CS281B/Stat241B: Advanced Topics in Learning & Decision Making

MDL Introduction and Source Coding

Lecturer: Michael I. Jordan *Scribes: Nemanja Isailovic*

1 Introduction to Minimum Description Length

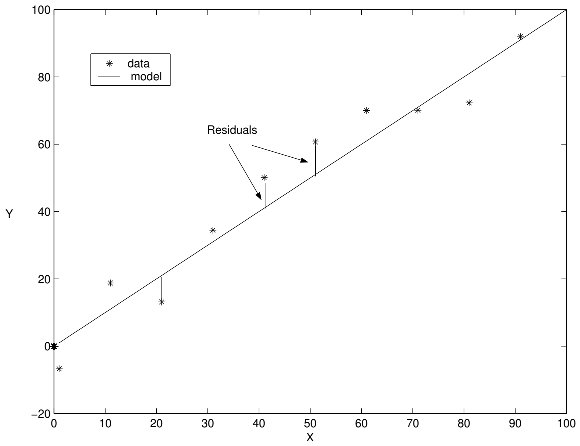
Minimum Description Length (MDL) is another approach to model selection. The machinery seems very different from the other methods we have previously seen but we'll see that at the end of the day, MDL will provide us a very familiar formula for approximating the marginal likelihood in which there are two terms: the likelihood and an additive penalty function.

There are various levels of MDL corresponding to different ways of coding the data. The simplest one leads to exactly the BIC approximation for the marginal likelihood (and for this reason BIC is sometimes called also MDL).

In the MDL framework, a sender wants to *transmit* the original data to a receiver; in order to do so, the data must be *encoded*. The encoded message is transmitted though a channel, is received by the receiver and decoded in order to retrieve the original message. In MDL we consider the case of lossless coding in which the message reconstructed by the receiver is identical to the original data, up to a quantization level.

The goal is to encode the original data in such a way that the number of bits sent is minimum; the number of bits needed to code the data gives a measure of the complexity of the data from the point of view of a model.

Consider the example in Figure 1. Suppose that both the receiver and the sender know the values for the covariates x . The sender needs therefore to send only the responses y, or alternatively, the model (the line) and the residuals. As the residuals are smaller in magnitude than the data y, fewer bits are necessary to send the residuals than the original data. If then the sender finds a better model, the residuals will be even smaller, and fewer bits are transmitted. It seems like that the more complex the model, the smaller the residuals, and fewer bits need to be transmitted. This is in fact true, but there is a caviat: the model



needs to be transmitted too and in order to do so, the model parameters must be quantized and transmitted; a more complex model has more parameters and more bits are needed. The bit gain we got from having smaller residuals thanks to a more complex model is compensated by more bits needed to send the model.

MDL tries to minimize the *sum* of the bits needed for both the residuals and the model encoding, and to find the model for which this sum is minimum.

2 Introduction to Information Theory and Random Facts

Key ideas in Information Theory:

- entropy: gives a lower bound on code length (source coding: Huffman codes, arithmetic codes)
- mutual information: characterizes channel capacity (error control coding: convolutional codes, algebraic codes, turbo codes)

(Aside: Shannon proved lower bounds in both areas.)

Our focus will be source coding:

- a source generates data
- data needs to be transmitted to receiver
- in the worst case, send it as is; but we'd like to compress it (code it) first

Weak Law of Large Numbers:
Given that random variables X_i are IID and $E[X_i] = \mu < \infty$
$((1/n)\sum_{i=1}^{n} X_i) \to \mu$ as $n \to \infty$
In other words, if the mean is finite, the probability that the sample average approaches the mean as the sample size gets large is one.

Strong Law of Large Numbers:
Given that random variables X_i are IID and $E[|X_i|] < \infty$
$((1/n)\sum_{i=1}^{n} X_i) \to \mu$ "almost surely" as $n \to \infty$

Some definitions:
$P(X = x) = Pr\{\omega : X(\omega) = x\}$
Convergence in probability ($X_n \to X$): $P(|X_n - X| > \epsilon) \to 0$ as $n \to \infty$
Almost sure convergence ($X_n \to X$): $\{\omega : X_n(\omega) \to X(\omega) \text{ as } n \to \infty\}$ has probability one

3 Prefix Codes

Random Variable: X: $\Omega \to \chi$ (where χ is the outcome)
Code: C: $\chi \to D^*$ (where D^* is the set of all possible strings)
(Meaning: The code assigns to each outcome of the random variable a string in D^*.)
$C(x)$: codeword corresponding to x
$L(x)$: length of $C(x)$
expected length of code: $L(C) = \sum_x p(x)L(x)$ where p(x) is the probability of x

Prefix Codes:

- no codeword is a prefix of any other
- corresponds to a tree

Lemma (*Kraft Inequality*) For any prefix code over an alphabet of size D, we have:

$$\sum_i D^{-Li} \leq 1.$$

Proof: Let L_{max} be the longest codeword length. Build a balanced tree of depth L_{max}. We have D^{Lmax} leaves. Consider a codeword of length L_i. Consider all leaves that lie beneach this codeword in the full tree. There are $D^{Lmax-Li}$ of these. The sets of leaves are disjoint by the prefix condition. Thus $\sum_i D^{Lmax-Li} \leq D^{Lmax}$, which implies that $\sum_i D^{-Li} \leq 1$.

Theorem: The expected length $L(C)$ of any (prefix) code satisfies $L(C) \geq H(x)$, where the **entropy** $H(x) = -\sum_x p(x) \log p(x)$ by definition.

Proof:

$$\begin{aligned}
L(C) - H(x) &= \sum_i p_i L_i + \sum_i p_i \log p_i \\
&= -\sum_i p_i \log_D D^{-Li} + \sum_i p_i \log p_i \\
&= -\sum_i p_i \log_D (D^{-Li} / \sum_i D^{-Li}) + \sum_i p_i \log p_i - \log_D (\sum_i D^{-Li}) \\
&= \sum_i p_i \log(p_i/q_i) - \log_D (\sum_i D^{-Li}),
\end{aligned}$$

where $q_i = D^{-Li} / \sum_i D^{-Li}$. The first term ($\sum_i p_i \log(p_i/q_i)$) is just KL-divergence, which is greater than or equal to 0 by definition.

Also, $\sum_i D^{-Li} \leq 1$ by the Kraft Inequality, so the second term is greater than or equal to 0.

Thus, we have $L(C) - H(x) \geq 0$.

What about achieving the lower bound?
Attempt #1: Use $L_i = \log(1/p_i) = -\log p_i$
But $\log p_i$ is not necessarily an integer, so approximate using $L_i = \lceil -\log p_i \rceil$. This yields:

$$-\log p_i \leq L_i \leq -\log p_i + 1$$

and thus:

$$H(x) \leq L(C) < H(x) + 1.$$

So we come within one bit of the entropy lower bound. But this isn't necessarily very good.

Using the "wrong" distribution: (we don't have p_i) so use q_i and design a code based on $-\log q_i$:

$$\begin{aligned}
L(C_{suboptimal}) &= \sum_i p_i (-\log q_i) \\
&\geq \sum_i p_i \log(1/q_i)
\end{aligned}$$

$$\begin{aligned} &= \sum_i p_i \log(p_i/q_i) - \sum_i p_i \log p_i \\ &= D(p_i||q_i) + H(x) \\ &= \text{KL-divergence + entropy.} \end{aligned}$$

Conclusion: length of suboptimal code = length of optimal code + KL-divergence.

So the KL-divergence is the extra price in terms of average codelength you pay for using the wrong probability distribution.

4 Asymptotic Equipartition Property (AEP)

Random variables X_i: IID, coming from some distribution $p(x)$
Lemma: $(-1/n)\log p(x_1, x_2, x_3, ...x_n) \to H(X)$
Proof: $(-1/n)\log p(x_1, x_2, x_3, ...x_n) = (-1/n)\log \prod_{i=1}^n p(x_i) = (-1/n)\sum_{i=1}^n \log p(x_i)$
By the weak law of large numbers this approaches $-\sum p(x_i)\log p(x_i) = H(x)$.

Typical Set: $A_\epsilon^{(n)}$: a set of sequences $(x_1, x_2, x_3, ...x_n)$ such that
$2^{-n(H(x)+\epsilon)} \leq p(x_1, x_2, x_3, ...x_n) \leq 2^{-n(H(x)-\epsilon)}$
Properties:

- $p(A_\epsilon^{(n)}) > 1 - \epsilon$ for sufficiently large n
  Meaning: The typical set has very high probability.
- $|A_\epsilon^{(n)}| \leq 2^{n(H(x)+\epsilon)}$
  Meaning: It's a relatively small set.
- $|A_\epsilon^{(n)}| \leq (1-\epsilon)2^{n(H(x)-\epsilon)}$

In other words, a typical set is a relatively small set of elements which comprise the vast majority of the probability of the universe and which all have nearly uniform probability.

Example of a Typical Set: We have a coin with 0.7 probability of heads and 0.3 probability of tails. (Let heads = 1, tails = 0.) Suppose we flip it 100 times. The most probable outcome is all heads: 11111...11. But there's only one such configuration. There are many more configurations possible as we equalize the number of heads and tails. Logically, we expect the outcome of about 70 heads and 30 tails. And this turns out to be the typical set (give or take a few heads or tails).

5 Achievability of Entropy

Coding $A_\epsilon^{(n)}$: index each member
We have $2^{n(H(x)+\epsilon)}$ members (all at the same probability). So we need $n(H(x)+\epsilon)+1$ bits (round up). Plus we'll add one bit to represent $A_\epsilon^{(n)}$ versus not $A_\epsilon^{(n)}$. So the total size is $n(H(x)+\epsilon)+2$ bits.

Coding not $A_\epsilon^{(n)}$: number of total sequences = $|X|^n$
So use $n\log X + 1$ bits. It's a very dumb code, but it's very low probability, so who cares?

"Code length for a sequence of length n":

$$\begin{aligned}
L^{(n)}(C) &= \sum_{xn} p(x^n)L(x^n) \\
&= \sum_{xn\epsilon A} p(x^n)L(x^n) + \sum_{xn\epsilon notA} p(x^n)L(x^n) \\
&\leq \sum_{xn\epsilon A} p(x^n)(n(H+\epsilon)+2) + \sum_{xn\epsilon notA} p(x^n)(n\log|X|+2) \\
&= p(A_\epsilon^{(n)})(n(H+\epsilon)+2) + p((A_\epsilon^{(n)})^C)(n\log|X|+2) \\
&\leq n(H+\epsilon)+2+\epsilon(n\log|X|+2) \\
&= n(H+\epsilon')
\end{aligned}$$

where $\epsilon' = \epsilon + \epsilon\log|X| + 2/n$. Given that ϵ' can be made arbitrarily small, $(1/n)L^{(n)}(C)$ can be made arbitrarily close to H.

6 Design of Codes

6.1 Huffman Codes

Build a tree from the bottom up:
List the probability for each outcome.

- Take the smallest pair of probabilities and combine them into a supernode with two leaves (one for each of the probabilities).
- Remove these probailities from the list, but add their sum to the list.
- Repeat until you're down to one probabilty in the list.

Example:
List of probabilities: $x_1 - 0.02, x_2 - 0.03, x_3 - 0.04, x_4 - 0.04, x_5 - 0.12, x_6 - 0.26, x_7 - 0.49$
Sum of smallest pair: $x_1 + x_2 = 0.05$
New list: $x_1x_2 - 0.05, x_3 - 0.04, x_4 - 0.04, x_5 - 0.12, x_6 - 0.26, x_7 - 0.49$
Sum of smallest pair: $x_3 + x_4 = 0.08$
New list: $x_1x_2 - 0.05, x_3x_4 - 0.08, x_5 - 0.12, x_6 - 0.26, x_7 - 0.49$
Sum of smallest pair: $x_1x_2 + x_3x_4 = 0.13$
New list: $x_1x_2x_3x_4 - 0.13, x_5 - 0.12, x_6 - 0.26, x_7 - 0.49$
Sum of smallest pair: $x_1x_2x_3x_4 + x_5 = 0.25$
New list: $x_1x_2x_3x_4x_5 - 0.25, x_6 - 0.26, x_7 - 0.49$
Sum of smallest pair: $x_1x_2x_3x_4x_5 + x_6 = 0.51$
New list: $x_1x_2x_3x_4x_5x_6 - 0.51, x_7 - 0.49$
Sum of smallest pair: $x_1x_2x_3x_4x_5x_6 + x_7 = 1$
New list: $x_1x_2x_3x_4x_5x_6x_7 - 1$

Using these calculations, a possible coding would be:
$x_7 : 1$
$x_6 : 01$
$x_5 : 001$

x_4 : 00011
x_3 : 00010
x_2 : 00001
x_1 : 00000

Result:
$L(C) = 2.02$
$H(x) = 2.0128$

6.2 Other Codes

Arithmetic Codes: good for big block lengths; online version of Huffman Codes that saves time and space
Lempel-Ziv: an adaptive code based on a dictionary.