6.883: Online Methods in Machine Learning

Alexander Rakhlin

LECTURE 1

0.1 Preamble

This course will focus on online methods in machine learning. Roughly speaking, online methods are those that "process one datum at a time." This is a somewhat vague definition, but the scope of methods will be clear as we go through the course. We will focus on the algorithmic aspect ("how it works") and the theoretical aspect ("why it works"). We will also discuss applications, and you will get to implement the methods for homework/projects. For the more application-oriented students there will be plenty of opportunities to try things out in practice; for the more theoretically-inclined, there will be plenty of new mathematical ideas and research directions.

We can divide the scope of online methods into those that have to process one datum at a time because

- there is too much data to fit into memory, or
- the application is itself inherently online.

We will cover both of these and draw many connections between them. While the first group of online methods is often covered in machine learning classes (e.g. run stochastic gradient descent on a neural network), the second is more rare. In this second part, we shall focus on very recent developments in online learning and develop powerful tools for creating new algorithms (that may or may not look like some form of a gradient descent). With these new tools we will be able to derive and implement methods for prediction in social networks, ad placement, and so forth.

0.2 Bit prediction

In 1950's, David Hagelbarger [Hag56] at the Bell Labs built the so-called "mind reading machine" to play the game of matching pennies. His colleague and friend, Claude Shannon [Sha53], built a simplified version (which is currently in the MIT Museum's storage facility in Somerville [Pou14]). According to some accounts, the machine was consistently predicting the sequence of bits produced by untrained players better than 50%. Here is a modern version of this machine: `www.mindreaderpro.appspot.com` by Yoav Freund and colleagues. Try to beat it!

Of course, the only reason the machine might be able to predict the sequence entered by a human is that these sequences tend to be nonrandom. How can we capture the structure in the sequence and exploit it to our advantage? (If you're coming from information theory, you might be familiar with the close connection between predictability of the sequence and the ability to compress it.)

Let us try to formalize the problem. Even in this simple setting there will be a few surprises. Let $y = (y_1, \dots, y_n) \in \{-1, 1\}$ be as sequence of (signed) bits. A deterministic

prediction strategy can be written as $\mathcal{A} = (\widehat{y}_1, \ldots, \widehat{y}_n)$, with $\widehat{y}_t = \widehat{y}_t(y_1, \ldots, y_{t-1}) \in \{-1, 1\}$. If we employ the deterministic strategy $\widehat{y}$ on $(y_1, \ldots, y_n)$, we make

$$\frac{1}{n} \sum_{t=1}^{n} \mathbf{1}\{\widehat{y}_t \neq y_t\}$$

average number of mistakes. You should be able to convence yourself that for any such deterministic strategy there exists a sequence on which this strategy makes mistakes all the time, thus incurring an average loss of 1. This upsetting issue, however, is fixed by considering randomized strategies, as we show next.

A randomized algorithm will be denoted by $\mathcal{A} = (q_1, \ldots, q_n)$, with $q_t = q_t(y_1, \ldots, y_{t-1}) \in [-1, 1]$. Each value q_t is to be understood as parametrizing the mean of a distribution on $\{-1, 1\}$. For any sequence $y_1, \ldots, y_n$, we now consider the expected value of the loss

$$\ell(\mathcal{A}; y_1, \ldots, y_n) = \mathbb{E}\left[\frac{1}{n} \sum_{t=1}^{n} \mathbf{1}\{\widehat{y}_t \neq y_t\}\right],$$

where the expectation is *over the randomization of the algorithm.* The value $\ell(\mathcal{A}; y_1, \ldots, y_n)$ tells us how many mistakes, on average, the randomized algorithm $\mathcal{A}$ is expected to make on the given sequence. An algorithm that guesses randomly will incur the average loss of $1/2$ for any sequence, and so there is no longer a single "bad" sequence for the algorithm.

Can $\ell(\mathcal{A}; y_1, \ldots, y_n)$ be always smaller than 50%? That is, can we find a super algorithm that will predict all the sequences better than a random guess? Let's take a look. Let $\epsilon_1, \ldots, \epsilon_n$ denote independent Rademacher random variables (unbiased coin flips). Then

$$\frac{1}{2^n} \sum_{(y_1, \ldots, y_n)} \ell(\mathcal{A}; y_1, \ldots, y_n) = \mathbb{E}_{\epsilon_1, \ldots, \epsilon_n} \ell(\mathcal{A}; \epsilon_1, \ldots, \epsilon_n) = \frac{1}{2}.$$

Convince yourself of all these steps.

Conclusion? Any algorithm will necessarily pay larger expected loss on some sequences and smaller on the others, so that, on average, the loss is 50%.

This sounds like a simple "negative" result, but we can put a positive spin on it. If we care about particular sequences (e.g. those produced by humans) we may try to develop an algorithm $\mathcal{A}$ that has small expected number of mistakes on these chosen sequences, and large number of mistakes on the ones we will not expect to encounter.

Of course, the next question is: how does one construct a good algorithm, given that we have some idea of the sequences we'll likely encounter in practice. Let $\phi(y_1, \ldots, y_n)$ denote the target expected value we'd like to achieve on sequence $(y_1, \ldots, y_n)$, and let us specify the values of ϕ on all such sequences. We may think of ϕ as a function on the binary hypercube $\{-1, 1\}^n$. Let us also posit that ϕ does not change too fast on nearby vertices:

$$|\phi(\ldots, +1, \ldots) - \phi(\ldots, -1, \ldots)| \leq \frac{1}{n}. \tag{1}$$

If $\mathbb{E}\phi(\epsilon_1, \ldots, \epsilon_n) < 1/2$, no algorithm can achieve $\ell(\mathcal{A}; y_1, \ldots, y_n) = \phi(y_1, \ldots, y_n)$ for all sequences. This follows from the previous argument. But the interesting question is whether we can achieve any ϕ with $\mathbb{E}\phi \geq \frac{1}{2}$. This simple but somewhat surprising result is stated in [Cov65].

Lemma 1. *Let $\phi : \{\pm 1\}^n \to \mathbb{R}$ be such that (1) holds and $\mathbb{E}\phi = 1/2$ under the uniform distribution on $\{\pm 1\}^n$. Then there exists a randomized algorithm $\mathcal{A}$ with*

$$\forall (y_1, \ldots, y_n), \quad \ell(\mathcal{A}; y_1, \ldots, y_n) = \phi(y_1, \ldots, y_n).$$

Proof. Define the following shorthand to ease the proof a bit:

$$\phi_t(y_1, \ldots, y_t) = \mathbb{E}\phi(y_1, \ldots, y_t, \epsilon_{t+1}, \ldots, \epsilon_n)$$

where the expectation is over $\epsilon_{t+1}, \ldots, \epsilon_n$. Showing $\ell(\mathcal{A}; y_1, \ldots, y_n) = \phi(y_1, \ldots, y_n)$ is equivalent to exhibiting an algorithm with

$$\mathbb{E}\left[\sum_{t=1}^{n} \frac{1}{n}\mathbf{1}\{\widehat{y}_t \neq y_t\} + \phi_{t-1}(y_{1:t-1}) - \phi_t(y_{1:t}) - \frac{1}{2n}\right] = 0 \tag{2}$$

(you should check this by simplifying the expression and using the assumptions of the lemma). To show (2), we start from the end and define algorithm $\mathcal{A}$ on round n as

$$q_n \triangleq n\left(\phi(y_1, \ldots, y_{n-1}, -1) - \phi(y_1, \ldots, y_{n-1}, +1)\right).$$

Writing $\mathbf{1}\{a \neq b\} = \frac{1}{2}(1 - ab)$ for $a, b \in \{\pm 1\}$ (a useful representation to keep in mind for the rest of the course), and using linearity of expectation, the expected loss on round n is

$$\mathbb{E}[\mathbf{1}\{\widehat{y}_n \neq y_n\}] = \frac{1}{2}(1 - q_n y_n).$$

We now peel off the last term from the sum in (2):

$$\frac{1}{n}\mathbb{E}[\mathbf{1}\{\widehat{y}_n \neq y_n\}] + \phi_{n-1}(y_{1:n-1}) - \phi_n(y_{1:n}) - \frac{1}{2n} \tag{3}$$

$$= \phi_{n-1}(y_{1:n-1}) - \frac{1}{2}\left(\phi(y_1, \ldots, y_{n-1}, +1) + \phi(y_1, \ldots, y_{n-1}, -1)\right) = 0. \tag{4}$$

We also remark that the choice of q_n equalizes the values of (3) for the possibilities $y_n = 1$ and $y_n = -1$. Such a strategy is called an *equalizer*. For intermediate steps, the algorithm takes on the form

$$q_t(y_1, \ldots, y_{t-1}) \triangleq n(\phi_t(y_1, \ldots, y_{t-1}, -1) - \phi_t(y_1, \ldots, y_{t-1}, +1)). \tag{5}$$

Continuing in this fashion from n backwards to $t = 1$ proves the claim. □

We may interpret ϕ as a potential function. Then the algorithm at time t is saying: The prediction should be biased towards that label whose potential (under a random evolution of future) is higher. What is especially interesting, we are justified (by the fact that our method is optimal) in using coin flips $\epsilon_{t+1}, \ldots, \epsilon_n$ for the future even though the sequence $y_1, \ldots, y_{t-1}$ will continue to evolve in reality in an arbitrary way on which we place no stochastic assumptions. This is a fortuitous consequence of our choice of an equalizer strategy.

A trivial corollary is:

Corollary 2. *For any $\phi : \{\pm 1\}^n \to \mathbb{R}$, satisfying (1) and $\mathbb{E}\phi \geq 1/2$, there exists a randomized algorithm $\mathcal{A}$ making (on average over its randomization) the number of mistakes no more than $\phi(y_1, \ldots, y_n)$, for any sequence:*

$$\forall (y_1, \ldots, y_n), \quad \ell(\mathcal{A}; y_1, \ldots, y_n) \leq \phi(y_1, \ldots, y_n).$$

Why is this simple result surprising? It tells us that existence of a prediction method can be verified by simply checking some probabilistic inequality ($\mathbb{E}\phi \geq 1/2$). In the next part of the course, we will spend some time learning several techniques for checking (slightly more complicated) expressions for certifying existence of prediction methods, and for finding them.

We now have a way of choosing ϕ that will work well for the problem at hand. For instance, if we are predicting class labels for nodes of a graph, we may tilt ϕ towards "smooth" labelings, expecting homogeneity with respect to the graph structure. Moreover, this is done is a completely deterministic fashion, without reliance on a probabilistic source for the data.

Later in the course we will cover more general techniques that do not arise from simply taking a ϕ function and defining an algorithm the way we did. But for now we can already try to construct good ϕ's for the problem of predicting sequences generated by humans. One of the first tools is a method for combining several good predictors into a meta-predictor. After that, we may define a collection of ϕ's that predict human-generated sequences according to some finite state machines, and create a meta-predictor that does as well as the best of them on the given sequence. (For those interested, the version of the mind-reader on the web seems to be using context weighted trees).

Performing as well as a given finite collection of predictors is often called the "experts setting." Assume that we have access to predictions of k experts $\mathcal{E}_1, \ldots, \mathcal{E}_k$, which produce on round t (in a non-anticipating manner) a number (called advice) $\mathcal{E}_j(y_1, \ldots, y_{t-1}) \in \{-1, 1\}$. For each $j \in [k]$, define

$$\phi^j(y_1, \ldots, y_n) = \frac{1}{n} \sum_{t=1}^{n} \mathbf{1}\{\mathcal{E}_j(y_1, \ldots, y_{t-1}) \neq y_t\},$$

the performance of expert j, which is only known at the end. We check that $\mathbb{E}\phi^j(\epsilon_1, \ldots, \epsilon_n) = 1/2$. Of course, we can do as well as a single expert j by following her advice, and also incurring ϕ^j number of mistakes. The question is whether it is possible to do (almost) as well as the best expert? The first attempt is to take the function

$$\min_{j \in [k]} \phi^j(y_1, \ldots, y_n)$$

as the overall quality function $\phi(y_1, \ldots, y_n)$. However, we observe that the expectation of this function under the uniform distribution is less than $1/2$ (prove it!). Luckily, we need only a small correction:

$$\phi(y_1, \ldots, y_n) = \min_{j \in [k]} \phi^j(y_1, \ldots, y_n) + \sqrt{\frac{c \log(k)}{n}}$$

Homework: find a value of c that ensures $\mathbb{E}\phi \geq 1/2$ for this function.

Now that we've certified $\mathbb{E}\phi \geq 1/2$, there must exist a method that makes the number of mistakes no more than the *best* of the experts, plus a vanishing term. What is the method? There are several. Of course, the one in (5) works. However, it requires certain averaging with random signs. There are more efficient methods, and a popular one is called the Exponential Weights (or, Multiplicative Weights). It can be derived in a principled manner with the tools we will introduce. Two caveats: the argument in the lemma requires us to be able to simulate the values of experts on some hypothetical future given by random

coin flips, and the experts cannot know future. As we will see, the Exponential Weights algorithm works even if these two requirements are removed.

We finish this lecture with a concrete example. Suppose you have a hunch that human-entered sequences tend to have imbalance of 1's and −1's. Can we exploit it? Here is what follows immediately from our previous discussion:

> There is a (simple) algorithm that will make at most
>
> $$\min\{\bar{p}, 1-\bar{p}\} + O(1/\sqrt{n})$$
>
> mistakes on any sequence with $\bar{p}$ proportion of 1's, and the method does not need to know this proportion.

For example, if the sequence happens to consist of 80% 1's, the algorithm makes roughly 20%, if n is large enough. This is not a trivial statement, since we do not assume that the sequence is i.i.d. The conclusion is: we can easily build a prediction method that will win over any unbalanced sequence entered by a human.

References

[Cov65] Thomas M Cover. Behaviour of sequential predictors of binary sequences. In *Proc. 4th Prague Conf. Inform. Theory, Statistical Decision Functions, Random Processes*, 1965.

[Hag56] DW Hagelbarger. Seer, a sequence extrapolating robot. *Electronic Computers, IRE Transactions on*, (1):1–7, 1956.

[Pou14] W. Poundstone. *How to Predict the Unpredictable: The Art of Outsmarting Almost Everyone*. 2014.

[Sha53] Claude E Shannon. A mind-reading machine. *Bell Laboratories memorandum*, 1953.