COS 511: Theoretical Machine Learning

Lecturer: Rob Schapire
Scribe: Amy Tai

Lecture #21
April 22, 2014

1 Online Learning with Log-Loss

1.1 Problem

Recall the online learning problem from last time:

$i \in N$ experts
for $t = 1, \ldots, T$ **do**
  each expert i chooses distribution $p_{t,i}$ over X
  master combines into distribution q_t over X

  observe $x_t \in X$
end for

At the end of this learning procedure, we want the following bound on the log-loss of the master:

$$-\sum_{t=1}^{T} \ln q_t(x_t) \leq \min_i \left(-\sum_{t=1}^{T} \ln p_{t,i}(x_t) \right) + \epsilon \tag{1}$$

where ϵ is small, and the minimization term is interpreted as the cumulative loss of the best expert, in hindsight.

1.2 Universal Compression

Before we develop an algorithm that allows us to arrive at the desired bound in Equation 1, we take an aside to motivate online learning with log-loss.

Here is a problem from coding theory: Suppose we have a sender, Alice, and a receiver, Bob. Suppose Alice wants to send a message to Bob, and suppose the message is comprised of letters from alphabet X and $p(x)$ is the probability of choosing $x \in X$. Then for messages of length 1 letter, the optimal way of coding the message is $-\lg p(x)$ bits.

But the more interesting problem is extending this encoding to messages of arbitrary length, $x_1, x_2, \ldots$. We could just assume that the x_t are independently drawn from the same distribution p, in which case the optimal encoding simply has length $\sum_t -\lg p(x_t)$.

However, independence is a poor assumption. In English, for example, knowing that the message so far is "`I am goi`", basically tells us that the next letter is going to be `n`. However, using the typical distribution of letters over the English alphabet, the strawman approach would guess that `e` has the highest likelihood of being the next letter.

From this example, we learn that ideally, we want p_t to be the probability of the next letter, x_t, *given* the context $\equiv \langle x_1, \ldots, x_{t-1} \rangle$. Denote this context as x_1^{t-1}. If we have such p_t, then the length of the optimal encoding will be $\sum_t -\lg p_t(x_t)$ bits.

However, it is nearly impossible to learn p_t, because of the nearly limitless number of contexts. Instead, we consider a learning algorithm that takes a bunch of candidate methods

for encoding the messages, and combine them into a master method. For example, a sample candidate method would only look 1 letter back, and have contexts of length 1 letter.

Formally, suppose the ith method does the following: Given x_1^{t-1}, encodes x_t using $-\lg p_{t,i}(x_t)$ bits, where $p_{t,i}$ is method i's estimated distribution of x_t, given x_1^{t-1}.

Then we combine these into the master method, which encodes x_t with $-\lg q_t(x_t)$ bits. Then the master uses a total of $\sum_{t=1}^{T} -\lg q_t(x_t)$ bits to encode the entire message. We want:

$$\sum_{t=1}^{T} -\lg q_t(x_t) \leq \min_i \sum_{t=1}^{T} -\lg p_{t,i}(x_t) + \epsilon$$

Namely, we want the total number of bits used by the master to be at most the number of bits used by the best method, plus some small ϵ. This is exactly what the online learning model that we presented in Section 1.1 does.

2 Bayes Algorithm

Now we go back to describing an algorithm for arriving at Equation 1. To derive such an algorithm, we *pretend* that the data is random. However, we will see in the next section that the proof of the algorithm's bounds holds for any arbitrary data sequence and does not depend on the randomness assumption. Then suppose $x_1, \ldots, x_T$ is random and generated as follows:

- Expert i^* chosen at random (assume uniformly across all experts). So $Pr[i^* = i] = \frac{1}{N}$
- Generate x_t according to distribution p_{t,i^*}. Then $Pr[x_t | x_1^{t-1}, i^* = i] = p_i(x_t | x_1^{t-1})$

We also denote $p_{t,i}(x_t) = p_i(x_t | x_1^{t-1})$ and $q_t(x_t) = q(x_t | x_1^{t-1})$. Furthermore, by definition, $q(x_t | x_1^{t-1}) = Pr[x_t | x_1^{t-1}]$. Then we have:

$$\begin{aligned}
q(x_t | x_1^{t-1}) &= Pr[x_t | x_1^{t-1}] && \text{By definition} \\
&= \sum_i Pr[x_t, i^* = i | x_1^{t-1}] && \text{Marginalizing over all experts} \\
&= \sum_i Pr[i^* = i | x_1^{t-1}] \cdot Pr[x_t | i^* = i, x_1^{t-1}] && \text{Product rule} \\
&= \sum_i w_{t,i} \cdot p_i(x_t | x_1^{t-1})
\end{aligned}$$

where we denote $w_{t,i} = Pr[i^* = i | x_1^{t-1}]$. Then we seek to specify how $w_{t,i}$ is updated from round to round. First, we observe that $w_{1,i} = Pr[i^* = i | \emptyset] = Pr[i^* = i] = \frac{1}{N}$.

Now suppose that we have $w_{t,i}$ and want $w_{t+1,i}$. Then:

$$\begin{aligned}
w_{t+1,i} &= Pr[i^* = i | x_1^t] \\
&= Pr[i^* = i | x_t, x_1^{t-1}] \\
&= \frac{Pr[i^* = i | x_1^{t-1}] \cdot Pr[x_t | i^* = i, x_1^{t-1}]}{Pr[x_t | x_1^{t-1}]} && \text{Bayes' Rule} \\
&= \frac{w_{t,i} \cdot p_i(x_t | x_1^{t-1})}{\text{norm}} && \text{Consider } Pr[x_t | x_1^{t-1}] \text{ as a normalization}
\end{aligned}$$

Because $Pr[x_t|x_1^{t-1}]$ is independent of i, we can consider it as a normalization factor. Then we observe that we have a simple update algorithm for $w_{t,i}$. We then modify and fill in the learning problem from Section 1.1 with the following algorithm:

Bayes' Algorithm
$i \in N$ experts
Initialize $w_{1,i} = \frac{1}{N}$
for $t = 1, \ldots, T$ **do**
each expert i chooses distribution $p_{t,i}$ over X
$q_t(x) = \sum_i w_{t,i} p_{t,i}(x)$
Update $w_{t+1,i} = \frac{w_{t,i} \cdot p_{t,i}(x_t)}{\text{norm}}$
observe $x_t \in X$
end for

We also observe that this algorithm is very similar to the weight-update online learning algorithms that we saw earlier, in which $w_{t+1,i} = \frac{w_{t,i} \cdot \beta^{\text{loss}}}{Z_t}$. In this case, loss is the log-loss, or $-\ln p_{t,i}(x_t)$. If we let $\beta = e^{-1}$, then $\beta^{\text{log-loss}}$ is precisely the update factor in Bayes' algorithm.

Now we prove that this algorithm works on any given data $x_1, \ldots, x_T$, and achieves the log-loss bound presented in Equation 1.

3 Bounding Results of Bayes' Algorithm

First, we extend the definition of q so it is defined for entire sequences $x_1, \ldots, x_T$.

Define:

$$\begin{aligned} q(x_1^T) &= q(x_1) \cdot q(x_2|x_1) \cdot q(x_3|x_2, x_1) \ldots \\ &= \prod_t q(x_t|x_1^{t-1}) \\ &= \prod_t Pr[x_t|x_1^{t-1}] \end{aligned}$$

Similarly, define:

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

Then:

$$\begin{aligned} -\sum_t \ln q_t(x_t) &= -\sum_t \ln q(x_t|x_1^{t-1}) \\ &= -\ln \prod_t q(x_t|x_1^{t-1}) \\ &= -\ln q(x_1^T) \end{aligned}$$

Similarly, $-\sum_t \ln p_{t,i}(x_t) = -\ln p_i(x_1^T)$. Then:

$$\begin{aligned} q(x_1^T) = Pr[x_1^T] &= \sum_i Pr[i^* = i] \cdot Pr[x_1^T | x^* = i] \\ &= \frac{1}{N} \sum_i p_i(x_1^T) \\ \Rightarrow \text{Log-loss of } q = -\ln q(x_1^T) &= -\ln(\frac{1}{N} \sum_i p_i(x_1^T)) \\ &\leq -\ln \frac{1}{N} p_i(x_1^T) \qquad\qquad \sum_i p_i(x_1^T) \geq p_i(x_1^T) \end{aligned}$$

Because this last inequality is true for all i, it must be true that the Log-loss of $q \leq \min_i(-\ln p_i(x_1^T)) + \ln N = \min_i(-\sum_t \ln p_{t,i}(x_t)) + \ln N$, which is exactly as we desired in Equation 1.

We note here that Alice could have used an offline algorithm to determine the best encoding of a message. Namely, Alice runs all i methods on her message, determines the encoded message with optimal length, which would be $\min_i(-\sum_t \lg p_{t,i}(x_t))$ bits, and sends this across to Bob, along with a specification of the method she used for the encoding. The encoding of the message takes $\lg N$ bits, if there are N methods. Hence, using an offline method, Alice achieves the same bounds on the length of her optimal message. However, using the online method, Alice can encode the message as it comes in a stream, and does not have to store as much data for her calculations.

We also note that we could have used a different prior probability other than the uniform distribution. We could use $Pr[i^* = i] = \pi_i$, instead of $\frac{1}{N}$. Then the only modification we need to make to the original Bayes' algorithm is the initialization of the weights, which we change to $w_{1,i} = \pi_i$. Then the bound becomes

$$-\sum_{t=1}^{T} \ln q_t(x_t) \leq \min_i \left(-\sum_{t=1}^{T} \ln p_{t,i}(x_t) - \ln \pi_i \right)$$

4 An Example

Suppose $X = \{0, 1\}$. Expert p predicts 0 with probability $1 - p$ and 1 with probability p. We have an expert for all $p \in [0, 1]$, hence we have infinite experts.

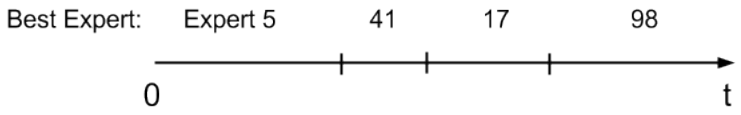
Hence $w_{t,p} = \frac{1}{\text{norm}} \prod_{t=1}^{t-1} \begin{cases} p & \text{if } x_t = 1 \\ 1-p & \text{otherwise} \end{cases}$. Suppose there are h 1's so far. Then $w_{t,p} = \frac{p^h(1-p)^{t-1-h}}{\text{norm}}$.

Then $q_t = \sum_p w_{t,p} \cdot p = \int_0^1 w_{t,p} \cdot p dp = \frac{h+1}{(t-1)+2}$. This is known as Laplace smoothing.

But in this case, $N = \infty$, so the $\ln N$ from Bayes' algorithm is not helpful. In a later lecture, we will get general bounds that can be applied to this particular case.

5 Switching Experts

Next time, we will cover a new learning model. Hitherto, we have assumed that there is one single, best expert. But this is not always a reasonable assumption. Instead, this new model assumes that the best expert switches based on data that the model sees, and this may change as time goes on. For example in the following diagram, the best expert changes during each epoch:



We will also assume that the number of switches is bounded.