

ELEN6887

Lecture 9: Errors Bounds in Countably Infinite Spaces

R. Castro

2/17/2009

1 Recap: Bounds for finite classes of models

In the last lecture, we studied bounds of the following form: for any $\delta > 0$, with probability at least $1 - \delta$,

$$R(f) \leq \widehat{R}_n(f) + \sqrt{\frac{\log |\mathcal{F}| + \log\left(\frac{1}{\delta}\right)}{2n}}, \quad \forall f \in \mathcal{F}$$

which led to upper bounds on the estimation error of the form

$$E[R(\widehat{f}_n)] - \min_{f \in \mathcal{F}} R(f) \leq \sqrt{\frac{\log |\mathcal{F}| + \log(n) + 2}{n}}$$

The key assumptions made in deriving the error bounds were:

- (i) - bounded loss function $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow [0, 1]$
- (ii) - finite collection of candidate functions

The bounds are valid for every P_{XY} and are called *distribution-free*.

2 Deriving Bounds for Countably Infinite Spaces

In this lecture we will generalize the previous results in a powerful way by developing bounds applicable to possibly infinite collections of candidates. This generalization is going to lead us to complexity regularization.

To start let us suppose that \mathcal{F} is a countable, possibly infinite, collection of candidate functions (as a reminder remember that \mathbb{N} and \mathbb{Q} are countably infinity sets). Assign a positive number $c(f)$ to each $f \in \mathcal{F}$, such that

$$\sum_{f \in \mathcal{F}} e^{-c(f)} \leq 1.$$

The number $c(f)$ can be interpreted as

- (i) - measures of complexity of f
- (ii) - $-\log$ of prior probabilities
- (iii) - length of a codeword describing f

The interpretation (ii) is perhaps the most familiar. Suppose you have a *prior* probability distribution $p(\cdot)$ over \mathcal{F} , so that $p(f) \geq 0$ for all $f \in \mathcal{F}$ and

$$\sum_{f \in \mathcal{F}} p(f) = 1.$$

Then taking $c(f) = \log p(f)$ guarantees that $\sum_{f \in \mathcal{F}} e^{-c(f)} = 1$.

Recall the steps of the derivation in the previous lecture. Hoeffding's inequality tells us that for each f and every $\epsilon > 0$

$$P\left(R(f) - \widehat{R}_n(f) \geq \epsilon\right) \leq e^{-2n\epsilon^2},$$

or, in other words, for every $\delta > 0$

$$P\left(R(f) - \widehat{R}_n(f) \geq \sqrt{\frac{\log\left(\frac{1}{\delta}\right)}{2n}}\right) \leq \delta.$$

Now let $\delta > 0$ and use the values $c(f)$ to define $\delta(f) = \delta e^{-c(f)}$. We can then write

$$P\left(R(f) - \widehat{R}_n(f) \geq \sqrt{\frac{\log\left(\frac{1}{\delta(f)}\right)}{2n}}\right) \leq \delta(f).$$

Furthermore we can apply the union bound as follows

$$\begin{aligned} P\left(\exists f \in \mathcal{F} : R(f) - \widehat{R}_n(f) \geq \sqrt{\frac{\log(1/\delta(f))}{2n}}\right) &= P\left(\sup_{f \in \mathcal{F}} \left\{R(f) - \widehat{R}_n(f) - \sqrt{\frac{\log(1/\delta(f))}{2n}}\right\} \geq 0\right) \\ &= P\left(\bigcup_{f \in \mathcal{F}} \left\{R(f) - \widehat{R}_n(f) - \sqrt{\frac{\log(1/\delta(f))}{2n}} \geq 0\right\}\right) \\ &\leq \sum_{f \in \mathcal{F}} P\left(R(f) - \widehat{R}_n(f) \geq \sqrt{\frac{\log\left(\frac{1}{\delta(f)}\right)}{2n}}\right) \\ &\leq \sum_{f \in \mathcal{F}} \delta(f) = \sum_{f \in \mathcal{F}} e^{-c(f)} \delta = \delta. \end{aligned}$$

So for any $\delta > 0$ with probability at least $1 - \delta$, we have that $\forall f \in \mathcal{F}$

$$\begin{aligned} R(f) &\leq \widehat{R}_n(f) + \sqrt{\frac{\log \frac{1}{\delta(f)}}{2n}} \\ &= \widehat{R}_n(f) + \sqrt{\frac{c(f) + \log \frac{1}{\delta}}{2n}} \end{aligned}$$

2.1 A Special Case - \mathcal{F} finite

Suppose \mathcal{F} is finite and let $c(f) = \log |\mathcal{F}| \quad \forall f \in \mathcal{F}$. Then

$$\sum_{f \in \mathcal{F}} e^{-c(f)} = \sum_{f \in \mathcal{F}} e^{-\log |\mathcal{F}|} = \sum_{f \in \mathcal{F}} \frac{1}{|\mathcal{F}|} = 1,$$

and $\delta(f) = \frac{\delta}{|\mathcal{F}|}$ which implies that for any $\delta > 0$ with probability at least $1 - \delta$, we have

$$R(f) \leq \widehat{R}_n(f) + \sqrt{\frac{\log |\mathcal{F}| + \log\left(\frac{1}{\delta(f)}\right)}{2n}}, \quad \forall f \in \mathcal{F}.$$

Note that this is precisely the bound we derived in the last lecture.

3 Choosing the values $c(f)$

The generalized bounds allow us to handle countably infinite collections of candidate functions, but we need to have a set of value $c(f)$ for all $f \in \mathcal{F}$ such that

$$\sum_{f \in \mathcal{F}} e^{-c(f)} \leq 1 .$$

If we have a proper prior probability distribution over \mathcal{F} we already seen a way of assigning the values $c(f)$ to each model f . However, it may be difficult to design a probability distribution over an infinite class of candidates. Another approach, that based on coding arguments, is much more practical for our purposes.

Assume that we have assigned a uniquely decodable binary codeword to each $f \in \mathcal{F}$, and let $c(f)$ denote the codelength for f . That is, the codeword for f is $c(f)$ bits long. A very useful class of uniquely decodable codes are called **prefix codes**.

Definition 1 *A code is called a prefix or instantaneous code if no codeword is a prefix of any other codeword.*

Example 1 *(From Cover & Thomas '91)*

Consider an alphabet of symbols, say A, B, C, and D and the codebooks in Figure 1. In the singular codebook

Symbol	Singular Codebook	Nonsingular But Not Uniquely Decodable	Uniquely Decodable But Not a Prefix Code	Prefix Code
A	0	0	10	0
B	0	010	00	10
C	0	01	11	110
D	0	10	110	1110

Figure 1: Four possible codes for an alphabet of four symbols.

we assign the same codeword to each symbol - a system that is obviously flawed! In the second case, the codes are not singular but the codeword 010 could represent B or CA or AD. Hence it is not a uniquely decodable codebook.

The third and fourth cases are both examples of uniquely decodable codebooks, but the fourth has the added feature that no codeword is a prefix of another. Prefix codes can be decoded from left to right since each codeword is “self-punctuating” - in this case with a zero to indicate the end of each word.

To design a uniquely decodable codebook in general is as challenging as the problem of selecting $c(f)$ to satisfy

$$\sum_{f \in \mathcal{F}} e^{-c(f)} \leq 1$$

However, prefix codes can often be easily designed or specified and they are inherently decodable. Moreover, prefix codes satisfy an important inequality called the **Kraft Inequality**.

4 The Kraft Inequality

For any binary prefix code, the codeword lengths c_1, c_2, \dots satisfy

$$\sum_{i=1}^{\infty} 2^{-c_i} \leq 1$$

Conversely, given any c_1, c_2, \dots satisfying the inequality above we can construct a prefix code with these codeword lengths. We prove this result later, but now let's see how this is useful in our learning problem.

Assume that we have assigned a binary prefix codeword to each $f \in \mathcal{F}$, and let $c(f)$ denote the bit-length of the codeword for f . Set $\delta(f) = 2^{-c(f)}\delta$. Then

$$\begin{aligned} P\left(\bigcup_{f \in \mathcal{F}} R(f) - \widehat{R}_n(f) \geq \sqrt{\frac{\log\left(\frac{1}{\delta(f)}\right)}{2n}}\right) &\leq \sum_{f \in \mathcal{F}} P\left(R(f) - \widehat{R}_n(f) \geq \sqrt{\frac{\log\left(\frac{1}{\delta(f)}\right)}{2n}}\right) \\ &\leq \sum_{f \in \mathcal{F}} \delta(f) = \sum_{f \in \mathcal{F}} 2^{-c(f)}\delta = \delta \end{aligned}$$

This implies that for any $\delta > 0$ with probability at least $1 - \delta$ we have $\forall f \in \mathcal{F}$

$$\begin{aligned} R(f) &\leq \widehat{R}_n(f) + \sqrt{\frac{\log\left(\frac{1}{\delta(f)}\right)}{2n}} \\ &= \widehat{R}_n(f) + \sqrt{\frac{c(f) \log 2 + \log\left(\frac{1}{\delta}\right)}{2n}} \end{aligned}$$

4.1 Application - Structural Risk Minimization

Let $\mathcal{F}_1, \mathcal{F}_2, \dots$, be a sequence of finite classes of candidate models with $|\mathcal{F}_1| < |\mathcal{F}_2| < \dots$. Let $\mathcal{F} = \bigcup_{i=1}^{\infty} \mathcal{F}_i$. We can design a prefix code for all the elements of \mathcal{F} in a simple way: Use the codes 0, 10, 110, 1110, ... to encode the subscript i in \mathcal{F}_i . For each class \mathcal{F}_i , construct a set of binary codewords of length $\lceil \log_2 |\mathcal{F}_i| \rceil$ to uniquely encode each function in \mathcal{F}_i . Now we just need to concatenate the two codes. For any given function $f \in \mathcal{F}$ use the first code to encode the smallest index i such that $f \in \mathcal{F}_i$, followed by a codeword of length $\lceil \log_2 |\mathcal{F}_i| \rceil$ identifying $f \in \mathcal{F}_i$. You can easily show this is a prefix code.

Example 2 Histogram Classifiers: Let $\mathcal{X} = [0, 1]^d$ and $\mathcal{Y} = \{0, 1\}$. Let $\mathcal{F}_k, k = 1, 2, \dots$, denote the collection of histogram classification rules with k equal volume bins. Note that $|\mathcal{F}_k| = 2^k$. We can use the approach just described to encode any histogram classifier f in the class $\mathcal{F} = \bigcup_{i=1}^{\infty} \mathcal{F}_i$. Use k bits to indicate the smallest k such that $f \in \mathcal{F}_k$, and then use $\log_2 |\mathcal{F}_k| = k$ bits to indicate which of the 2^k possible histogram rules it is. Thus we have constructed a prefix code, and for any $f \in \mathcal{F}_k$ for some $k \geq 1$ the codeword assigned to f has $c(f) = k + k = 2k$ bits. It follows that for any $\delta > 0$ with probability at least $1 - \delta$ we have $\forall f \in \bigcup_{k \geq 1} \mathcal{F}_k$

$$R(f) \leq \widehat{R}_n(f) + \sqrt{\frac{2k_f \log 2 + \log\left(\frac{1}{\delta}\right)}{2n}},$$

where k_f is the number of bins in histogram corresponding to f . Contrast with the bound we had for the class of m bin histograms alone: with probability $\geq 1 - \delta$, $\forall f \in \mathcal{F}_m$

$$R(f) \leq \widehat{R}_n(f) + \sqrt{\frac{m \log 2 + \log\left(\frac{1}{\delta(f)}\right)}{2n}}$$

Notice the bound for all histograms rules is almost as good as the bound for on only the m -bin rules. That is, when $k_f = m$ the bounds are within a factor of $\sqrt{2}$. On the other hand, the new bound is a big improvement, since it also gives us a guide for selecting the number of bins based on the data, suggesting the rule

$$\hat{f}_n = \arg \max_{f \in \mathcal{F}} \widehat{R}_n(f) + \sqrt{\frac{2k_f \log 2 + \log\left(\frac{1}{n}\right)}{2n}}.$$

5 Proof of the Kraft Inequality

We will prove that for any binary prefix code, the codeword lengths c_1, c_2, \dots , satisfy $\sum_{k \geq 1} 2^{-c_k} \leq 1$. The converse is easy to prove also, but it not central to our purposes here (for a proof, see Cover & Thomas '91). Consider a binary tree like the one shown in Figure 5

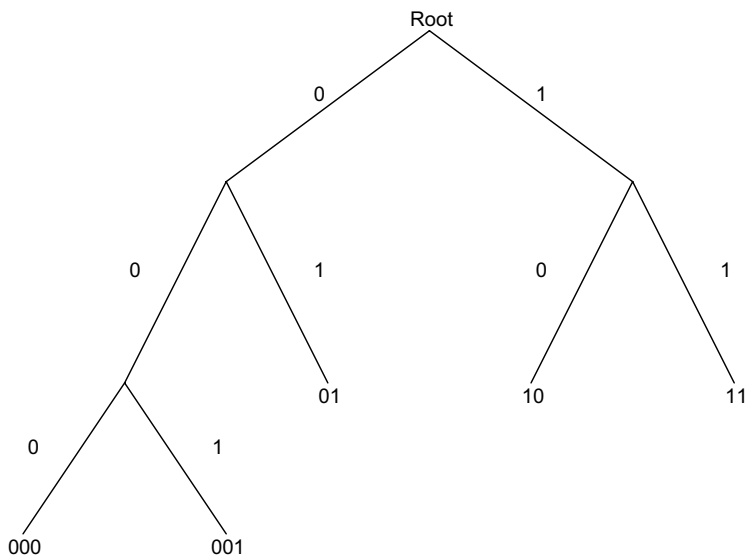


Figure 2: A binary tree.

The sequence of bit values leading from the root to a leaf of the tree represents a codeword. The prefix condition implies that no codeword is a descendant of any other codeword in the tree. Therefore we can each leaf of the tree represents a codeword in our code. Let c_{\max} be the length of the longest codeword (also the number of branches to the deepest leaf) in the tree.

Consider a node in the tree at level c_i . This node in the tree can have at most $2^{c_{\max} - c_i}$ descendants at level c_{\max} . Furthermore, for each leaf of the tree the set of possible descendants at level c_{\max} is disjoint (since no codeword can be a prefix of another). Therefore, since the total number of possible leaves at level c_{\max} is $2^{c_{\max}}$, we have

$$\sum_{i \in \text{leafs}} 2^{c_{\max} - c_i} \leq 2^{c_{\max}} \Rightarrow \sum_{i \in \text{leafs}} 2^{-c_i} \leq 1$$

which proves the case when the number of codewords is finite.

Suppose now that we have a countably infinite number of codewords. Let b_1, b_2, \dots, b_{c_i} be the i^{th} codeword and let

$$r_i = \sum_{j=i}^{c_i} b_j 2^{-j}$$

be the real number corresponding to the binary expansion of the codeword (in binary $r_i = 0.b_1 b_2 b_3 \dots$). We can associate the interval $[r_i, r_i + 2^{-c_i})$ with the i^{th} codeword. This is the set of all real numbers whose binary expansion begins with b_1, b_2, \dots, b_{c_i} . Since this is a subinterval of $[0, 1]$, and all such subintervals corresponding to prefix codewords are disjoint, the sum of their lengths must be less than or equal to 1. This proves the case where the number of codewords is infinite.