COS 511: Theoretical Machine Learning

Lecturer: Rob Schapire
Scribe: Anais Ta

Lecture # 21
April 23, 2017

1 Introduction

1.1 Recap from last time

Last time, we started looking at online log-loss, which is a very important loss function. The setting was the following one:

- $N = \#$ experts
- For $t = 1, \ldots, T$, expert i predicts $p_{t,i}$ (distribution on $\mathcal{X}$)
- The master combines all predictions into distribution q_t
- We observe $x_t \in \mathcal{X}$
- The loss is given by : Loss $= -\ln[q_t(x_t)]$

We want:

$$-\sum_{t=1}^{T} \ln[q_t(x_t)] \leq \min_i -\sum_{t=1}^{T} \ln[p_{t,i}(x_t)] + small$$

That is, we want the cumulative loss to be bounded by the cumulative loss of the best expert in hindsight, plus a small term.

1.2 Universal Compression

A very different motivation for online learning with log-loss comes from **coding theory**:

Suppose Alice wants to send Bob one message from a set $\mathcal{X}$. For instance, $\mathcal{X}$ might be the set of English letters : $\mathcal{X} = \{\text{“}a\text{”}, \text{“}b\text{”}, ...\}$. Let $p(x)$ be the probability of choosing x in $\mathcal{X}$. To send a message of that kind, we can use $-\lg[p(x)]$ bits.

Now suppose Alice wants to send an entire text or sequence of messages. Suppose so far she has sent “I am goin__”. If we were to ignore context, we'd predict an “e” since it is the most frequent letter in English, whereas with context, we will most probably predict a “g” . Thus the idea is to figure out a dynamic distribution at time t. That is, we want to estimate $p_t(x_t)$, the conditional probability of x_t given context x_1^{t-1}, i.e. all x from 1 to $t-1$ are known, and then encode x_t using $-\lg[p_t(x_t)]$ bits. In our setting, $p_t(x_t)$ is a conditional probability, therefore we need a way to estimate the conditional probabilities.

Suppose we already have coding methods (for instance one for French, one for German...). How can we code this message without knowing ahead of time which encoding is going to be the best? We are looking for an algorithm that will combine all coding methods in the best fashion. We want to prove the bound on the cumulative log-loss. Let N be the number of coding methods. The bound mentioned above can be interpreted as following:

$\sum_{t=1}^{T} -\lg(q_t(x_t))$ is the number of bits used

$\sum_{t=1}^{T} -\lg(p_{t,i}(x_t))$ is the number of bits used by the i^{th} coding method.

Thus, the bound tells us that the master coding method will be almost as good as the best of the given coding methods since it will be bounded by the best coding method plus a small term.

This kind of algorithm is called **universal compression**.

2 Framework

We will introduce the following notation so that the probability measures p and q look like conditional probabilities.

- $p_{t,i}(x_t) = p_i(x_t|x_1^{t-1})$
- $q_t(x_t) = q(x_t|x_1^{t-1})$

The notation x_1^t indicates the conditioning on all elements from 1 to t, where x_1^t is the vector $x_1^t = \langle x_1, ..., x_t \rangle$. This notation is meant to look like we are conditioning on everything that has happened so far. In reality, $p_i(x_t|x_1^{t-1})$ is a distribution that is given to us by "black box" predictors, and $q(x_t|x_1^{t-1})$ is the prediction of the algorithm that we are deriving.

The approach we are going to pursue is to pretend that the data is *random*, according to the process described below. However, the results we'll derive will hold for any sequence of data, even without the randomness assumption.

* STEP 01 * Choosing a specific expert

- One expert i^* is chosen uniformly at random, that is, i^* is a random variable.
- We introduce $Pr[i^* = i] = \frac{1}{N}$ which represents the probability of the choice of i^* according to the random process.

* STEP 02 * Generate a sequence

- Now that we have chosen i^*, we pretend that $(x_1, ...x_T)$ is a sequence generated by p_{i*}. That is, the t^{th} item is generated given the first $(t-1)$ outcomes, so

$$Pr[x_t|x_1^{t-1}, i^* = i] = p_{i^*}(x_t|x_1^{t-1})$$

- At present, we only need to define $q(x_t|x_1^{t-1})$ in order to define the algorithm. Specifically, we define

$$q(x_t|x_1^{t-1}) = Pr[x_t|x_1^{t-1}]$$

- Now let's compute this quantity:

$$\begin{aligned} q[(x_t|x_1^{t-1}] &= \sum_{i=1}^{N} Pr[i^* = i, x_t|x_1^{t-1}] && \text{Marginalizing over the experts} \\ &= \sum_{i=1}^{N} Pr[i^* = i|x_1^{t-1}]Pr[x_t|i^* = i, x_1^{t-1}] && \text{Using } P[a,b] = P[a]Pr[b|a] \\ &= \sum_{i=1}^{N} w_{t,i}Pr[x_t|i^* = i, x_1^{t-1}] && \text{By defining } w_{t,i} = Pr[i^* = i|x_1^{t-1}] \end{aligned}$$

  In order to specify q, we can work out how $w_{t,i}$ is updated from round to round by computing the update step by step starting at $t = 1$.

- Let us compute $w_{t,i}$:
  - At $t = 1$, $w_{1,i} = Pr[i^* = i] = \frac{1}{N}$
  - We can compute $w_{t+1,i}$ inductively as follows :

$$\begin{aligned} w_{t+1,i} &= Pr[i^* = i|x_1^t] \\ &= Pr[i^* = i|x_1^{t-1}, x_t] && \text{By breaking up } Pr[i^* = i|x_1^t] \\ &= \frac{Pr[i^* = i|x_1^{t-1}]Pr[x_t|i^* = i, x_1^{t-1}]}{Pr[x_t|x_1^{t-1}]} && Pr[a|b] = Pr[b|a]\frac{Pr[a]}{Pr[b]} \\ &\propto w_{t,i}p_i(x_t|x_1^{t-1}) && \text{By ignoring the normalization term} \end{aligned}$$

We find that, the weight update is proportional to the previous one times a conditional probability.

3 Bayes Algorithm

The result we proved is called Bayes Algorithm since it uses Bayes' Rule. Here is a recap:

- $N = \#$ experts
- $\forall i \in \{1, ..., N\}$, $w_{1,t} = \frac{1}{N}$
- for $t = 1, \ldots, T$
  - Expert i predicts $p_{t,i}$ distribution on $\mathcal{X}$
  - The master combines into distribution q_t
  - $q_t(x) = \sum_{i=1}^{N} w_{t,i}p_{t,i}(x)$
  - Observe $x_t \in \mathcal{X}$
  - Loss $= -\ln[q_t(x_t)]$
  - $\forall i \in \{1, ..., N\}$, $w_{t+1,i} = \frac{w_{t,i}p_{t,i}(x_t)}{Z}$ where Z is the normalization term

Our analysis is going to hold for any sequence $(x_1, x_2, ..., x_T)$. We can derive the same update using multiplicative weights. In the Randomized Weighted Majority Algorithm, the loss was a zero-one loss. We computed the update rule of the $w_{t,i}$'s showing that: if there was a mistake, the weight was multiplied by β. On the contrary, if there was no mistake, the weight was multiplied by 1, remaining unchanged. The update rule, where *loss* refers to the zero-one loss, was:

$$w_{t+1,i} = \frac{w_{t,i}\beta^{loss}}{Z}$$

In our case we have a log loss, so by choosing the right value of β we get back to an analogous multiplicative weight update:

$$\begin{aligned} \beta^{loss} &= \beta^{-\ln(p_{t,i}(x_t))} = p_{t,i}(x_t) \\ \beta &= e^{-1} \end{aligned}$$

4 Analysis

In this section we'll prove the theorem which gives us a bound on the loss of Bayes' algorithm.

* STEP 01 * Defining q

We are **defining** $q(x_1^T)$ for any sequence x_1^T, such that it can be written in the form below:

Definition 4.1 $q(x_1^T) = q(x_1, \cdots, x_T) = q(x_1)q(x_2|x_1)\cdots q(x_T|x_1, \cdots, x_{T-1})$

We can rewrite this as following:

$$\begin{aligned} q(x_1^T) &= \prod_{t=1}^{T} q(x_t|x_1^{t-1}) \\ &= \prod_{t=1}^{T} Pr[x_t|x_1^{t-1}] \\ &= Pr[x_1]Pr[x_2|x_1]\cdots Pr[x^T|x_1^{T-1}] \\ &= Pr[x_1^T] \end{aligned}$$

This definition implies that $q(x_1^T)$ is equal to the probability of the sequence x_1^T according to our pretend random process.

Similarly, we can define a probability p_i on the x_1^T sequences in an analogous way, and thus it also implies that p_i is equal to the probability of the sequence conditioned on i being the chosen expert i^*.

$$\begin{aligned} p_i(x_1^T) &= \prod_{t=1}^{T} p_i(x_t|x_1^{t-1}) \\ &= Pr[x_1^T|i^* = i] \end{aligned}$$

*** STEP 02 * Looking at cumulative loss**

$$\begin{aligned} -\sum_t \ln[q_t(x_t)] &= -\sum_{t=1}^{T} \ln\left[q(x_t|x_1^{t-1})\right] \\ &= -\ln\left[\prod_{t=1}^{T} q(x_t|x_1^{t-1})\right] \\ &= -\ln\left[q(x_1^T)\right] \end{aligned}$$

We are treating the sequence as if it was generating a probability measure. We can do the same with each expert i, $i \in 1, ..., N$

$$-\sum_t \ln\left[p_{t,i}(x_t)\right] = -\ln\left[p_i(x_1^T)\right]$$

*** STEP 03 * Marginalization**

$$\begin{aligned} q(x_1^T) &= Pr[x_1^T] \\ &= \sum_{i=1}^{N} \underbrace{Pr[i^* = i]}_{\text{uniform probability}} \quad Pr[x_1^T | i^* = i] \\ &= \sum_{i=1}^{N} \frac{1}{N} Pr[x_1^T | i^* = i] && \text{By replacing } Pr[i^* = i] \\ &= \sum_{i=1}^{N} \frac{1}{N} p_i(x_1^T) && \text{By definition of } p_i(x_1^T) \\ &= \frac{1}{N} \sum_{i=1}^{N} p_i(x_1^T) \end{aligned}$$

*** STEP 04 * Putting everything together**

Now we can express log-loss:

$\forall x_1, ..., x_T,$

$$
\begin{aligned}
-\sum_t \ln[q_t(x_t)] &= -\ln[q(x_1^T)] \\
&= -\ln\left[\frac{1}{N}\sum_i p_i(x_1^T)\right] \\
&\leq -\ln\left[\frac{1}{N}p_i(x_1^T)\right] && \text{Because for any } i \sum_j p_j(x_1^T) \geq p_i(x_1^T) \\
&= -\ln[p_i(x_1^T)] + \ln(N) && \text{By breaking up the product into a sum} \\
&= -\sum_t \ln[p_{t,i}(x_t)] + \ln(N) && \text{By using STEP 02}
\end{aligned}
$$

We have proved the following theorem:

Theorem 1

$$-\sum_t \ln[q_t(x_t)] \leq \min_i \left\{ -\sum_t \ln[p_{t,i}(x_t)] + \ln(N) \right\}$$

This shows that we can bound the regret of the algorithm and it is of magnitude $\ln N$.

We can go back to the the coding theory setting that we presented at the beginning of the lecture: we could imagine an alternative approach in which Alice first computes which coding method is best for the particular message she wants to send, then sends the index i of that coding method, and finally sends the message using coding method i. This would cost $\lg N$ bits to send the index i, plus the number of bits of the best coding method. That is exactly the same bound she would get if using Bayes algorithm, but with Bayes algorithm, she can be doing the coding on the fly, rather than first evaluating all coding methods on the entire message (which might be expensive or unavailable).

In doing our construction, we could have picked the i^{th} expert with probability π_i, where $\forall i, \pi_i \geq 0$ and $\sum_i \pi_i = 1$.

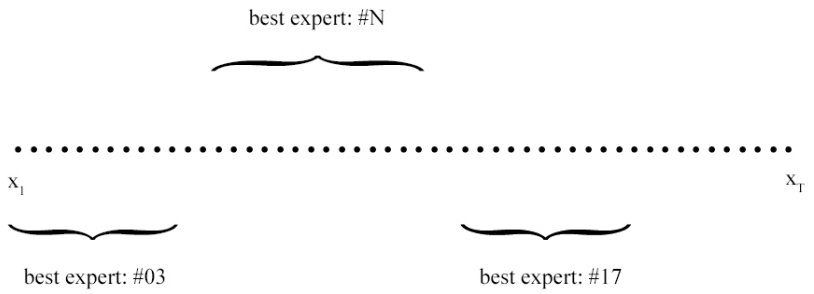
We would have gotten the following inequality:

$$-\sum_t \ln[q_t(x_t)] \leq \min_i \{ -\sum_t \ln p_{t,i}(x_t) - \ln(\pi_i) \}$$

That would be interesting if you had any information on which expert is going to perform the best. Indeed, modifying the probability with which we choose our expert is a way to minimize regret with respect to the best performing expert.

5 Switching Experts

Q. What happens if you have a sequence where one expert is good for a while then another one is good for another while ?



We want an online algorithm that is going to be able to switch from one expert to another. We are seeking an algorithm that will perform nearly as well as the best switching sequence of experts. We can relate this idea to that of coding theory, which we saw at the beginning of the class. If you want to encode a message that uses several languages, for instance, if a message contains both English and Spanish sentences, you want to use the English expert for the parts in English and the Spanish one for the parts in Spanish.

Ultimately, we want a bound that's competitive with the best switching sequence of experts. One approach is to create meta/super experts which imitate that behaviour (exact copies of an expert for a while) though leading to a very large family.

Next time, we'll talk about this gigantic challenge: how do you computationally deal with this?