Lecture Note: Entropy, Information, Cross-Entropy & ML as Minimal Description Length

Marc Toussaint

Learning & Intelligent Systems Lab, TU Berlin

August 15, 2024

Entropy

Entropy is *expected neg-log-likelihood*, which can also be desribed as expected surprise, expected code length, or expected error. The notes below are to explain this, as well as clarify the relation between ML objectives and minimal description length.

Let $p(x)$ be the probability or likelihood of a sample x.

Consider $x \in \{A, B, C, D\}$. If the distribution over these four symbols was $p(\cdot) = [.7, .1, .1, .1]$, then you would be less surprised to see a sample $x = A$ than a sample $x = B$. If you want an optimal encoding/compression of samples from this distribution, you would allocate a shorter code for the frequent symbol A, and a longer code for the infrequent symbols B, C, D (cf. Huffman code).

The neg-log-likelihood $-\log p(x)$ is also called *surprise*. It provides the optimal *code length* you should assign to a symbol x (modulo rounding): E.g., for $p(\cdot) = [.5, .25, .25, .0]$, $-\log_2 p(x)$ equals 1 bit for symbol $x = A$, 2 bits for symbols B and C, and D never appears. When using $\log_2$, bits is the unit of neg-log-likelihood and can also be thought of as unit of *information*: a sample A surprises you less and only gives 1 bit of information, a sample B surprises you more and gives you 2 bits of information.

The entropy

$$H(p) = -\sum_x p(x) \log p(x) = \mathbb{E}_{p(x)}\{-\log p(x)\} \tag{1}$$

is the expected neg-log-likelihood. It is a measure of the distribution p itself, not of a specific sample. It measures, how much *in average* samples of p surprise you. Or: What is the average coding length of samples of p. Or: What is the average information given by samples of p. The unit of entropy is bits (when using $\log_2$). That is, we can specifically say "the average information given by a sample of p is as much as the information given by $H(p)$ uniform binary variables".

Sometimes another unit of information is used: The *perplexity* of p is defined as $PP(p) = 2^{H(p)}$ (when using $\log_2$), which is one-to-one with entropy but uses different units of information. Note that a discrete uniform random variable of cardinality PP has entropy $H = \log_2 PP$, which explains the definition of perplexity. Therefore, we can now say "the average information given by a sample of p is as much as the information given by a uniform discrete random variable of cardinality $PP(p)$".

Given a gaussian distribution $p(x) \propto \exp(-\frac{1}{2}(x-\mu)^2/\sigma^2)$, the neg-log-likelihood of a specific sample x is $-\log p(x) = -\frac{1}{2}(x-\mu)^2/\sigma^2 + \textit{const}$. This can be thought of as the *square error* of x from μ, and its expectation (entropy) is the mean square error. Generally, the neg-log-likelihood $-\log p(x)$ often relates to an error or loss function.

Cross-Entropy

The cross-entropy

$$H(p,q) = -\sum_x p(x) \log q(x) = \mathbb{E}_{p(x)}\{-\log q(x)\} \tag{2}$$

is also an expected neg-log-likelihood, but expectation is w.r.t. p, while the nll is w.r.t. q. This corresponds to the situation that samples are actually drawn from p, but you model or encode them using q. So your measure of surprise or code length is relative to q. In ML, q is typically a learned distribution or function model, $H(p,q)$ measures the code length of using the learned model q to encode samples from the true distribution p.

In ML, cross-entropy is often used as a classification loss function $\ell(\bar{y}, q_\theta(y|x))$ between the true class label $\bar{y}$ and the predicted class distribution $q_\theta(y|x)$ (for some input x). For each individual data point, the true class label $\bar{y}$ is not probabilistic, but we can use it to define a one-hot-encoding $p_{\bar{y}}(\cdot) = [0, .., 0, 1, 0, .., 0]$ with 1 for $y = \bar{y}$. The cross-entropy is then nothing but the neg-log-likelihood of the true class label under the learned model:

$$H(p_{\bar{y}}, q_\theta(\cdot|x)) = -\log q_\theta(\bar{y}|x) \ . \tag{3}$$

Note that we could equally cast a square error loss as a cross-entropy: If y is continuous, $q_\theta(y|x)$ Gaussian around a mean prediction $f_\theta(x)$, and $p_{\bar{y}}(y) = \delta_{\bar{y}}(y)$ the Kronecker distribution (the "continuous one-hot-encoding"), then the cross-entropy loss $H(p_{\bar{y}}, q_\theta(\cdot|x))$ is nothing but square error.

Relative Entropy (KL-divergence)

The Kullback-Leibler divergence, also called relative entropy, is defined as

$$D\big(p \,\big\|\, q\big) = \sum_x p(x) \log \frac{p(x)}{q(x)} = \mathbb{E}_{p(x)}\left\{\log \frac{p(x)}{q(x)}\right\} \ . \tag{4}$$

Given our definitions above, we can rewrite it as

$$D\big(p \,\big\|\, q\big) = H(p,q) - H(p) \ . \tag{5}$$

Note that we described $H(p)$ as the expected code length when encoding p-samples using a p-model; and $H(p,q)$ as the expected code length when encoding p-samples using a q-model. The latter will always be larger than $H(q)$, as if $q \neq p$ you will not have a perfect model to define an encoding.

In general $D\big(p \,\big\|\, q\big) \geq 0$. The proof is directly given by the Jenson's inequality (the log is concave, and pulling a concave function out of an expectation (=linear interpolation) gives $\leq$).

The KL-divergence is a measure of similarity between q and p. As it is not symmetric, it is better to say: a measure of how similar q is to p. It measures how much longer the average code length is if you use q instead of the perfect knowledge p to define an encoding of p-samples.

In the context of ML, if you want to model data from from a true $p(y|x)$, even a perfect model will have non-zero error: If the data is inherently stochastic (e.g. Gaussian distributed around a mean function), then even if you learned the perfect mean function you will still have an expected error. The entropy $H(p(\cdot|x))$ is the smallest expected neg-log-likeihood your model could possibly achieve. The cross-entropy of your learnt model will therefore always be larger than this entropy. In this view, the KL-divergence $D\big(p \,\big\|\, q_\theta\big)$ is equal to the cross-entropy of your model, but subtracting $H(p)$ as the baseline. It measures only your error above the minimal achievable error.

The above establishes the close relation between Machine Learning and compression or Minimal Description Length: When ML minimizes a cross-entropy or KL-divergence, then it also minimizes the expected code length when encoding p-samples (data) using the learned q-model.