COS 511: Theoretical Machine Learning

Lecturer: Rob Schapire — Lecture #21
Scribe: Lawrence Diao — April 23, 2013

1 On-Line Log Loss

To recap the end of the last lecture, we have the following on-line problem with N experts. For each round $t = 1, ..., T$:

- each expert i predicts $p_{t,i}$ distribution on $\mathbb{X}$
- master predicts q_t distribution on $\mathbb{X}$
- observe $x_t \in \mathbb{X}$
- loss $= -\ln q_t(x_t)$

We want to get a bound on the total loss of the master q_t in comparison to the best expert.

$$\sum_{t=1}^{T} -\log q_t(x_t) \leq \min_i \sum_{t=1}^{T} -\log p_{t,i}(x_t) + \text{small} \tag{1}$$

where here we use the general log function of arbitrary base. We'll see that this on-line log loss setting manifests itself in many applications such as horse racing and coding theory.

2 Coding Theory

Here, we are concerned with how to efficiently send a message from Alice to Bob in as few bits as possible. In this setting we define $\mathbb{X}$ as the "alphabet", and each $x \in \mathbb{X}$ as a letter. Say Alice wants to send one letter x. Define $p(x)$ to be the probability of sending x, which you can estimate from a corpus. The best you can do is to take $-\lg p(x)$ bits to send x.

Now Alice is trying to send a sequence of letters $x_1, x_2, x_3, ...$ One way we can do this is to use $p(x)$ for each letter separately, but this is sub-optimal for English. For example, if we see the following string of characters "I am goi__", we can easily predict the next letter to be 'n' given the context, but if we simply use $p(x)$, then we might say that 'e' is the most likely, since it is the letter of highest frequency in the English language. Our goal is to use the context to use fewer bits to encode x.

If we define $p_t(x_t)$ to be the probability distribution of x_t given context $x_1^{t-1} = \langle x_1, ..., x_{t-1} \rangle$, then it takes $-\lg p_t(x_t)$ to encode the extra letter x_t. However, it is really hard to model this probability. You can't get it just by counting as we could with $p(x)$. Instead, we consider combining a collection of coding methods where we don't know which one will be best.

Let's say we have N coding methods (N experts). We try to pick a master coding method that uses at most a small amount more bits than the best encoding method.

Let $p_{t,i}(x_t)$ = probability of x_t given x_1^{t-1} according to the i-th coding method. So we have

$$\sum_{t=1}^{T} -\lg p_{t,i}(x_t) \leftarrow \text{ bits used by } i\text{-th coding method}$$
$$\sum_{t=1}^{T} -\lg q_t(x_t) \leftarrow \text{ bits used by arbitrary coding method } q_t$$

We are trying to come up with a coding method $q_t(x_t)$ to guarantee

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

Such an algorithm is called a "universal compression" algorithm, since it works about as well as the best coding method for any input. Note that the bound should hold for any sequence of x_t's, so there's no assumption on randomness of x_t. Also note that this bound is of the form of (1).

3 "Universal Compression" Algorithm

In this section we try to determine the algorithm for choosing the master coding method. To make the math cleaner, we change the base back to e, and try to achieve the following bound

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

We also make the following notation changes

$$q_t(x_t) \rightarrow q(x_t \mid x_1^{t-1})$$
$$p_{t,i}(x_t) \rightarrow p_i(x_t \mid x_1^{t-1})$$

Let's pretend that x_t are random even though they're not in order to motivate an algorithm for picking q. Pretend that x_t are picked as follows:

– select one expert i^* with $\Pr[i^* = i] = \frac{1}{N}$

– $x_1, x_2, ...,$ generated according to i^*:

$$\Pr[x_1 \mid i^* = i] = p_i(x_1)$$
$$\Pr[x_2 \mid x_1,\ i^* = i] = p_i(x_2 \mid x_1)$$
$$...$$
$$\Pr[x_t \mid x_1^{t-1},\ i^* = i] = p_i(x_t \mid x_1^{t-1})$$

Then the most natural way to pick q is:

$$\begin{aligned}
q(x_t \mid x_1^{t-1}) &= \Pr[x_t \mid x_1^{t-1}] \\
&= \sum_i \Pr[x_t,\ i^* = i \mid x_1^{t-1}] && \text{marginalize} \\
&= \sum_i \Pr[i^* = i \mid x_1^{t-1}] \cdot \Pr[x_t \mid i^* = i,\ x_1^{t-1}] && \text{conditional probability} \\
&= \sum_i w_{t,i} \cdot p_i[x_t \mid x_1^{t-1}] && w_{t,i} = \Pr[i^* = i \mid x_1^{t-1}]
\end{aligned}$$

If we can find these $w_{t,i}$ then we have an algorithm.

$$\begin{aligned}
w_{1,i} &= \Pr[i^* = i] = \frac{1}{N} && \text{intialization} \\
w_{t+1,i} &= \Pr[i^* = i \mid x_1^t] \\
&= \Pr[i^* = i \mid x_1^{t-1},\ x_t] \\
&= \frac{\Pr[i^* = i \mid x_1^{t-1}] \cdot \Pr[x_t \mid i^* = i,\ x_1^{t-1}]}{\Pr[x_t \mid x_1^{t-1}]} && \text{bayes rule} \\
&= \frac{w_{t,i} \cdot p_i(x_t \mid x_1^{t-1})}{\text{Normalization}}
\end{aligned}$$

So we are left with the following algorithm.

$\forall i:\ w_{1,i} = \frac{1}{N}$

On round t:

Choose $q(x_t \mid x_1^{t-1}) = \sum_i w_{t,i} p_i(x_t \mid x_1^{t-1})$

Update Weights: $\forall i:\ w_{t+1,i} = \dfrac{w_{t,i} p_i(x_t \mid x_1^{t-1})}{\text{Normalization}}$

We can see that this weight update is very similar to other weight-update online learning algorithms we have seen in the past, except we don't have to tune β since there is only one "correct" choice of $\beta = e^{-1}$ in this case.

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

4 Bounding the Log Loss

Here we are trying to prove (1), given our choice of $q(x_t \mid x_1^{t-1}) = \Pr[x_t \mid x_1^{t-1}]$
Theorem:

$$\sum_{t=1}^{T} -\log q_t(x_t) \le \min_i \sum_{t=1}^{T} -\log p_{t,i}(x_t) + \log N$$

$$\begin{aligned}
\text{Define } q(x_1^T) &= q(x_1)q(x_2 \mid x_1)q(x_3 \mid x_1,\ x_2)... \\
&= \prod_{t=1}^{T} q(x_t \mid x_1^{t-1}) \\
&= \prod_{t=1}^{T} \Pr[x_t \mid x_1^{t-1}] \\
&= \Pr[x_1^T] && \text{chain rule}
\end{aligned}$$

In the same way we can do this with each expert

$$p_i(x_1^T) = \Pr[x_i^T \mid i^* = i]$$

Additionally, the total loss of our algorithm is given by the following:

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

Similarly, for any expert,

$$-\sum_{t=1}^{T} \log p_{t,i}(x_t) = -\log p_i(x_1^T)$$

So we have the following bound:

$$\begin{aligned}
q(x_1^T) &= \Pr[x_1^T] \\
&= \sum_i \Pr[i^* = i] \cdot \Pr[x_i^T \mid i^* = i] && \text{marginalize} \\
&= \frac{1}{N} \sum_i p_i(x_1^T) \\
&\geq \frac{1}{N} p_i(x_1^T) && \forall i \\
\implies -\log q(x_1^T) &\leq -\log p_i(x_1^T) + \log N && \forall i \\
\implies \sum_{t=1}^{T} -\log q_t(x_t) &\leq \min_i \sum_{t=1}^{T} -\log p_{t,i}(x_t) + \log N
\end{aligned}$$

Here we consider $\log N$ to be "small". Note that this bound does not assume any randomness for x_t. Now, let's consider an alternative encoding scheme, where Alice waits for the entire message $x_1, x_2, ..., x_T$, chooses the best out of the N candidate encoding methods, uses $\lg N$ bits to encode which encoding method she used, and finally sends her message according to this chosen method. We can see that this scheme would use just as many bits as the right hand side of the bound, but using our online algorithm we don't have to wait for the whole message to start encoding/sending. We won't go into detail about decoding, but in order to decode, Bob effectively just simulates what Alice does to encode, so decoding is just as efficient as Alice's encoding, making algorithmic efficiency a non-factor.

5 Variations

5.1 Using a prior

In this section we consider a prior $\Pr[i^* = i] = \pi_i$ not necessarily uniform. Everything about our algorithm stays the same except the initial weights are now $w_{1,i} = \pi_i$, and the final bound ends up being

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

5.2 Infinite Experts

Consider the problem where $\mathbb{X} = \{0, 1\}$, and expert p predicts $x_t = \begin{cases} 1 & \text{with prob } p \\ 0 & \text{with prob } 1-p \end{cases}$ where we have all experts $p \in [0, 1]$. We need to figure out the weights $w_{t,p}$ to get q. In the finite case, we had $w_{t,i} = Pr[i^* = i \mid x_1^{t-1}]$, but applying this definition to the infinite case doesn't really make sense unless we're talking about the probability density:

$$\begin{aligned}
\Pr[p^* \in dp \mid x_1^{t-1}] &= \frac{\Pr[x_1^{t-1} \mid p^* \in dp] \cdot Pr[p^* \in dp]}{\Pr[x_1^{t-1}]} && \text{bayes rule} \\
&= \frac{\Pr[x_1^{t-1} \mid p^* \in dp] \cdot \Pr[p^* \in dp]}{\text{Normalization}} \\
&= \frac{\Pr[x_1^{t-1} \mid p^* \in dp]}{\text{Normalization}} && \text{assuming } \Pr[p^* \in dp] \text{ uniform} \\
&\propto p^h (1-p)^{t-h-1}
\end{aligned}$$

where h is the number of heads (1's) in the first $t-1$ rounds. Now, letting

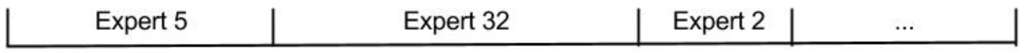
$$w_{t,p} = p^h (1-p)^{t-h-1}$$

$$q_t = \frac{\int_0^1 w_{t,p} p dp}{\int_0^1 w_{t,p} dp \leftarrow \text{Normalization}} = \frac{h+1}{(t-1)+2} \leftarrow \text{sometimes called laplace smoothing}$$

We can get a similar bound as before in this case but $\log \pi_i$ or $\lg N$ doesn't make sense. We'll see a bound in a future lecture.

6 Switching Experts

In this section we set up the problem for next class. Here, we no longer assume that one expert is good all the time. Instead, we change the model so that at any step, the "correct" expert can switch to another expert.



However, the learning algorithm has no idea when the experts are switching. Our goal is to design an algorithm that performs well with respect to the best "switching" sequence of experts. We'll look at this in the next lecture.