6.883: Online Methods in Machine Learning

Alexander Rakhlin

LECTURE 10

In the last lecture, we showed how uniform deviations allow us to compare the performance of methods minimizing empirical vs expected objectives. We also mentioned that for hinge loss and linear functions, an upper bound on uniform deviations is of the same order as the very convergence of SGD. Let us sketch the argument here.

0.1 Understanding uniform deviations for linear prediction with hinge loss

Through a technique called *symmetrization*, it is possible to prove that uniform deviations

$$\mathbb{E} \sup_{\|w\| \le B} \left\{ \mathbb{E}\boldsymbol{\ell}(\langle w, X\rangle, Y) - \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{\ell}(\langle w, X_i\rangle, Y_i) \right\} \tag{1}$$

are upper bounded by

$$2\mathbb{E} \sup_{\|w\| \le B} \frac{1}{n}\sum_{i=1}^{n} \epsilon_i \boldsymbol{\ell}(\langle w, X_i\rangle, Y_i) \tag{2}$$

where the expectation is over the data and over the independent Rademacher random variables $\epsilon_1, \dots, \epsilon_n$. Recall from Lecture 1 that these are just ± 1 unbiased coin flips. Incidentally, the expression in (2) is called the *Rademacher averages* of the function class

$$\{x \mapsto \boldsymbol{\ell}(\langle w, x\rangle, y) : \|w\| \le B\}$$

The next step, which we also shove under the rug, is that (2) is upper bounded by

$$2\mathbb{E} \sup_{\|w\| \le B} \frac{1}{n}\sum_{i=1}^{n} \epsilon_i \langle w, X_i\rangle, \tag{3}$$

the Rademacher averages of the function class

$$\{x \mapsto \langle w, x\rangle : \|w\| \le B\}.$$

At this point we might want to recall (or prove) that for the Euclidean norm

$$\sup_{\|w\| \le 1} \langle w, z\rangle = \|z\|.$$

Hence, by linearity

$$2\mathbb{E} \sup_{\|w\| \le B} \frac{1}{n}\sum_{i=1}^{n} \epsilon_i \langle w, X_i\rangle = 2\mathbb{E} \sup_{\|w\| \le B} \left\langle w, \frac{1}{n}\sum_{i=1}^{n} \epsilon_i X_i \right\rangle = 2B \times \mathbb{E} \left\| \frac{1}{n}\sum_{i=1}^{n} \epsilon_i X_i \right\| \tag{4}$$

Finally,

$$\mathbb{E}\left\|\sum_{i=1}^n \epsilon_i X_i\right\| = \mathbb{E}\sqrt{\left\|\sum_{i=1}^n \epsilon_i X_i\right\|^2} \le \sqrt{\mathbb{E}\left\|\sum_{i=1}^n \epsilon_i X_i\right\|^2} = \sqrt{\sum_{i=1}^n \mathbb{E}\left\|X_i\right\|^2} \le G\sqrt{n} \tag{5}$$

where $G^2 \ge \mathbb{E}\left\|X\right\|^2$. Putting it all together gives the desired bound

$$2 \times \frac{BG}{\sqrt{n}}$$

on uniform deviations over the Euclidean ball of radius B.

One may ask whether uniform deviations have the same rate as gradient descent only for the case of Euclidean ball $\{w : \|w\| \le B\}$. Consider, as another example, a probability simplex $\Delta = \{w : \sum_{j=1}^d w_j = 1, w_j \ge 0\}$. Rather than gradient descent, one should be using Mirror Descent in this situation (an algorithm we will cover later). The rate of Mirror Descent (equivalently, exponential weights) is

$$c\sqrt{\frac{\log d}{n}}.$$

How about uniform deviations? We have

$$\mathbb{E}\sup_{w\in\Delta}\left\{f(w) - \widehat{f}(w)\right\} \le 2\mathbb{E}\left\|\frac{1}{n}\sum_{i=1}^n \epsilon_i X_i\right\|_\infty \le c'\sqrt{\frac{\log d}{n}}$$

using basically the same argument as for the Euclidean ball, but then noting that the ℓ_∞ norm is a maximum of d random variables with variance $O(1/n)$. Once again, the result coincides with the rate of GD.

As mentioned before, the fact that uniform deviations have the same rate as gradient descent with n iterations is not a coincidence and comes about from a certain connection between uniform convergence for i.i.d. data and for martingales. Since some stochastic processes need to be introduced to understand the connection, we will postpone this treatment towards the end of the course.

0.2 Training error, test error, and overfitting

A typical approach to training a machine learning algorithm is to split the data into a training and testing set (and, possibly, a validation set). The training set is used, well, for fitting the data (e.g. SVM, SGD, or whatever your method is). The test error is supposed to be used once at the very end to assess the performance of the procedure, and the validation set is used to check a variety of parameters and choose the best model. When running an iterative optimization procedure, such as stochastic gradient descent, one often monitors the training error (which should hopefully go down) and the test error. If the model is flexible enough, one will typically see that the training error keeps going down, but the test error starts to increase at some point.

Strictly speaking, this process of monitoring the test error is not kosher, as one stops the optimization method when the test error is small enough. This introduces a bias. One can use a separate subset of data (validation set) to decide when to stop, but the final assessment of the quality of the solution should be made on a separate set of data, only used once. In practice, however, it is impossible to require that a set of data is used once.

What is overfitting? There is no precise definition, but, roughly speaking, it is the point in the training process when the training error becomes "significantly smaller" than the error on a fresh never-used-before sample. If the algorithm outputs a hypothesis $\widehat{w}$ that does not depend on data, the central limit theorem guarantees that the average fit to data $\widehat{f}(\widehat{w})$ is close (to within $O(1/\sqrt{n})$) to the expected error $f(\widehat{w})$. As the algorithm starts to depend more and more on the data, the difference

$$f(\widehat{w}) - \widehat{f}(\widehat{w})$$

between its empirical fit to data and expected loss is only controlled by uniform deviations, which may be too large if the model is large (see figure from last lecture). For instance, for a fixed w that does not depend on data, $f(w) - \widehat{f}(w)$ is on the order of $O(1/\sqrt{n})$ for the example of minimization on the simplex discussed earlier; however, uniform deviations scale with the dimension of the problem. In general, uniform deviations might be significantly worse than what is given by the central limit theorem.

In summary, one needs to be very careful reusing data for tuning parameters, for selecting a model, for monitoring performance. Most importantly, the final reported error of the method should be made on a fresh hold out sample.

We remark that the overfitting issues discussed here arise from our definition of the learning target $\mathbb{E}\boldsymbol{\ell}(w,(X,Y))$, phrased in terms of an unknown distribution $P_{X\times Y}$. All methods in this model of learning are *indirect*: the optimize some empirical objective in the hope of optimizing an expected objective. There is, however, a different model of learning (called 'online learning') that is more *direct*. Here, we will optimize the actual target of interest, rather than some indirect quantity. We will, however, have to make the protocol online.

1. ONLINE LEARNING

In the next few lectures we consider the *supervised online learning problem*, where the data comes in pairs $(x_1,y_1),\dots,(x_n,y_n)$, as before. [1]

The supervised online learning protocol is:

For $t = 1,\dots,n$
- Observe $x_t \in \mathcal{X}$
- Predict $\widehat{y}_t \in \mathbb{R}$
- Observe outcome $y_t \in \mathbb{R}$

Let $\boldsymbol{\ell}(\widehat{y}_t, y_t)$ measure the quality of prediction $\widehat{y}_t$ with respect to the actual outcome y_t. The goal is to develop methods that have small loss

$$\frac{1}{n}\sum_{t=1}^{n} \boldsymbol{\ell}(\widehat{y}_t, y_t). \tag{6}$$

The average loss (6) cannot be small for all sequences. The goal is to decide which sequences we care about, and have the algorithm focus on them. The choice of sequences we care about will be done *implicitly*, so we do not need to worry about enumerating them one by one.

To start, let us go back to Lecture 1:

[1] A related commonly studied setting is that of Online Convex Optimization (OCO); contrary to a popular opinion, OCO does not subsume supervised online learning.

1.1 Binary prediction with indicator loss and no side information

We proved in Lecture 1 that there exists a randomized algorithm that attains

$$\forall y_1, \dots, y_n \in \{\pm 1\}, \qquad \mathbb{E}\left[\frac{1}{n}\sum_{t=1}^{n} \mathbf{1}\{\widehat{y}_t \neq y_t\}\right] \leq \phi(y_1, \dots, y_n) \tag{7}$$

if and only if $\mathbb{E}\phi(\epsilon_1, \dots, \epsilon_n) \geq 1/2$. Here we only consider $\phi : \{\pm 1\}^n \to \mathbb{R}$ such that

$$|\phi(\dots, +1, \dots) - \phi(\dots, -1, \dots)| \leq 1/n. \tag{8}$$

As discussed in the first lecture, ϕ encodes the prior knowledge about what sequences we might encounter. One way to define it is by choosing a subset $F \subseteq \{\pm 1\}^n$ of the hypercube and setting

$$\phi(\boldsymbol{y}) = \phi(y_1, \dots, y_n) = d_H(\boldsymbol{y}, F) \triangleq \min_{f \in F} \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\{f_i \neq y_i\}, \tag{9}$$

the normalized Hamming distance between $\boldsymbol{y}$ and F. That is, we are insisting that the number of mistakes made by our algorithm is 0 on sequences in F, and degrades linearly with the distance to the set. This goal will be unattainable, but something very close to it will be possible.

First, we check that condition (8) holds for the normalized Hamming distance. Second, we need to verify $\mathbb{E}\phi(\boldsymbol{\epsilon}) \geq 1/2$. This condition fails, since expected distance from a random point on the hypercube to a nonempty F is less than $1/2$. But it cannot be too much smaller than $1/2$ if F is not too large. Let's see:

$$\mathbb{E}\min_{f \in F} \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\{f_i \neq \epsilon_i\} = \mathbb{E}\min_{f \in F} \frac{1}{n}\sum_{i=1}^{n} \frac{1}{2}(1 - f_i\epsilon_i) = \frac{1}{2} - \frac{1}{2}\mathbb{E}\max_{f \in F} \frac{1}{n}\sum_{i=1}^{n} f_i\epsilon_i \tag{10}$$

So, while Hamming distance does not pass the test, the function

$$\phi(\boldsymbol{y}) = d_H(\boldsymbol{y}, F) + \frac{1}{2n}\mathbb{E}\max_{f \in F} \langle f, \boldsymbol{\epsilon} \rangle \tag{11}$$

does. Note that the amount by which we need to increase the Hamming distance to F so that the expected value is at least $1/2$ is precisely (half of) the Rademacher averages of the set F.

Recall from Lecture 1 that there is a closed-form solution that guarantees (7), whenever $\mathbb{E}\phi \geq 1/2$. The solution is

$$q_t(y_1, \dots, y_{t-1}) \triangleq n(\mathbb{E}\phi(y_{1:t-1}, -1, \epsilon_{t+1:n}) - \mathbb{E}\phi(y_{1:t-1}, +1, \epsilon_{t+1:n})) \tag{12}$$

where the expectation is over the suffix $\epsilon_{t+1:n}$ of independent Rademacher random variables. The value q_t is the mean of the distribution on $\{-1, +1\}$ from which the algorithm should draw the prediction $\widehat{y}_t$.

We can further go to double randomization: at time t, draw coin flips $\epsilon_{t+1:n}$ and define

$$\widehat{q}_t(y_1, \dots, y_{t-1}, \epsilon_{t+1:n}) \triangleq n(\phi(y_{1:t-1}, -1, \epsilon_{t+1:n}) - \phi(y_{1:t-1}, +1, \epsilon_{t+1:n})) \tag{13}$$

One may check that whenever q_t in (12) satisfies (7), the double randomization strategy (13) also satisfies (7).