The Statistical Learning Framework

Lecturer: Kamalika Chaudhuri

April 24, 2020

Most of standard machine learning implicitly assumes the statistical learning framework, so we will begin by formally describing it. This lecture will mostly focus on supervised machine learning – specifically, classification – although similar frameworks apply to both other kinds of supervised learning such as regression and unsupervised learning.

1 The Framework

Instances are drawn from an instance space $\mathcal{X}$ and labels from a label space $\mathcal{Y}$. There is an underlying though unknown data distribution D from which instance and label pairs are drawn. Specifically, D is a distribution over $\mathcal{X} \times \mathcal{Y}$, and any sample from D is an (x, y) pair. The main assumption in the statistical learning framework is that all data – training, validation, test – are independent, identically distributed samples from this distribution D.

Observe that D is a joint distribution over (X, Y). We describe D by breaking it down into two components:

$$D(x, y) = \Pr(X = x) \times \Pr(Y = y | X = x)$$

The first $\Pr(X = x)$ is the marginal over the instances X, and we will denote it by $\mu(x)$. $\mu(x)$ represents the distribution of the unlabeled data. The second component $\Pr(Y = y|X = x)$ is the conditional distribution of the labels Y at an instance x, and is usually denoted by $\eta(x)$. Let us start with some examples of D.

- $\mathcal{X} = [0, 1]^2$, $\mathcal{Y} = \{-1, +1\}$. Suppose $\mu(x)$ is uniform over $[0, 1]^2$, and $\Pr(Y = 1|X = x) = 1$ if $x_1 + x_2 \leq 1$, and 0 otherwise. This is an example where the conditional distribution of labels is either 0 or 1 – and hence very certain. This happens in applications like image classification over natural images – where there is no uncertainty.
- $\mathcal{X} = [0, 1]^2$, $\mathcal{Y} = \{-1, +1\}$. $\mu(x) = (1 - e^{-x_1})/(1 - e^{-1})$, and $\Pr(Y = 1|X = x) = 1/(1 + e^{-x_1-x_2})$. Observe that this forms a valid distribution – as $\mu(x)$ integrates to 1. However, the conditional distribution is no longer just 0 or 1. This is the case in applications such as healthcare. If we are trying to predict if a patient will need to be readmitted to the hospital for example, we can never be absolutely certain based on the information we have.

Observe also that while we assume the existence of an underlying distribution D, we do not assume that we know D – for example, we do not know the parameters of D. Instead, we have samples from D – which form our training, test, and validation data sets.

In classification, our goal is to construct a predictor or a classifier, which is a function $c : \mathcal{X} \to \mathcal{X}$. We frequently talk about all classifiers of a certain type, which we call a concept class C. For example, C could be the set of all binary linear classifiers over d features.

The true error of a classifier c with respect to an underlying data distribution D is defined as:

$$\text{err}(c) = \Pr_{(X,Y)\sim D}(c(X) \neq Y)$$

The true error cannot usually be directly computed as D is unknown; we instead use as a proxy the empirical error of c on a dataset S that is drawn from D. If $S = \{(x_i, y_i), i = 1, \ldots, n\}$, then this empirical error is defined as:

$$\widehat{\text{err}}(c) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}(c(x_i) \neq y_i)$$

1.1 The Bayes Optimal Classifier

Given a data distribution D, the classifier that minimizes the true error over this distribution is called the Bayes Optimal Classifier. When $\mathcal{Y} = \{-1, 1\}$, for any x in the support of D, the Bayes Optimal classifier $h^*(x)$ is described as:

$$\begin{aligned} h^*(x) &= 1, \text{if } \Pr(Y = 1|x) \geq 1/2 \\ &= -1, \text{otherwise} \end{aligned}$$

Observe that the Bayes Optimal classifier itself cannot be directly computed – since we do not have access to the distribution D – but there are a number of classification algorithms that produce classifiers that are guaranteed to converge to the Bayes Optimal in the limit. Examples of such algorithms are k-nearest neighbors when k is set in a certain way, decision trees and kernel classifiers.

2 The Machine Learning Pipeline

The standard machine learning pipeline has two main phases – training and testing. In the training phase, we use a dataset that we call the training data to build or train a classifier. In the test phase, we evaluate this classifier using a separate test dataset which is also drawn from the same distribution, but is disjoint from the training data.

In light of the statistical learning framework, the goal of the training phase is to find a classifier c in the concept class C to minimize the true error

$$\Pr_{(X,Y)\sim D}(c(X) \neq Y)$$

This cannot be directly done because of two reasons. The first is that D is unknown; the solution to this is to use instead the empirical error on the training data. The second reason is that even if we use the empirical error, the zero-one classification loss is a discontinuous loss which is hard to optimize; for this reason, we use instead a more tractable loss. Thus, during training, we find a classifier c that minimizes:

$$\frac{1}{n}\sum_{i=1}^{n} \ell(c(x), y)$$

where ℓ is a convex loss. Examples of ℓ include the logistic loss where $\ell(y, y') = \log(1 + e^{-yy'})$ or the hinge loss where $\ell(y, y') = \max(0, 1 - yy')$.

2.1 Guarantees in the Statistical Learning Framework

What can we expect out of a classifier learnt in the statistical learning framework? Suppose the framework holds – that is, training and test data are drawn independently and identically from D, and suppose also that $\hat{h}$ is the classifier obtained at the end of training. Here is what we can expect.

- If all future data is drawn i.i.d from D, on expectation, the error of $\hat{h}$ in the future will be its true error.
- If the test data set is large enough, then the test error of $\hat{h}$ is close to its true error with high probability.
- Finally, if the training data set is large enough relative to the complexity of the concept class C, then the training error is also close to the true error. In other words, the classifier generalizes.

3 Departures

There are a number of ways in which most real applications of machine learning depart from the statistical learning framework. Here are some examples.

- **Distribution Shift.** Distribution shift happens when the distribution of test data is different from the training data distribution. For example, we might train a classifier to recognize tumors from medical images on a sample of patients in San Diego, and test it at a hospital in New York. Patients in New York might have different kinds of tumors than in San Diego, and hence the distribution is different.

  There are two main kinds of distribution shift – when only the distribution $\mu(x)$ of unlabeled examples change, and when $\Pr(Y = y|x)$ changes. The first kind is also known as covariate shift, and the tumor recognition problem is an example of this. The second case when $\Pr(Y|x)$ also changes is known as domain adaptation, and is more challenging. An example of this would be classifying news articles by topic; prior to 2016, an article on Donald Trump might be on entertainment, but now the topic is politics. Of course, if the data distribution shifts drastically from training to test, the trained classifier will be useless in predicting the test examples. So in general, our assumption is that there is only a limited amount of distribution shift.

- **Robustness.** The statistical learning framework does not talk about adversaries, but in real applications there may be malicious adversaries who might want to interfere with the machine learning classifiers. We usually study two main kinds of adversaries – training-time, and test-time.

  The training-time adversary can modify a small amount of training examples, and their goal is to cause the trained classifier to change in some specific way. Examples of training-time attacks are data poisoning attacks, where the adversary's goal is to cause the classifier to misclassify, and backdoor attacks, where the goal is to make the classifier produce a specific output on certain specific inputs. For example, a malware developer might wish to introduce some extra examples to a repository so that the learnt classifier lets his malware through.

The test-time adversary has no control over the training data or the training process, and instead only has access to a classifier at test-time. Their goal is to modify legitimate test inputs slightly so as to cause a classifier to misclassify. For example, an adversary might put strategic stickers on a stop sign so that a computer vision classifier will misclassify it as an yield sign, and run into an accident.

- **Privacy.** A lot of machine learning classifiers are trained on sensitive data, such as medical information, financial and genetic information. How can we do this while preserving the privacy of individuals in the training data is yet another aspect that is not handled by the statistical learning framework.

- **Interpretability.** In many applications of machine learning, the classifier needs to be able to explain how it came across its decision. This is a topic of much research, and is in its infancy. Again this is not something handled by the statistical learning framework.

- **Causal Inference.** Machine learning classifiers are completely associational, and say nothing causal about the data. This is another aspect ignored by the statistical learning framework.

- **Fairness.** As machine learning classifiers are increasingly used in societal applications – such as screening candidates for employment and loans, and risk scores for reoffense, it is increasingly important to understand how they work for minority, at-risk and underserved populations. Again, the statistical learning framework does not address any of these issues.

We will talk about the first four or five aspects in this class.