10-704: Information Processing and Learning **Spring 2015**

Lecture 12: Feb 19

Lecturer: Aarti Singh *Scribe: Aarti Singh*

12.1 Review

Complexity Penalized ERM via prefix codes

Last time, we talked about Complexity Penalized ERM via prefix codes for loss functions that are bounded. This includes classification under 0-1 loss. The complexity penalized ERM predictor is given as follows:

$$\widehat{f} = \arg\min_{f \in \mathcal{F}} \left\{ \widehat{R}(f) + \sqrt{\frac{c(f) + \ln(1/\delta)}{2n}} \right\}$$

where $\widehat{R}(f) = \frac{1}{n}\sum_{i=1}^{n} \text{loss}(f(X_i), Y_i))$ for n i.i.d. samples $\{X_i, Y_i\}_{i=1}^n$, $c(f)$ is the prefix codelength for encoding the predictor f and $\delta \in (0,1)$ is a confidence parameter. For binary classification, for example, the $0/1$ loss is $\text{loss}(f(X_i), Y_i) = 1_{f(X_i) \neq Y_i}$. We also showed the following bound on the expected excess risk for the complexity penalized ERM predictor:

$$\mathbb{E}[R(\widehat{f})] - R^* \leq \min_{f \in \mathcal{F}} \left\{ R(f) - R^* + \sqrt{\frac{c(f) + \ln(1/\delta)}{2n}} \right\} + \delta$$

for all $\delta \in (0,1)$ where R^* is the Bayes optimal risk (i.e. risk of the best possible predictor not restricted to be in $\mathcal{F}$). The first two terms can be regarded as approximation error (bias) and the third term as estimation error (variance). Thus, complexity penalized ERM automatically does model selection, i.e. picks a predictor with performance comparable to the one in the class $\mathcal{F}$ that balances approximation and estimation errors. On the other hand, simply doing ERM is prone to over-fitting.

Application to Classification

We also studied two examples of complexity penalized ERM via prefix codes - histogram and dyadic decision tree classifiers. We derived prefix codes for them and studied approximation properties for Lipschitz class of boundaries, demonstrating rates of convergence which indicate that decision trees can outperform histogram classifiers when the boundary is well-behaved.

12.2 Complexity Penalized ERM for Regression and Density Estimation

To derive performance bounds on complexity penalized ERM predictors under bounded losses, we had used Hoeffding's concentration inequality to control the deviation of true and empirical risk. For unbounded

loss functions, we can use other concentration inequalities such as Bernstein inequality which under some moment conditions on the variables imply a tighter control on the deviation of true and empirical risk: With probability $\geq 1-\delta$, for all $f \in \mathcal{F}$

$$R(f) \leq \widehat{R}(f) + \frac{c(f) + \ln(1\delta)}{n}$$

Notice the absence of square-root on the deviation bound. The complexity penalized ERM rule is given as

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

and its expected excess risk can be bounded as follows:

$$\mathbb{E}[R(\widehat{f})] - R^* \leq \min_{f \in \mathcal{F}} \left\{ R(f) - R^* + \frac{c(f) + \ln(1/\delta)}{n} \right\} + \delta$$

For a derivation, see for example http://nowak.ece.wisc.edu/SLT09/lecture12.pdf.

Examples of unbounded loss functions arise in Regression/de-noising where the squared loss $\text{loss}(f(X_i), Y_i) = (f(X_i) - Y_i)^2$ is used, and density estimation where negative log likelihood loss $\text{loss}(f(X_i)) = -\log f(X_i)$ is used. Lets consider application of these bounds to wavelet denoising and density estimation under a Markov chain model.

12.2.1 Application to Wavelet de-noising

Wavelets are a basis that is good for representing inhomogeneous functions such as piece-wise smooth functions or functions supported locally on a small part of the domain. For example, the Haar wavelet basis (shown below) provides a good approximation for piece-wise constant functions.

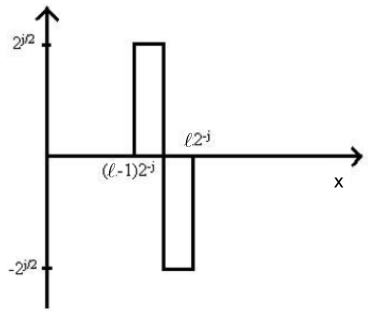

Figure 12.1: A haar wavelet basis element, $\phi_{\ell k}(x)$

A wavelet basis consists of orthonormal basis functions indexed by a scale parameter $\ell \in \{0, 1, \dots\}$ and a location parameter $k \in \{1, 2, \dots, 2^\ell\}$. For example, there is only one location at scale $\ell = 0$ and a single basis element which is supported on the entire domain and for Haar basis is a positive value on one half of the domain and a negative value on the other half. The next scale corresponding to $\ell = 1$ has two basis elements at locations $k = 1, 2$ that consist of the first element shrunk by half and supported on first half (positive on first quarter and negative on second quarter for Haar basis) and second half of the domain (positive on third quarter and negative on fourth quarter for Haar basis). And so on.

Just like any basis representation, we represent a function in the wavelet basis as follows:

$$f(x) = \sum_{\ell \geq 0} \sum_{k=1}^{2^\ell} a_{\ell k} \phi_{\ell k}(x) + c_0$$

where ϕ_{jk} is the wavelet basis element at scale ℓ and location k. Since the basis is orthonormal, we have $a_{\ell k} = \int f(x)\phi_{\ell k}(x)dx =< f, \phi_{\ell k} >$.

Wavelet basis elements satisfy certain constraints called *vanishing moments*. For example, the Haar wavelet basis satisfies $\int \phi_{\ell k}(x)dx = 0$. More generally, a s^{th}-order wavelet basis satisfies $\int x^r \phi_{\ell k}(x)dx = 0$ for all $r \leq s$ and is said to possess s vanishing moments. A nice consequence of this property is that if the function f is smooth over the support of $\phi_{\ell k}$, say is it constant for haar basis or looks like x^r for higher-order basis, then the corresponding coefficient $a_{\ell k} = 0$. Thus, wavelet basis provides a *sparse representation* for functions that are inhomogeneous. For example, the only non-zero coefficients in the Haar wavelet basis for a piece-wise constant function are the ones whose basis elements are supported over discontinuities of the function.

Typically, the function f is only evaluated at n samples and in this case, only wavelet basis up to resolution $\ell = \log_2 n - 1$ suffice i.e. n basis elements and their corresponding coefficients.

Choosing a wavelet estimator for data $\{X_i, Y_i\}_{i=1}^n$ correspond to picking a function (model) f expressed in terms of its wavelet basis expansion. We can pick a good model from this class by using complexity penalized ERM:

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

where $\widehat{R}(f) = \frac{1}{2\sigma^2 n}\sum_{i=1}^n (Y_i - f(X_i))^2$ where σ^2 is the noise variance in the model $Y_i = f^*(X_i) + \epsilon_i$, and $c(f)$ is the length of a prefix code for f.

How can we encode a wavelet estimator f? Notice that encoding f is equivalent to encoding the non-zero coefficients - their scale, location and the value. For each non-zero coefficient, there are n possible combinations of scale, location (since there are n samples) and hence that can be encoded simply with $\log_2 n$ bits (it is possible to improve this by encoding the structure of the binary tree corresponding to the recursive dyadic partitions on which the wavelet basis elements are defined, as for decision tree classifiers discussed earlier, but this suffices since we will argue that we need same order $\log_2 n$ bits to encode the value of each coefficient). The value can be any real (but bounded) number and representing this perfectly could require infinite bits, but again since we have only n samples, it suffices to quantize the wavelet coefficients and encode them. Specifically, we consider the class $\overline{\mathcal{F}}$ where the functions f have wavelet representations with coefficients quantized to an accuracy of $2M/\sqrt{n}$ if the original coefficients are bounded between $[-M, M]$ where M is some constant. Then a coefficient can be encoded with $\log_2(\sqrt{n}) = \frac{1}{2}\log_2 n$ bits. By doing this, we lose a little bit (order $1/n$ in the approximation error, which is best possible accuracy one can hope for anyways) but gain a lot in the estimation error (since we can encode functions in $\overline{\mathcal{F}}$ using few bits). Therefore,

$$c(f) = \frac{3}{2}\log_2 n \times \text{number of non-zero coeffs.}$$

Therefore, the wavelet estimator can be written as:

$$\widehat{f}(x) = \widehat{\Phi}(x)\widehat{\mathbf{a}}$$

where $\Phi(x)$ is a row vector with the wavelet basis elements concatenated with 1 and $\widehat{\mathbf{a}}$ is a column vector containing estimates of the corresponding coefficients and c_0. Thus, $c(f) = \frac{3}{2}\|\mathbf{a}\|_0 \log_2 n$. The estimated coefficients are given as:

$$\widehat{\mathbf{a}} = \arg\min_{\mathbf{a} \in A} \{\|Y - \Phi(X)\mathbf{a}\|^2 + 3\sigma^2\|\mathbf{a}\|_0 \log_2 n\}$$

where $A = [-M+M/\sqrt{n}, -M+3M/\sqrt{n}, ..., M-M/\sqrt{n}]$ is the set of quantized coefficients, $Y = [Y_1, Y_2, \ldots, Y_n]$, and $\Phi(X) = [\Phi(X_1); \Phi(X_2); \ldots; \Phi(X_n)]$. It can be shown that the solution $\widehat{\mathbf{a}}$ can be written in closed form as a hard-thresholding estimate, i.e. if $Y = \Phi(X)\mathbf{b}$, then

$$\widehat{a}_i = b_i \cdot \mathbf{1}\left(|b_i| \geq \sqrt{3\sigma^2 \log_2 n}\right)$$

where $\mathbf{1}(S) = 1$ if S is true and 0 otherwise.

Thus, a wavelet estimator is often referred to as a "non-linear" estimator since the basis elements used in the representation are the ones with largest coefficients and not simply the first few basis elements as is typical.

We can also bound the expected excess risk of the complexity penalized wavelet estimator.

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

Second inequality follows since $\overline{\mathcal{F}} \subset \mathcal{F}$. If the true underlying function f^* has d discontinuities and we are using a Haar wavelet basis, then the estimation error behaves like $(\|\mathbf{a}\|_0 \log_2 n)/n = O((d\log_2^2 n)/n)$ since every discontinuity results in no more than $\log_2 n$ non-zero coefficients (all basis elements with supports that include the point of discontinuity). Notice that $R(f) - R^* = \mathbb{E}[(f(X) - f^*(X))^2]/(2\sigma^2)$ and hence it can be shown that the approximation error behaves like $O(d/n)$ since the best wavelet estimator is quantized to $1/\sqrt{n}$ in coefficient value and to $1/n$ spatially (i.e. in the d regions of discontinuity, each of size $1/n$, the error is $O(1)$, and elsewhere the squared error is $1/n$). Thus, the overall mean square error behaves like $O((d\log_2^2 n)/n)$, which is the parametric rate of error convergence $O(\text{no. of parameters}/n)$ up to log factors.

12.2.2 Application 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:

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

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 $p(X_i|X_{i-1}, \ldots, X_{i-m})$ 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 $[1/(2\sqrt{n}), 3/(2\sqrt{n}), \ldots, 1 - 1/(2\sqrt{n})]$. See Figure 14.2. The number

[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 $O(|\mathcal{X}|^{m+1})$.

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.

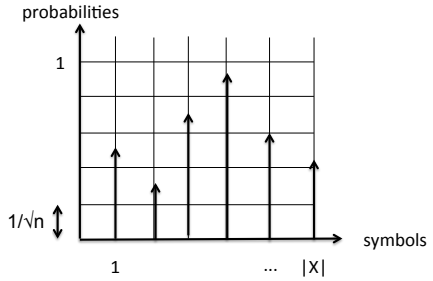

Figure 12.2: An example of quantized probabilities for $0th$-order Markov chain, into accuracy of $\frac{1}{\sqrt{n}}$

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

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

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.

12.3 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:

$$\widehat{f} = \arg\min_{f \in \mathcal{F}} \{L_f(X_1, \ldots, X_n) + c(f)\} \tag{12.1}$$

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.