Stochastic Encoding and the "Bits-Back" Argument

Jason Rennie
jrennie@ai.mit.edu

May 16, 2003

Abstract

The Minimum Description Length framework is powerful but is often overlooked. I believe that one reason for this is that methods for attaining efficient encodings are subtle. In this paper, I discuss one of those techniques, stochastic encoding. When there are multiple nearly equally valuable choices of a parameter, it is more valuable to choose stochastically—according to a probability distribution— rather than selecting the single best choice. Why? Because information can be transmitted in which parameter is chosen. This is exactly the "bits-back" argument given by Hinton and Zemel in [1].

In the Minimum Description Length (MDL) framework, the objective is to encode the data plus the model with the fewest number of bits possible. An important advantage to this framework is that the regularizer is simple and innate. Any complexity of the model must be encoded alongside the data. Hence, it is of the utmost importance that the model be encoded efficiently. In particular, the model must be encoded at the proper level of precision. Some parameters of the model may be encoded with a low degree of precision to achieve the desired benefit, while other parameters will need a high degree of precision. Designing a code to handle this in a dynamic fashion is not easy. So, we do something similar to what we do with encoding data.

When encoding data, we don't try to construct a code that actually encodes the data. That would force us to deal with the discreteness of real codes and the need to adapt the code to different distributions. Instead, we simply encode based on the uncertainty. If our model says that a label is highly likely, it takes us little encoding; if our model goofs and declares a label unlikely, we pay by using many bits to encode that label. We use the encoded model to determine a conditional probability for each label given

its example, $p(y|x)$. Assuming the label is encoded efficiently according to that probability, we use a code length of

$$-\log p(y|x) \tag{1}$$

bits. Thus, we can encode the data (the labels) efficiently without having to worry about constructing a code.

We apply the same reasoning to the encoding of model parameters. Consider transmitting model parameters using as few bits as possible. We cannot transmit with infinite precision since doing so would require infinite bandwidth. One option is to limit our choices to a discrete set. However, this poses difficulties since we must establish at what level of precision we wish to transmit each parameter. Another option is to transmit a random parameter chosen according to a distribution. The entropy of the distribution determines the effective precision of our encoding. It may seem that this requires as much bandwidth as transmitting parameters with infinite precision. But, we don't choose a single value to transmit; we are satisfied with randomly selecting from a distribution. Assume that we have established an encoding for w which is optimal according to $p(w)$. Let $q(w)$ be the distribution from which we choose a parameter to send. Then, the expected length of transmitting w is

$$l(q) = E_{w\sim q}[-\log p(w)] = -\int q(w)\log p(w)dw. \tag{2}$$

But, we are not done. Since we do not send a single value but rather according to a distribution q, we could encode information in the values that we select. The average amount information we transmit is the entropy of q. So, the net (average) encoding length we need is

$$l(q) = E_{w\sim q}[-\log p(w)] - H(q). \tag{3}$$

Or, in a more recognizable form, this is the KL-divergence,

$$l(q) = \int_\theta q(w)\log\frac{q(w)}{p(w)}dw. \tag{4}$$

That is, the average net bits we consume is the difference between the number of bits needed to send q via its optimal encoding and the number of bits needed to send q via p's optimal encoding. We get $H(q)$ "bits back" since we could utilize those bits for transmitting other information. This is the argument found in [1]. The clear benefit here is that we no longer need to

concern ourselves with establishing discrete values for the parameters. If we use a parametric family for q, we can directly optimize those parameters.

That which remains is to choose a suitable prior for p, a suitable parametric family for q and to evaluate the KL-divergence integral. We would also like to know whether there are certain choices that lead us to a convex optimization.

References

[1] Geoffrey E. Hinton and Richard S. Zemel. Autoencoders, minimum description length, and helmholtz free energy. In *Advances in Neural Information Processing Systems 6*, 1994.