner blue square, and 0 otherwise. The area of the gray square in the picture is 2 and the area of the blue square is 1.
- Consider the following predictor:

$$h_S(x) = \begin{cases} y_i & \text{if } \exists i \in [m] \text{ s.t. } x_i = x \\ 0 & \text{otherwise.} \end{cases}$$

Overfitting

- No matter what the sample is, $L_S(h_S) = 0$, and therefore this predictor may be chosen by an ERM algorithm.
- On the other hand, the true error of any predictor that predicts the label 1 is, in this case, 1/2. Thus, $L_D(h_S) = 1/2$.
- So we have found a predictor whose performance on the training set is excellent, yet its performance on the true "world" is very poor. This phenomenon is called overfitting.
- That is, being correct by chance.
- Intuitively, overfitting occurs when our hypothesis fits the training data "too well".

Empirical Risk Minimization with Inductive Bias

- We have just demonstrated that the ERM rule might lead to overfitting.
- Rather than giving up on the ERM paradigm, we will look for ways to rectify it. We will search for conditions under which there is a guarantee that ERM does not overfit, namely, conditions under which when the ERM predictor has good performance wrt the training data, it is also highly likely to perform well over the underlying data distribution.
- A common solution is to apply the ERM learning rule over a restricted search space.

Empirical Risk Minimization with Inductive Bias

- Formally, the learner should choose in advance (before seeing the data) a set of predictors. This set is called a hypothesis class and is denoted by H. Each $h \in H$ is a function from X to Y. For a given class H, and a training sample, S, the ERM_H learner uses the ERM rule to choose a predictor $h \in H$, with the lowest possible error over S. Formally,

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

  where argmin stands for the set of hypotheses in H that achieve the minimum value of $L_S(h)$ over H.

Empirical Risk Minimization with Inductive Bias

- By restricting the learner to choosing a predictor from H, we bias it toward a particular set of predictors. Such restrictions are often called an inductive bias.
- Since the choice of such a restriction is determined before the learner sees the training data, it should ideally be based on some prior knowledge about the problem to be learned. For example, for the papaya problem we may choose the class H to be the set of predictors that are determined by axis aligned rectangles. We will later show that ERM_H over this class is guaranteed not to overfit. On the other hand, the example of overfitting that we have seen previously, demonstrates that choosing H to be a class of predictors that includes all functions that assign the value 1 to a finite set of domain points does not suffice to guarantee that ERM_H will not overfit.

- A fundamental question in learning theory is, over which hypothesis classes ERM_H learning will not result in overfitting.
- Intuitively, choosing a more restricted hypothesis class better protects us against overfitting but at the same time might cause us a stronger inductive bias.
- We will study this fundamental tradeoff.

Finite Hypothesis Classes

- The simplest type of restriction on a class is imposing an upper bound on its size (that is, the number of predictors h in H).
- We will show that if H is a finite class then ERM_H will not overfit, provided it is based on a sufficiently large training sample (this size requirement will depend on the size of H).

Finite Hypothesis Classes

- Limiting the learner to prediction rules within some finite hypothesis class may be considered as a reasonably mild restriction.
- For example, H can be the set of all predictors that can be implemented by a Python program written in at most 10^9 bits of code.

Finite Hypothesis Classes

- Another example of H is the class of axis aligned rectangles for the papaya learning problem, with discretized representation.

Performance Analysis of ERM_H

- H is a finite class.
- For a training sample, S, labeled according to some $f : X \to Y$, let h_S denote a result of applying ERM_H to S, namely,

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

Performance Analysis of ERM_H

The Realizability Assumption:

There exists $h^* \in H$ s.t. $L_{D,f}(h^*) = 0$.

Note that this assumption implies that with probability 1 over random samples, S, where the instances of S are sampled according to D and are labeled by f, we have $L_S(h^*) = 0$.

Performance Analysis of ERM_H

- … to be continued …