COS 511: Foundations of Machine Learning

Rob Schapire
Scribe: Monirul I Sharif

Lecture #18
April 10, 2003

1 Review From Last Time

Last time, we were talking about how to model distributions, and we had this setup:
Given

- examples $x_1, \ldots, x_m \in X$ i.i.d from $\mathcal{D}$ which we are trying to model and $|X| = N$
- features $f_1, \ldots, f_n$, where $f_j : X \to \mathcal{R}$
- empirical average of features $\widehat{E} f_j$ where,

$$\widehat{E} f_j = \frac{1}{m} \sum_{i=1}^{m} f_j(x_i)$$

- and the two sets $\mathcal{P}$ and $\mathcal{Q}$ defined as:

$$\mathcal{P} = \left\{ p : E_p f_j = \widehat{E} f_j \quad \forall j \right\}$$

$$\mathcal{Q} = \left\{ q \text{ of the form } q(x) = \frac{\exp(\sum_j \lambda_j f_j(x)}{Z_\lambda} \right\}$$

We also introduced a theorem which is the duality theorem

Theorem 1 *The following are equivalent:*

1. $q^* = \arg\max_{p \in \mathcal{P}} H(p)$
2. $q^* = \arg\max_{q \in \overline{\mathcal{Q}}} \frac{1}{m} \sum_i \ln q(x_i)$
3. $q^* \in \mathcal{P} \cap \overline{\mathcal{Q}}$.

Moreover, q^* is uniquely defined by any one of these conditions.

Today we talk about how to find this q^*. We will describe a nice simple algorithm called the Iterative Scaling algorithm, that can be used to find q^*.

2 Iterative Scaling Algorithm

2.1 Derivation

We are looking for a sequence of these λ's that lead to the λ's that minimize the negative log likelihood, i.e. maximize the likelihood.

Let us introduce some new notations. Let g_λ be the linear combination of features for a particular λ, and let q_λ be the associated distribution, where:

$g_\lambda(x) = \sum_j \lambda_j f_j(x)$ and $q_\lambda(x) = e^{g_\lambda(x)} / Z_\lambda$

Now, log loss is $L(\lambda)$ where,

$$L(\lambda) = -\frac{1}{m}\sum_i \ln q_\lambda(x_i) \tag{1}$$

We are looking for a sequence of vectors $\vec{\lambda}_1$, $\vec{\lambda}_2$, ... and we want L to be going down for each subsequent vector.

So in a single time step say from t to $t+1$ the difference can be defined as below, and we want to upper bound this.

$$\Delta L = L(\vec{\lambda}_{t+1}) - L(\vec{\lambda}_t) \tag{2}$$

Let $\vec{\lambda}' = \vec{\lambda}_{t+1}$ and $\vec{\lambda} = \vec{\lambda}_t$. Also let α_j be such that $\lambda'_j = \lambda_j + \alpha_j$. From (2) we get

$$\begin{aligned}
\Delta L &= L(\vec{\lambda}_{t+1}) - L(\vec{\lambda}_t) \\
&= -\frac{1}{m}\sum_i \left[\ln \frac{e^{g_{\lambda'}(x_i)}}{Z_{\lambda'}} - \ln \frac{e^{g_\lambda(x_i)}}{Z_\lambda}\right] \\
&= \frac{1}{m}\sum_i \left[g_\lambda(x_i) - g_{\lambda'}(x_i)\right] - \ln Z_\lambda + \ln Z_{\lambda'} \qquad (3)
\end{aligned}$$

Now the first term of (3)

$$\begin{aligned}
&= -\frac{1}{m}\sum_i \sum_j \alpha_j f_j(x_i) \\
&= -\sum_j \frac{\alpha_j}{m}\sum_j f_j(x_i) \\
&= -\sum_j \alpha_j \widehat{E} f_j \qquad (4)
\end{aligned}$$

Now,

$$\begin{aligned}
\frac{Z_{\lambda'}}{Z_\lambda} &= \frac{\sum_x \exp\left(\sum_j \lambda'_j f_j(x)\right)}{Z_\lambda} \\
&= \frac{\sum_x \exp\left(\sum_j \lambda_j f_j(x)\right)\exp\left(\sum_j \alpha_j f_j(x)\right)}{Z_\lambda} \quad \text{(plugging in the value of } \lambda'_j\text{)} \\
&= \sum_x q_\lambda(x)\exp\left(\sum_j \alpha_j f_j(x)\right) \\
&\le \sum_x q_\lambda(x)\sum_j f_j(x)e^{\alpha_j} \quad \text{(by convexity)} \\
&= \sum_j e^{\alpha_j}\sum_x q_\lambda(x) f_j(x) \\
&= \sum_j e^{\alpha_j} E_{q_\lambda} f_j \qquad (5)
\end{aligned}$$

Plugging (4) and (5) into (3) we can shown that,

$$\boxed{\Delta L \leq -\sum_j \alpha_j \widehat{E} f_j + \ln\left(\sum_j e^{\alpha_j} E_{q_\lambda} f_j\right)}$$

We have derived an upper bound of the change in the loss function. Now, to optimize, we can take the derivative to choose the α_j that is the smallest.

$$\frac{\partial}{\partial \alpha_j} = -\widehat{E}_j + \frac{E_j e^{\alpha_j}}{\sum_j E_j e^{\alpha_j}} = 0 \tag{6}$$

where $\widehat{E}_j = \widehat{E} f_j$ and $E_j = E_{q_\lambda} f_j$. The thing to notice is that if we have any solution for this, say α'_j we can add a constant c and get another solution, $\alpha_j = \alpha'_j + c$. The reason is that the constants get cancelled. We will choose this constant in such a way that the denominator of the second term equals 1.

By using that trick we can set

$$\boxed{\alpha_j = \ln \frac{\widehat{E}_j}{E_j}}$$

2.2 The Algorithm

The algorithm works iteratively by calculating successive $\vec{\lambda}_t$'s in each round

for $t = 1, 2, \ldots$

$$\lambda_{t+1,j} = \lambda_{t,j} + \ln \frac{\widehat{E} f_j}{E_{q_{\lambda_t}} f_j}$$

The above can be written using probability distributions.

$$p_{t+1}(x) = \frac{1}{Z} \times p_t(x) \prod_j \left(\frac{\widehat{E} f_j}{E_{p_t} f_j}\right)^{f_j(x)}$$

We are trying to find a distribution where it has this form, and converges to q^*. Let us try to understand this intuitively. Assume that $\widehat{E} f_j > E_{p_t} f_j$ (actually, we want them to be equal). The ratio inside the product above will be greater than 1, and therefore the distribution will concentrate more on the points with higher feature values, and therefore increase the expected value of the distribution, bringing it closer to the empirical average. Let us prove this.

2.3 Proving That the Algorithm Works

Theorem 2 *$p_t \to q^*$ as $t \to \infty$*

To prove the above, we are going to use something called an *auxiliary functions*.
Auxiliary function: fn A : prob distributions $\to R$, and satisfies the following properties:
(1) continuous

(2) $L(\vec{\lambda}_{t+1}) - L(\vec{\lambda}_t) \leq A(p_t) \leq 0$ we are bounding the change in the loss
(3) $A(p) = 0 \Longrightarrow \widehat{E} f_j = E_p[f_j] \quad \forall j \quad (p \in \mathcal{P})$

Proof: The idea is we will first show that if an auxiliary function A exists then p_t converges to q^* and then we show that there does exist an auxiliary function.

Our First claim is $A(P_t) \to 0$ if there does exist an auxiliary function. As A cannot become positive, L is always decreasing. And we also know that L is always nonnegative from its definition in (1). Therefore their differences has to converge to 0. So, A is being squeezed and will also converge to 0.

Now let's say

$$p = \lim_{x \to \infty} p_t$$

Therefore, $p \in \overline{\mathcal{Q}}$
Now,

$$A(p) = A\left(\lim_{t\to\infty} p_t\right) = \underbrace{\lim_{t\to\infty} A(p_t)}_{\text{by continuity of } A} = 0 \Rightarrow \underbrace{p \in \mathcal{P}}_{\text{by condition (3)}}$$

So, $p \in \mathcal{P} \cap \overline{\mathcal{Q}}$ which, by the duality theorem, implies that $p = q^*$.
Now we plug in our choice of α_j and can get the following:

$$\begin{aligned} \Delta L &\leq -\sum_j \widehat{E}_j \ln \frac{\widehat{E}_j}{E_j} \\ &= \underbrace{-\text{RE}\left(\widehat{E}_j \,||\, E_j\right)}_{A(p_t)} \end{aligned}$$

$A(p) = -\text{RE}\left(\widehat{E} f_j \,||\, E_p[f_j]\right)$

We have found our auxiliary function because our required properties are met. First of all, relative entropy is continuous. Second of all, it is nonnegative, and therefore its negative is nonpositive. And finally, the last property, what happens if $A(p) = 0$? That means relative entropy has to be equal to 0, and that can only happen when the two distributions are identical. That is exactly what the third required property of our auxiliary function is.

3 Log Loss in Online Learning

Now let us change gears. We were talking about log loss. We will now look at it in a simpler setting - the online learning model. It has applications in information theory and portofolio management. Applications such as data compression are also connected to log loss, especially in the online learning model.

Now let us see how log loss comes in the on-line learning model i.e. learning with expert advice. We have a bunch of experts. Before the experts were predicting classifications, but now as we are trying to learn a distribution, each expert predicts a distribution. The master algorithm combines these predictions and makes a combined prediction. When an example is seen, everybody suffers a loss, that is the log loss of that example that was observed.

So the model can be setup in the following way:

for $t = 1, 2, \ldots, T$:
- each expert i is predicting dist $p_{t,i}$ on X
- master algorithm predicts q_t (also a prob. dist.)
- observe $x_t \in X$ (arbitrary)
- master suffers loss $= -\ln q_t(x_t)$
- expert i suffers loss $= -\ln p_{t,i}(x_t)$

So, the algorithm observes x_1, x_2... x_{t-1}, and repeatedly tries to predict the next outcome x_t. For example, consider the case that you are listening to someone talking and you want to predict the next word that he says. You cannot predict the word exactly, but what you can do is you can predict a probability distribution of the next word that he might say i.e. how likely is it that each word will follow the last word.

It must be mentioned that we are not going to suppose that we have a perfect expert. We are also not assuming that the x_t's are random, they are arbitrary.

We want to prove that the total loss or the cumulative loss of the learning algorithm is not much worse than the loss of the best expert.

$$\overbrace{-\sum_{t=1}^{T} \ln q_t(x_t)}^{\text{cumulative loss}} \quad \leq \quad \overbrace{\min_i \left(-\sum_t \ln p_{t,i}(x_t) \right)}^{\text{cumulative loss of best expert}} + \quad \text{small amount} \tag{7}$$

Here,

$$\begin{aligned} p_{t,i}(x_t) &= p_i(x_t \mid x_1^{t-1}) \\ q_t(x_t) &= q(x_t \mid x_1^{t-1}) \end{aligned}$$

where, x_1^{t-1} means $x_1, \ldots, x_{t-1}$

Let us talk a little bit about compression. Suppose we want to encode or compress messages that are coming from X. Think of X as an alphabet. You have at your disposal, a set of compression algorithms. Naturally, we want to pick the one which gives the best compression, but we want to do that online. The set of approaches that we are working towards is called "Universal Compression" in information theory.

3.1 Algorithm for solving this model

Let us talk about an algorithm for solving this model. Here is the motivation behind the algorithm:

We are going to pretend:

- expert i^* chosen uniformly random
  $\Pr[i^* = i] = 1/N$ (this is called a prior)
- sequence x_1, x_2,... is generated according to distribution of that expert i.e. p_{i^*}
  $\Pr[x_t \mid x_1^{t-1}, i^* = i] = p_i(x_t \mid x_1^{t-1})$

We will compute

$$q(x_t \mid x_1^{t-1}) = \Pr[x_t \mid x_1^{t-1}] \tag{8}$$

For computing the above we just use Bayes' rule.

$$\begin{aligned}\Pr[x_t \mid x_1^{t-1}] &= \sum_i \Pr\left[x_t \wedge i^* = i \mid x_1^{t-1}\right] \\ &= \sum_i \underbrace{\Pr\left[i^* = i \mid x_1^{t-1}\right]}_{\text{called the ``posterior''}} . \underbrace{\Pr\left[x_t \mid x_1^{t-1}, i^* = i\right]}_{p_i(x_t \mid x_1^{t-1})} \end{aligned} \tag{9}$$

From Bayes' rule we know,

$$\Pr[B|A] = \frac{\Pr[A|B]\Pr[B]}{\Pr[A]}$$

Now we use Bayes' rule for the first factor in (9)

$$\Pr\left[i^* = i \mid x_1^{t-1}\right] = \frac{\Pr\left[x_1^{t-1} \mid i^* = i\right] . \overbrace{\Pr[i^* = i]}^{1/N}}{\Pr\left[x_1^{t-1}\right]} \tag{10}$$

Now,

$$\begin{aligned}\Pr\left[x_1^{t-1} \mid i^* = i\right] &= \Pr[x_1|i].\Pr[x_2|x_1, i]. \quad \ldots \quad .\Pr[x_{t-1}|x_1^{t-2}, i] \\ &= p_i(x_1).p_i(x_2|x_1). \quad \ldots \quad .p_i(x_{t-1}|x_1^{t-2}) \\ &= \prod_{t'=1}^{t-1} p_i(x_{t'}|x_1^{t'-1}) \\ &= w_{t,i} \qquad \text{(assume)} \end{aligned} \tag{11}$$

Final simplified computation

Compute:

By using (9),(10) and (11) in (8) we get

$$q\left(x_t|x_1^{t-1}\right) = \frac{\overbrace{\sum_i w_{t,i}\, p_i(x_t|x_1^{t-1})}^{\text{the weighted average of the predictions of experts}}}{\underbrace{\sum_i w_{t,i}}_{\text{normalization}}}$$

Update:

$$w_{t+1,i} = w_{t,i} \,.\, p_i(x_t|x_1^{t-1})$$

3.2 Comparison with previous algorithms

We have seen algorithms like this before. For example in the weighted majority algorithm the computation step also calculated a weighted average. The update function is almost similar. In the update rule we had β^{loss}, where loss was 1 if we made mistake and 0 if not. In this case we can think of that term being

$$\beta^{-\ln p_i(x_t|x_1^{t-1})}$$

If we choose $\beta = e^{-1}$ then this term becomes $p_i(x_t|x_1^{t-1})$. In other words, we can think of this as not having to compute β, and that it has a natural value which is e^{-1}.

In the next lecture we will analyze this algorithm.