10-704: Information Processing and Learning **Fall 2016**

Lecture 13: Oct 12

Lecturer: Aarti Singh

Note: *These notes are based on scribed notes from Spring15 offering of this course. LaTeX template courtesy of UC Berkeley EECS dept.*

13.0.1 Application of Complexity Penalized ERM to Markov chains

Lets consider an example of density estimation now. We consider learning the parameters of a Markov chain or order m. Recall that the distribution of a $m-$order Markov chain factorizes as follows:

$$p(X_1, \dots, X_n) = \prod_{i=1}^{n} p(X_i|X_{i-1}, \dots, X_{i-m})$$

where we assume that the terms with negative indicies are omitted from conditioning.

How to encode a Markov chain? If the order m of the Markov chain is known, then encoding the Markov chain is simply encoding its parameters. Lets consider discrete-variable Markov chains where each variable X_i takes values in some alphabet $\mathcal{X}$ of size $|\mathcal{X}|$, e.g. for binary random variables $\mathcal{X} = 0, 1$ and $|\mathcal{X}| = 2$. Also, lets consider stationary Markov chains, i.e. the distribution of $p(X_i|X_{i-1}, \dots, X_{i-m})$ is same for all i. How many parameters are there in a m^th-order stationary Markov chain defined on $|\mathcal{X}|$ alphabet size?

Lets start by arguing this for a $0th$-order chain - notice that the zero-th order Markov chain simply corresponds to i.i.d. sampling. In this case, there are $|\mathcal{X}| - 1$ parameters, corresponding to the probability of any variable X_i taking values $x \in \mathcal{X}$, minus one since the probabilities need to sum to one. For a $1st$-order Markov chain, we need to investigate the size of probability table for $p(X_i|X_{i-1})$. Thus, the number of parameters is $|\mathcal{X}|(|\mathcal{X}| - 1)$ since the conditioning variable can take $|\mathcal{X}|$ values and for each value that conditioning variable takes, X_i takes $|\mathcal{X}|$ values but the corresponding probabilities for those assignments need to sum to 1. More generally, for a m^th-order Markov chain, there are $|\mathcal{X}|^m(|\mathcal{X}| - 1) = O(|\mathcal{X}|^{m+1})$ parameters since we need to condition on m variables. [1]

Since each parameter is a probability value, we need to quantize it to be able to encode it. Again, as for the previous example, it suffices to quantize the parameters to accuracy of $1/\sqrt{n}$, i.e. restrict the parameters (probabilities) take values in $[1/(2\sqrt{n}), 3/(2\sqrt{n}), \dots, 1 - 1/(2\sqrt{n})]$. See Figure 13.1. The number of possible values each quantized parameter can take is $\sqrt{n}$ and the total possible values we need to encode is $(\sqrt{n})^{O(|\mathcal{X}|^{m+1})}$. A simple way to encode these parameters is using $O((|\mathcal{X}|^{m+1} \log_2 n)$ bits.

Thus, the complexity penalized Markov-chain estimator is given as:

$$\widehat{f} = \arg\min_{f \in \overline{\mathcal{F}}} \{\widehat{R}(f) + \frac{c(f) + \ln(1/\delta)}{n}\}$$

[1] To be precise, we also need to count the initial probabilities, e.g. $p(X_0 = x)$ for $x \in \mathcal{X}$ for a first-order Markov chain which can be different than the transition probabilities $p(X_i|X_{i-1})$, but that does not change the order of the number of parameters which is still $O(|\mathcal{X}|^{m+1})$.

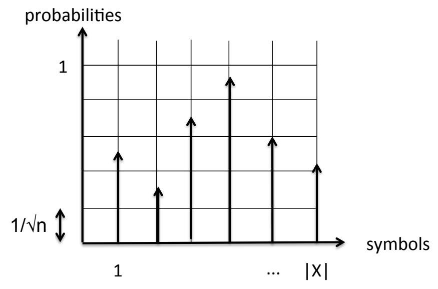

Figure 13.1: An example of quantized probabilities for $0th$-order Markov chain, into accuracy of $\frac{1}{\sqrt{n}}$

where $\overline{\mathcal{F}}$ is the class of all order m Markov chain distributions with quantized parameters, $\widehat{R}(f) = -\log f(X_1, \ldots, X_n)$ and $c(f)$ is the prefix codelength for f. We can also bound its expected excess risk:

$$\begin{aligned}
\mathbb{E}[R(\widehat{f})] - R^* &\leq \min_{f \in \overline{\mathcal{F}}} \left\{ R(f) - R^* + \frac{c(f) + \ln(1/\delta)}{n} \right\} + \delta \\
&\leq \min_{f \in \overline{\mathcal{F}}} \left\{ D_n(f^* || f) + \frac{O((|\mathcal{X}|^{m+1} \log_2 n) + \ln(1/\delta))}{n} \right\} + \delta
\end{aligned}$$

where $D_n(f^*||f) = \mathbb{E}_{f^*}[\log f^*(X_1, \ldots, X_n)/f(X_1, \ldots, X_n)]$.

It can be shown that if the true data-generating distribution f^* is indeed a Markov chain of order m, then $D_n(f^*||f) = O(1/n)$ and hence the complexity penalized approach achieves the parametric rate of error convergence $O(\text{no. of parameters}/n)$ up to log factors.

Take away message: ERM is good enough if all models in the class are equally complex or equally likely apriori, e.g., histograms of fixed resolution, Markov chains of fixed order, etc. In this case, a simple encoding using $\log |\mathcal{F}|$ bits per model works, and Complexity penalized ERM reduces to regular ERM.

However, if the models have different complexities which is typically the case when doing model selection, e.g. decision trees, wavelets, histograms of different resolutions, markov chains of unknown order, etc., then we should do complexity penalized ERM using prefix codes that are longer for more complex models (decision trees with lots of leaves, wavelet estimators with lots of non-zero coefficients, histograms of fine resolution, markov chains of large order) and shorter for simpler models.

13.1 Minimum Description Length Principle and Model selection

The Minimum Description Length (MDL) Principle states that the best description of the data is given by the model which compresses it the best. Thus, modeling is equivalent to capturing regularity in the data which is equivalent to compressing it. More we can compress the data, more we learn from it and better we are at predicting it.

Thus, MDL suggests picking the probability model that describes the data using the shortest codelength. However, one needs to be careful in interpreting it. When using MDL for model selection (the true model is unknown and we are seeking best representation from models within a class), it implies that the overall codelength needed to describe the data using a model as well as to describe a model within the class should be as small as possible. This can be achieved by a two-stage MDL procedure which connects with the complexity penalized ERM approach.

Two-stage MDL uses a prefix coding scheme where we 1) we encode the data given a model and 2) encode the model, and the overall code is a concatenation of the two prefix codes (which is itself a prefix code). Thus, the overall codelength is the sum of the codelength needed to encode the data given a model plus the codelength needed to encode the model. This suggests solving the following minimization:

$$\widehat{f} = \arg\min_{f\in\mathcal{F}} \{L_f(X_1,\ldots,X_n) + c(f)\} \tag{13.1}$$

A natural choice of $L_f(X_1,\ldots,X_n) = \log 1/f(X_1,\ldots,X_n)$, the Shannon information content when using model f to represent data $X_1,\ldots,X_n$. This is precisely the complexity-penalized likelihood loss.

13.2 Universal and Sequential Prediction and Coding

In universal prediction and coding, the source distribution is not known. The goal in universal prediction is to find a q that has $D(p||q)$ small for all $p \in \mathcal{P}$. Such a coding distribution would be universal for $\mathcal{P}$. The complexity penalized ERM predictors we considered were shown to yield low error for an entire class $\mathcal{P}$ e.g. class of Lipschitz functions or decision boundaries. We aim for such results in two scenarios - the adversarial case and an average case, as we discuss next.

Additionally, we would like to consider prediction and coding strategies that are sequential, i.e. they can predict or code on the fly using data seen up to that time, instead of having to wait to see all n data points. We focus on prediction first, i.e. coming up with models of the form $p(x_i|x_1^{i-1})$ where $x_1^{i-1} = x_{i-1},\ldots,x_1$. This is going to be our focus going forward.

13.2.1 Adversarial Case

Given a sequence $x_1^n \in \mathcal{X}^n$, we define the regret of using distribution q over p. Notice that in the adversarial case, we do not assume that x_1^n is generated according to p. Instead we think of picking a model q that does as well as a candidate model p.

$$Reg_n(q,p) := \sup_{x_1^n}\ \log\frac{1}{q(x_1^n)} - \log\frac{1}{p(x_1^n)} = \sup_{x_1^n} \sum_{i=1}^{n} \log\frac{1}{q(x_i|x_1^{i-1})} - \log\frac{1}{p(x_i|x_1^{i-1})} \tag{13.2}$$

We care about the worst-case regret with respect to a class of $\mathcal{P}$:

$$Reg_n(q,\mathcal{P}) := \sup_{p\in\mathcal{P}} Reg_n(q,p) \tag{13.3}$$

13.2.2 Redundancy minimization

Here is a less adversarial case where we assume data x_1^n is generated according to p. We define the redundancy which is the expected regret under p.

$$Red_n(q,p) := \mathbb{E}_{x_1^n\sim p}\big[\log\frac{1}{q(x_1^n)} - \log\frac{1}{p(x_1^n)}\big] = D(p||q) \tag{13.4}$$

The worst-case redundancy with respect to a class $\mathcal{P}$ is

$$Red_n(q,\mathcal{P}) := \sup_{p\in\mathcal{P}} Red_n(q,p) \tag{13.5}$$

This average setting was the focus in our previous analysis when we considered negative log likelihood loss and redundancy is simply the excess risk of model q under the negative log likelihood loss.

13.3 Minimax Strategies for Regret

There are two questions here.

- How low regret can we hope for?
- How do we achieve this low regret?

Let's define the complexity of set Θ.

$$Comp_n(\Theta) := \log \int_{\mathcal{X}^n} \sup_{\theta\in\Theta} p_\theta(x_1^n) d\mu(x_1^n) \tag{13.6}$$

where μ is some base measure on $\mathcal{X}^n$. Note that we may have $Comp_n(\Theta) = +\infty$. It turns out that the complexity equals to the minimax regret in adversarial setting.

Theorem 13.1 *The minimax regret for $\mathcal{P} = \{p_\theta\}, \theta \in \Theta$*

$$\inf_Q Reg_n(q, \mathcal{P}) = Comp_n(\Theta) \tag{13.7}$$

And if $Comp_n(\Theta) < +\infty$, then the normalized maximum likelihood estimator (as known as Shtarkov distribution) $\overline{q}$, defined with density

$$\overline{q}(x_1^n) = \frac{\sup_{\theta\in\Theta} p_\theta(x_1^n)}{\int \sup_{\theta\in\Theta} p_\theta(x_1^n) dx_1^n} \tag{13.8}$$

is uniquely minimax optimal.

Notice that the normalized maximum likelihood estimator, for each given sequence, simply assigns the probability proportional to the probability given by the candidate model with highest likelihood. However, a normalization is needed to make sure such an assignment is a valid probability distribution.

Proof: Assume $Comp_n(\Theta) < +\infty$. The normalized maximum likelihood distribution $\overline{q}$ has constant regret:

$$\begin{aligned}
Reg_n(\overline{q}, \mathcal{P}) &= \sup_{x_1^n \in \mathcal{X}} \left[\log \frac{1}{\overline{q}(x_1^n)} - \log \frac{1}{\sup_\theta p_\theta(x_1^n)}\right] && (13.9)\\
&= \sup_{x_1^n \in \mathcal{X}} \left[\log \frac{\int \sup_{\theta\in\Theta} p_\theta(x_1^n) dx_1^n}{\sup_{\theta\in\Theta} p_\theta(x_1^n)} - \log \frac{1}{\sup_\theta p_\theta(x_1^n)}\right] && (13.10)\\
&= Comp_n(\Theta) && (13.11)
\end{aligned}$$

Moreover, for any distribution Q on $\mathcal{X}^n$, we have

$$\begin{aligned}
Reg_n(q, \mathcal{P}) &\geq \int \left[\log \frac{1}{q(x_1^n)} - \log \frac{1}{\sup_\theta p_\theta(x_1^n)}\right] \overline{q}(x_1^n) d\mu(x_1^n) && (13.12)\\
&= \int \left[\log \frac{\overline{q}(x_1^n)}{q(x_1^n)} + Comp_n(\Theta)\right] \overline{q}(x_1^n) dx_1^n && (13.13)\\
&= D(\overline{q}||q) + Comp_n(\Theta) && (13.14)
\end{aligned}$$

The first inequality follows by simply lower bounding $\sup_{x_1^n}$ with $\mathbb{E}_{x_1^n \sim \overline{q}}$. The result follows by Gibbs inequality which states that KL divergence is positive unless $q = \overline{q}$ and hence any other q is strictly suboptimal. ■

Remarks 13.2

- Note that the normalized likelihood distribution is not a sequential predictor, since we must know the entire sequence to compute $q(x_1^n)$.

13.4 Mixture (Bayesian) Strategies

Given the drawback of the Normalized Maximum Likelihood estimator which is minimax optimal for regret, we consider alternate estimators. In particular, we consider a mixture approach, which is based on choosing q as convex combination (mixture) of all the possible source distribution p_θ for $\theta \in \Theta$.

In particular, given a prior π over Θ, we consider the mixture model

$$q^\pi(x_1^n) = \int_\Theta \pi(\theta) p_\theta(x_1^n) d\theta \tag{13.15}$$

To make it sequential, we start with some initial prior π and our algorithm will update the model and prior as we go as follows:

$$q^\pi(x_i|x_1^{i-1}) = \int_\Theta p_\theta(x_i|x_1^{i-1}) \pi(\theta|x_1^{i-1}) d\theta \tag{13.16}$$

$$\begin{aligned} \pi(\theta|x_1^{i-1}) &= \frac{\pi(\theta) p_\theta(x_1^{i-1})}{\int_\Theta \pi(\theta') p_{\theta'}(x_1^{i-1}) d\theta'} && (13.17) \\ &\propto \pi(\theta) e^{-\log \frac{1}{p_\theta(x_1^{i-1})}} && (13.18) \end{aligned}$$

This is referred to as the exponential weights update algorithm. It is a workhorse algorithm in online learning and we will see that it has good regret as well as redundancy guarantees.