CSCI 5525 Machine Learning Fall 2019

Lecture 12: Model selection and Structural Risk Minimization

Oct 10th 2019

Lecturer: Akshay Krishnamurthy *Scribe: Akshay Krishnamurthy*

1 Recap

In the last lecture, Steven kick-offed some discussion of statistical learning theory. Let's recap briefly: we have labeled samples $\{(x_i, y_i)\})_{i=1}^n$ all drawn independently and identically from some unknown distribution P. For any predictor $f : \mathcal{X} \to \mathcal{Y}$ the empirical risk is defined as:

$$\hat{\mathcal{R}}(f) := \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, f(x_i))$$

where $\ell(\cdot, \cdot)$ is our loss function. Today we will *always* use the $0/1$-loss $\ell_{0/1}(y, \hat{y}) = \mathbf{1}\{y \neq \hat{y}\}$.

The population risk is the main object of interest and it is defined as $\mathcal{R}(f) = \mathbb{E}_{(X,Y)\sim P}\ell(Y, f(X))$. Typically guarantees control the *excess risk* and last lecture, Steven showed you this decomposition: for any algorithm that outputs $\hat{f}$ from some function class $\mathcal{F}$, we have

$$\mathcal{R}(\hat{f}) - \mathcal{R}(\tilde{f}) = \underbrace{\hat{\mathcal{R}}(\tilde{f}) - \mathcal{R}(\tilde{f})}_{\text{Sampling error}} + \underbrace{\hat{\mathcal{R}}(\hat{f}) - \hat{\mathcal{R}}(\tilde{f})}_{\text{computational error}} + \underbrace{\mathcal{R}(\hat{f}) - \hat{\mathcal{R}}(\hat{f})}_{\text{generalization error}}$$

Where $\tilde{f}$ is any function in $\mathcal{F}$, typically we take $\tilde{f} := \min_{f\in\mathcal{F}} \mathcal{R}(f)$. In this lecture lets assume that we *exactly* compute the ERM, so the computational error is negative, and we will always just upper bound it by zero. Last lecture, Steven showed you how to upper bound the excess risk for finite function classes. When $\hat{f}$ is the ERM, we get that with probability at least $1 - \delta$:

$$\mathcal{R}(\hat{f}) - \min_{\tilde{f}\in\mathcal{F}} \mathcal{R}(\tilde{f}) \leq \sqrt{\frac{\ln(2/\delta)}{2n}} + \sqrt{\frac{\ln(2|\mathcal{F}|/\delta)}{2n}} \leq \sqrt{\frac{2\ln(2|\mathcal{F}|/\delta)}{n}}.$$

2 Motivation

The departure point for today's lecture is that this guarantee is *not* what we actually we want. We do not really care about the *excess risk* if the best predictor in our class is terrible, we want a guarantee that the actual risk $\mathcal{R}(\hat{f})$ is low. This is not provided by the excess risk bound at all!

What we really want is to compare our predictor with the *Bayes optimal predictor* $f^\star(x) := \text{argmax}_y\{P(Y = y|X = x)\}$. This predictor achieves the lowest loss you could hope to achieve on the classification problem defined by distribution P. (If there is no noise, then $\mathcal{R}(f^\star) = 0$, which is great!) So what we really would like to bound is:

$$\mathcal{R}(\hat{f}) - \mathcal{R}(f^\star) = \underbrace{\mathcal{R}(\hat{f}) - \min_{\tilde{f}\in\mathcal{F}} \mathcal{R}(\tilde{f})}_{\text{Estimation error}} + \underbrace{\min_{\tilde{f}\in\mathcal{F}} \mathcal{R}(\tilde{f}) - \mathcal{R}(f^\star)}_{\text{Approximation error}}$$

From last lecture, we know how to get a handle on the estimation error, but understanding the approximation error is typically much harder. Note that the data and the learning algorithm do not appear here at all. The approximation error is small if the best predictor in the class you chose has good performance, but it does not decrease as we get more and more data, unlike the training error.

Intuitively the approximation error is the best performance we could get, in the limit of infinite training data, assuming we commit to the function class $\mathcal{F}$. So making sure we have small approximation error involves *choosing the right function class*, but there is always a tradeoff here. Typically to lower the approximation error, we will have to choose a bigger function class, which means the estimation error will be higher.

Example 2.1. *As a simple example, consider a binary classification problem where the feature is* $\mathcal{X} = [0, 1)$. *We know that for classification,* $f^\star(x) \in \{0, 1\}$. *So define*

$$\mathcal{F}_m = \left\{ x \mapsto \sum_{i=1}^{2^m} b_i \mathbf{1}\{x \in H_i\} : H_i = [(i-1)/2^m, i/2^m), b_i \in \{0, 1\} \right\}.$$

These are the piecewise constant functions with 2^m *splits where the splits happen at powers of* 2. *Naturally, as we drive* m *to infinity, we will eventually capture* $f^\star$. *And as* m *increases the approximation error of* $\mathcal{F}_m$ *is going to zero. But as we can see* $|\mathcal{F}_m| = 2^m$, *so as* m *increases, the estimation error is increasing. As* n *increases, estimation error is decreasing. So as* n *increases we can choose larger and larger function classes* $\mathcal{F}_m$. *But for a given* n, *we need to choose the right size function class, to properly tradeoff estimation and approximation error. Today we will discuss some strategies for doing this.*

3 Choosing a function class, aka Model Selection

Basically all of the approaches for this take the following form:

1. Choose many function classes $\mathcal{F}_1, \mathcal{F}_2, \ldots,$, the can even be an infinite sequence.
2. Compute the empirical risk minimizer $\hat{f}_m \in \mathcal{F}_m$ for each function class.
3. Use the data, in some way, to choose a global $\hat{f}$ from $\{\hat{f}_1, \hat{f}_2, \ldots, \}$.

The two methods we will discuss today differ in how they choose the global $\hat{f}$. The goal is to get what is referred to as an *oracle inequality*, which takes the following form:

$$\mathcal{R}(\hat{f}) \leq \min_m \left\{ \min_{\tilde{f}_m \in \mathcal{F}_m} \mathcal{R}(\tilde{f}_m) + \text{gen-err}(\mathcal{F}_m) + \text{penalty}(m) \right\}.$$

If penalty(m) is very small (which it will be), then this is a very good guarantee, much more useful than the one from the previous lecture. With this oracle inequality, if for some class index $m^\star$, we have that $f^\star \in \mathcal{F}_{m^\star}$ then we immediately get an actual risk bound:

$$\text{If } f^\star \in \mathcal{F}_{m^\star} : \mathcal{R}(\hat{f}) \leq \mathcal{R}(f^\star) + \text{gen-err}(\mathcal{F}_{m^\star}) + \text{penalty}(m^\star)$$

This is of course also true in the setup of the previous lecture, but by allowing us to choose many function classes and selecting between them, we get to adapt to the complexity of $f^\star$.

The oracle inequality is actually much more powerful. If there is a class $\mathcal{F}_i$ that is relatively small that has low approximation error, then we might want to choose that class, because it provides a better balance between estimation and approximation error. Thus the oracle inequality lets us *adapt* to the complexity of the problem!

3.1 Strategy 1: Cross validation

The simplest strategy is *cross validation*, which I imagine many of you have seen or used in some form. The simplest version assumes we have a finite set of function classes $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_M$. The simplest cross validation strategy is a data splitting approach, with a formal specification as follows:

1. Split training data in half, where one sample $\{(x_i, y_i)\}_{i=1}^{n/2}$ is the *training sample* and the other sample $\{(x_i, y_i)\}_{i=n/2+1}^{n}$ is the *validation sample*.

2. For each class $\mathcal{F}_m$, compute the empirical risk minimizer $\hat{f}_m$ *on the training data only*

3. Choose the class index $\hat{m} = \operatorname{argmin}_{m \in [M]} \hat{\mathcal{R}}_V(\hat{f}_m)$ where $\hat{\mathcal{R}}_V(f) := \frac{1}{n/2} \sum_{j=n/2+1}^{n} \ell(f(x_i), y_i)$, is the empirical risk on the validation set. Then set $\hat{f} = \hat{f}_{\hat{m}}$.

The main result for this strategy is

Theorem 3.1. *With probability at least* $1 - \delta$*, we have*

$$\mathcal{R}(\hat{f}) \leq \min_{m \in [M]} \left\{ \min_{\tilde{f}_m \in \mathcal{F}_m} \mathcal{R}(\tilde{f}_m) + 4\sqrt{\frac{\ln(|\mathcal{F}|/\delta)}{n}} + 4\sqrt{\frac{\ln(4M)}{n}} \right\}$$

Proof. We know that for each class $m \in [M]$, with probability at least $1 - \delta_m$, we have the generalization bound we previously derived, but we have to rebind $n \mapsto n/2$ since we are using half of the sample. Further, we want this bound to hold for all $m \in [M]$ and we also need some other events to hold, so we have to take a union bound and rebind $\delta \mapsto \delta/(2M)$. Thus we get

$$\forall m \in [M] : \mathcal{R}(\hat{f}_m) - \min_{\tilde{f}_m \in \mathcal{F}_m} \mathcal{R}(\tilde{f}_m) \leq 2\sqrt{\frac{\ln(4M|\mathcal{F}_m|/\delta)}{n}}.$$

So far we have only used the training data. But when we look at the validation data, the learned predictors $\hat{f}_m$ are *completely independent* of this sample. This means we can control their deviations using the "sampling error" argument from the previous lecture. Again with another union bound, we get

$$\forall m \in [M] : \left| \hat{\mathcal{R}}_V(\hat{f}_m) - \mathcal{R}(\hat{f}_m) \right| \leq \sqrt{\frac{\ln(4M/\delta)}{n}}$$

All of these inequalities hold, simultaneously, except for with probability $1 - \delta$.

Now our algorithm choose $\hat{f} = \hat{f}_{\hat{m}}$ where $\hat{m}$ is chosen to minimize the validation error, when comparing with any other $m \in [M]$, this gives

$$\begin{aligned}
\mathcal{R}(\hat{f}) &= \hat{\mathcal{R}}_V(\hat{f}_{\hat{m}}) + \mathcal{R}(\hat{f}_{\hat{m}}) - \hat{\mathcal{R}}_V(\hat{f}_{\hat{m}}) && \text{Add and subtract } \hat{\mathcal{R}}_V \\
&\leq \hat{\mathcal{R}}_V(\hat{f}_{\hat{m}}) + \sqrt{\frac{\ln(4M/\delta)}{n}} && \text{Concentration of } \hat{\mathcal{R}}_V \\
&\leq \hat{\mathcal{R}}_V(\hat{f}_m) + \sqrt{\frac{\ln(4M/\delta)}{n}} && \text{Definition of } \hat{m} \\
&\leq \mathcal{R}(\hat{f}_m) + 2\sqrt{\frac{\ln(4M/\delta)}{n}} && \text{Concentration of } \hat{\mathcal{R}}_V \\
&\leq \min_{\tilde{f}_m \in \mathcal{F}_m} \mathcal{R}(\tilde{f}_m) + 2\sqrt{\frac{\ln(4M|\mathcal{F}_m|/\delta)}{n}} + 2\sqrt{\frac{\ln(4M/\delta)}{n}} && \text{Generalization bound} \\
&\leq \min_{\tilde{f}_m \in \mathcal{F}_m} \mathcal{R}(\tilde{f}_m) + 4\sqrt{\frac{\ln(|\mathcal{F}_m|/\delta)}{n}} + 4\sqrt{\frac{\ln(4M)}{n}} && \text{Simple algebra } \square
\end{aligned}$$

K-fold and LOO. There are many different ways to do cross validation, you may have heard of k-fold cross validation, and leave-one-out cross validation. The mostly just correspond to different ways to split the dataset.

We actually used leave-one-out cross validation in a paper a few years ago. The goal was to estimate the entropy of a distribution, where the entropy for a discrete distribution over domain $\mathcal{X}$ is defined as

$$H(P) = -\sum_{x \in \mathcal{X}} P(x) \log_2 P(x) = \mathbb{E}_{x \sim P}\left[-\log_2 P(x)\right].$$

This last expression motivates a data-splitting estimator. Use half of the data to estimate the probability mass function $\hat{P}_{1:n/2}$, and then use the other half of the data to estimate the outer expectation:

$$\hat{H}_{\text{split}} = \frac{1}{n/2} \sum_{i=n/2+1}^{n} -\log \hat{P}_{1:n/2}(x_i).$$

The leave-one-out approach forms n different estimators for the PMF, where the i^{th} estimator is based on all but the i^{th} data point, let us call this $\hat{P}_{-i}$. Then we evaluate this estimate on the i^{th} sample:

$$\hat{H}_{\text{LOO}} = \frac{1}{n} \sum_{i=1}^{n} -\log \hat{P}_{-i}(x_i)$$

This is more data-efficient in practice. But the estimator introduces lots of statistical dependencies, so the analysis is much more difficult. In theory, you only gain in constant factors, so people don't really analyze this.

Adaptive Data Analysis. One thing to be careful about is how you use your validation data. If you notice in the description of cross validation above, we committed to the classes $\mathcal{F}_1, \ldots, \mathcal{F}_M$, before looking at the validation data. If we run this procedure and find that no predictor $\hat{f}_m$ is very good, can we

just go back and create some new classes $\mathcal{F}_{M+1}$? At a high level, the answer is no! Now the choice of $\mathcal{F}_{M+1}$, and therefore $\hat{f}_{M+1}$ depends on the validation data, so the validation error for this predictor will not be a good estimate of it's performance.

More generally, if you interleave choosing hypotheses and evaluating on the validation set, you risk overfitting to the validation set. This has been a major issue in scientific investigation, where it is common to test many hypotheses, chosen adaptively, on a single, expensive-to-collect, dataset. As a result, this has come to be known as *adaptive data analysis*, which is a thriving research area within machine learning. Actually Steven is kind of an expert in this topic, so if you are curious about it, you should ask him more!

3.2 Strategy 2: Structural risk minimization

The other strategy I wanted to discuss is called *structural risk minimization*. This is a form of bound minimization algorithm, which proceeds as follows

1. Compute ERMs $\hat{f}_m$ for each class $\mathcal{F}_m$, using all of the data
2. Compute index $\hat{m} = \operatorname{argmin}_m \hat{\mathcal{R}}(\hat{f}_m) + \sqrt{\frac{2\ln(2w(m)|\mathcal{F}_m|/\delta)}{n}}$ where $w(m)$ is some weighting function satisfying $\sum_m w(m) \leq 1$. Return $\hat{f} = \hat{f}_{\hat{m}}$.

You should think of $w(m)$ as some function that encodes our prior beliefs about what function class is a good one. If we have just M function classes, we could take $w(m) = 1/M$, which is what we did before. One advantage of using these weighting functions is that you could in principle have infinite sequences of classes if you used weighting $w(m) = \frac{6}{\pi^2 m^2}$ which is a convergent series.

Theorem 3.2. *With probability* $1-\delta$ *we have*

$$\mathcal{R}(\hat{f}) \leq \min_m \left\{ \min_{\tilde{f}_m \in \mathcal{F}_m} \mathcal{R}(\tilde{f}_m) + 2\sqrt{\frac{2\ln(2w(m)|\mathcal{F}_m|/\delta)}{n}} \right\}.$$

Proof. For any class m

$$\begin{aligned}
\mathcal{R}(\hat{f}) &\leq \hat{\mathcal{R}}(\hat{f}_{\hat{m}}) + \sqrt{\frac{2\ln(2w(\hat{m})|\mathcal{F}_{\hat{m}}|/\delta)}{n}} && \text{Generalization + Union bd} \\
&\leq \hat{\mathcal{R}}(\hat{f}_m) + \sqrt{\frac{2\ln(2w(m)|\mathcal{F}_m|/\delta)}{n}} && \text{Def of } \hat{m} \\
&\leq \min_{\tilde{f}_m \in \mathcal{F}_m} \mathcal{R}(\tilde{f}_m) + 2\sqrt{\frac{2\ln(2w(m)|\mathcal{F}_m|/\delta)}{n}}. && \text{Generalization bd for } \mathcal{F}_m \quad \square
\end{aligned}$$

Remark 3.3. While SRM is a simple approach that is theoretically justified, you should always be wary when concentration inequalities are used algorithmically like they are here. The reason is that this bakes in worst case assumptions about the problem and prevents you from adapting to favorable conditions. The performance of this algorithm is tied to the deviation bound that you can prove – if you can prove a better deviation bound then you get a better algorithm. This is unsatisfying since you'd like the algorithm and the analysis to be detached from each other.