6.883: Online Methods in Machine Learning

Alexander Rakhlin

LECTURE 13

0.1 Recap

We started discussing the problem of online prediction with the simple result of Cover. In this setting, there is just a sequence of bits $y_1, \dots, y_n$ to predict, and we do so by introducing ϕ as the prior knowledge about what sequences we expect to see. Our algorithms work for all sequences $y_1, \dots, y_n$. We then extended the result to include i.i.d. side-information $x_1, \dots, x_n$. This is a hybrid i.i.d.-adversarial (or, i.i.d.-worst-case) scenario, which we study in a bit more detail below. Then, we will turn to the easier case of "transductive learning" where $x_1, \dots, x_n$ are known ahead of time. Later in the course we will also discuss what can be achieved for arbitrary $x_1, \dots, x_n$ sequences of side information. We can list these scenarios roughly in the order of their "complexity":

1. No side information
2. Transductive: $x_1, \dots, x_n$ are known ahead of time
3. i.i.d. $x_1, \dots, x_n$ (or a stochastic process which we can simulate)
4. Arbitrary $x_1, \dots, x_n$, possibly adversarially chosen

0.2 Defining ϕ as a function class benchmark

As discussed in the previous lecture, a standard way of defining $\phi : \{\pm 1\}^n \to \mathbb{R}$ is to choose a subset $F \subseteq \{\pm 1\}^n$ of the hypercube and set

$$\phi(\boldsymbol{y}) = d_H(\boldsymbol{y}, F) + C_n$$

The advantage of this definition is that the smoothness condition is automatically satisfied, and C_n can be chosen large enough to ensure $\mathbb{E}\phi \geq 1/2$.

The same approach can be taken when covariates are present. In statistics and machine learning, one often posits that the relationship between x and y is explained well, in the vague sense of

$$y_t \approx f(x_t),$$

by some unknown function $f : \mathcal{X} \to \mathbb{R}$ (in our case, $\mathcal{X} \to \{\pm 1\}$). While the function is unknown, one chooses a model class $\mathcal{F}$ in the hopes of capturing f or a good approximation of it.

Let $\mathcal{F}$ be a collection of functions $\mathcal{X} \to \{\pm 1\}$. We can then define

$$F = \{(g(x_1), \dots, g(x_n)) : g \in \mathcal{F}\} \triangleq \mathcal{F}|_{x_{1:n}},$$

the subset of the hypercube realized by functions in $\mathcal{F}$ when applied to covariates. We may say that $\mathcal{F}|_{x_{1:n}}$ is a projection of $\mathcal{F}$ onto data.

We then define

$$\phi(x_{1:n}, y_{1:n}) = d_H(y_{1:n}, \mathcal{F}|_{x_{1:n}}) + C_n.$$

The calculation of C_n is exactly as before, and we find that

$$C_n = \frac{1}{2n}\mathbb{E}\sup_{g\in\mathcal{F}}\sum_{t=1}^{n}\epsilon_t g(x_t),$$

known as the **Rademacher averages** of $\mathcal{F}$ (the expectation is over $x_{1:n}, \epsilon_{1:n}$). This complexity of $\mathcal{F}$ is a central object in understanding rates of convergence in Statistics and Statistical Learning.

We have the following corollary of the Theorem from the preceding lecture:

Corollary 1. *Consider the hybrid i.i.d.-adversarial scenario. Given a class of real-valued functions $\mathcal{F}$ on $\mathcal{X}$, there is a randomized prediction strategy such that for any sequence $y_1, \ldots, y_n \in \{\pm 1\}$,*

$$\mathbb{E}\left[\frac{1}{n}\sum_{t=1}^{n}\mathbf{1}\{\widehat{y}_t \neq y_t\}\right] \leq \mathbb{E}\left[\inf_{f\in\mathcal{F}}\frac{1}{n}\sum_{t=1}^{n}\mathbf{1}\{f(x_t) \neq y_t\}\right] + C_n \tag{1}$$

where C_n is the Rademacher averages of $\mathcal{F}$.

If $\mathcal{F}$ contains a function such that $y_t = f(x_t)$ "most of the time," then the number of mistakes made by the algorithm on average is comparable to that of this best function, plus Rademacher averages of $\mathcal{F}$. These averages typically go to zero with increasing n, unless $\mathcal{F}$ is too large. Notably, Rademacher averages are precisely the performance bound one obtains in Statistical Learning with i.i.d. data. Here, we were able to obtain the same result with individual sequence of y's, and in the online (rather than batch) scenario. Vapnik-Chervonenkis theory (a part of the larger field of Empirical Process Theory) gives a precise understanding of how Rademacher averages reflect complexity of the abstract class $\mathcal{F}$.

0.3 Transductive version

In certain cases, $x_{1:n}$ are known and fixed ahead of time. In this case, the algorithm

$$\widehat{q}_t(x_{1:n}, y_{1:t-1}, \epsilon_{t+1:n}) = n\big[\phi(x_{1:n}, y_{1:t-1}, -1, \epsilon_{t+1:n}) - \phi(x_{1:n}, y_{1:t-1}, +1, \epsilon_{t+1:n})\big]$$

is still guaranteeing the bound of $\phi(x_{1:n}, y_{1:n})$, and we no longer need to draw $x_{t+1:n}$. This setting is called *transductive* learning. To see why the same algorithm works, go through the proof of the theorem from last lecture and omit the step where we took expectation with respect to side-information.

One important subtlety is whether the order of $x_1, \ldots, x_n$ needs to be known ahead of time. This depends on ϕ. If ϕ is symmetric with respect to $x_{t+1:n}$ (that is, has the same value for any permutation), then the order of $x_1, \ldots, x_n$ need not be known in advance. One example where the function is symmetric is in the definition of ϕ as a benchmark with respect to a class, as discussed above. The statement we just made requires a proof, and we leave it as an exercise.

The subtlety of "knowing the order" will be discussed in the next example of collaborative filtering.

1. COLLABORATIVE FILTERING: FIRST ATTEMPT

We will now develop a method of predicting, sequentially, whether users like or dislike movies. Rather than predicting a rating (we can work that out later in the course) we shall simply predict thumbs up (+1) or thumbs down (−1).

The setup is as follows. On each round, we observe the identity of the user and the identity of the movie, and make the thumbs up/down prediction. We then observe the actual outcome of whether the user liked the movie. Once the rating for the user-movie pair has been revealed, we do not ever come back to the same user-movie pair (this assumption allow us to treat the setting as *transductive*). We shall start with no information whatsoever and slowly build the matrix of ratings. Arguably, this is a very simplified version of the Netflix task. Yet, typical statistical analyses of matrix completion require much more unrealistic assumptions (such as independence and uniform sampling of the entries).

The user-movie pair $(i_t, j_t) \in [k] \times [p]$ is the side information x_t we receive on round t. For now we will assume that the time horizon is $n = kp$; hence, all $x_1, \ldots, x_n$ are known in advance, except the order in which they arrive. As we will argue, the order does not make a difference.

A function that maps side information to a prediction is simply a matrix with ±1 entries. The benchmark is then defined with respect to

$$\min_{M \in F} \frac{1}{n} \sum_{t=1}^{n} \mathbf{1}\{M(i_t, j_t) \neq y_t\} \tag{2}$$

for some collection F of matrices that we hope will explain well the ratings given by users. What collection should we choose? One of the standard ways to capture the underlying phenomenon is to consider low-rank matrices. Specifically, it is believed that people are not all that different and if the matrix were to be filled out completely, it would be well approximated as

$$M \approx A \times B$$

with $A \in \mathbb{R}^{k \times r}, B \in \mathbb{R}^{r \times p}$ for some $r << k, p$. We could take the set F to be the set of all ±1 matrices of rank r (a parameter of choice). However, this set is computationally difficult, and it is common to replace the low-rank assumption by the assumption of low trace norm.

1.1 Trace-norm-bounded benchmark

Recall that trace (or nuclear) norm is defined as

$$\|M\|_* = \sum_{j=1}^{\min(k,p)} \sigma_j = \operatorname{trace}\left(\sqrt{M^* M}\right). \tag{3}$$

A convexification of the benchmark (2) is

$$\min_{\|M\|_* \leq b} \frac{1}{n} \sum_{t=1}^{n} \mathbf{1}\{M(i_t, j_t) \neq y_t\}, \tag{4}$$

the best explanation of ratings by a low-trace-norm matrix with ±1 entries. The choice b is a parameter of the model, and we now think of it as fixed. We will discuss this model selection choice later.

Since evaluation of the benchmark (4) is a difficult integer program, we relax it further. We interpret each entry of $M \in [-1,1]^{k\times p}$ as the expected value of a randomized strategy that predicts the corresponding user-movie pair. The loss is then

$$\mathbb{E}_{y'\sim M(i_t,j_t)}\mathbf{1}\{y' \neq y_t\} = \frac{1}{2}|M(i_t,j_t) - y_t| = \frac{1}{2} - \frac{1}{2}y_t M(i_t,j_t). \tag{5}$$

The benchmark (and the function ϕ) is now defined as

$$\phi(x_{1:n}, y_{1:n}) = \min_{\|M\|_*\le b, \|M\|_\infty \le 1} \frac{1}{n}\sum_{t=1}^{n}\frac{1}{2}|M(i_t,j_t) - y_t| + C_n \tag{6}$$

where $x_t = (i_t, j_t)$, $\|M\|_\infty$ is the largest element of M in absolute value, and C_n is an appropriate constant that guarantees $\mathbb{E}\phi \ge 1/2$. Smoothness of ϕ with respect to a flip of a coordinate y_t is immediate. Write

$$\begin{aligned}\phi(x_{1:n}, y_{1:n}) - C_n - \frac{1}{2} &= -\frac{1}{2}\sup_{\|M\|_*\le b, \|M\|_\infty\le 1}\frac{1}{n}\sum_{t=1}^{n} y_t M(i_t,j_t)\\ &= -\frac{1}{2n}\sup_{\|M\|_*\le b, \|M\|_\infty\le 1} Y \bullet M \end{aligned}\tag{7}$$

where $Y \bullet M = \sum_{i,j} Y(i,j)M(i,j)$ is the vectorized matrix product, and $Y(i,j)$ is y_t for t corresponding to $(i_t, j_t) = (i,j)$. Assuming $n = kp$, all entries of Y are defined this way.

1.2 Calculating C_n for trace norm benchmark

As before, and in view of (7), the condition $\mathbb{E}\phi \ge 1/2$ means that C_n needs to be chosen to be at least

$$\frac{1}{2n}\mathbb{E}\sup_{\|M\|_*\le b, \|M\|_\infty\le 1} \mathcal{E}\bullet M \tag{8}$$

where $\mathcal{E}$ is a matrix filled with independent Rademacher random variables. By dropping the $\|M\|_\infty$ constraint we get a larger value (supremum is over a larger collection of matrices), and, hence, it is sufficient to set

$$C_n = \frac{1}{2n}\mathbb{E}\sup_{\|M\|_*\le b} \mathcal{E}\bullet M \tag{9}$$

$$= \frac{b}{2n}\mathbb{E}\sup_{\|M\|_*\le 1} \mathcal{E}\bullet M \tag{10}$$

$$= \frac{b}{2n}\mathbb{E}\,\|\mathcal{E}\|_\sigma \tag{11}$$

for the spectral (or, operator) norm (largest singular value). The last equality holds because the two norms (trace and spectral) are dual to each other. An upper bound on the spectral norm of a random matrix is known, and it is

$$\mathbb{E}\,\|\mathcal{E}\| \le c(\sqrt{k} + \sqrt{p})$$

for the $k \times p$ matrix $\mathcal{E}$, with some constant c (see [BvH14] and references therein).

References

[BvH14] Afonso S Bandeira and Ramon van Handel. Sharp nonasymptotic bounds on the norm of random matrices with independent entries. *arXiv preprint arXiv:1408.6185*, 2014.