An overview of the source coding theorem

Iain Murray

September 29, 2014

This note attempts to bring together some of the main ideas covered in the first four lectures. It doesn't contain everything we covered, but I hope it's a helpful overview for understanding the lecture notes and the textbook.

1 The ideas behind storage and source coding (lossless compression)

How many bits of storage do we need to represent an item? If items come from an alphabet $\mathcal{A}_X$, we assign integers to each possible item, and represent one as a binary integer using $\lceil \log_2 |\mathcal{A}_X| \rceil$ bits of storage space[1]. (See lecture 1.)

What if many of the possible items rarely occur? We could give some items shorter codes codes than others. It would make sense to give the most probable items the shortest codes.

For theoretical purposes we consider the simplest possible schemes[2]. We identify a short-list of special items T, and give them short codes. If we ignore the remaining items, we just need $\lceil \log_2 |T| \rceil$ bits to represent an item from the short-list. We can deal with the remaining items by using $C_1 = \lceil \log_2(|T| + 1) \rceil$ bits to represent either one of the items from the short list or a flag that we will then use C_2 more bits to encode an item that's not in our special set T.

C_2 is some number of bits sufficient to uniquely identify an item not in the short-list T. Using $C_2 = \log_2 |\mathcal{A}_X|$ bits for a naive encoding would be more than enough.

Encoding an outcome x requires C_1 bits if $x \in T$, otherwise $C_1 + C_2$ bits. In expectation:

$$\mathbb{E}[\text{length}] = C_1 + P(x \notin T)\, C_2.$$

In what follows, we'll choose large sets T containing all 'typical' outcomes. Then $P(x \notin T)$ is negligible, so the expected length of the encoding is dominated by $\log_2 |T|$. (The '$+1$' above is also negligible if T is large.)

2 Background knowledge: probability and information

You need to know the stuff on the expectations sheet and sums of variables sheet. And know the probability basics and notation from p22–24 of MacKay.

You need to know how logarithms work. For logarithms of any base: $\log a^b = b \log a$, and $\log ab = \log a + \log b$. For a logarithm of base a, $\log_a a = 1$, so $\log_2 2^x = x$ and $\ln e^y = y$. In this course, $\log$ means $\log_2$, whereas in most maths books $\log$ means $\log_e$ also called $\ln$.

Information content of an outcome: $h(x) = -\log P(x) = \log 1/P(x)$. Measured in bits for $\log_2$ or nats for $\log_e$ (or bans for $\log_{10}$).

Information content is itself a random variable. If you generate an outcome from an ensemble, look up its probability, and record the corresponding $h(x)$, that value is a random number. If you repeat the process you'll get a different number.

Entropy is the average information content. Probability notation is often heavily overloaded, and not standardized. Some of the forms you'll see:

$$H(X) = \mathbb{E}[h(x)] = -\sum_i P(x{=}a_i) \log P(x{=}a_i) = -\sum_x P(x) \log P(x) = -\sum_i p_i \log p_i, \ldots$$

1. Notation: $\lceil x \rceil = \text{ceil}(x)$ is 'x rounded up to the next biggest integer'. $|\mathcal{A}_X|$ is the number of elements in set $\mathcal{A}_X$.
2. Practical coding schemes (week 3 and 4 notes) have more sophisticated variable-length encoding schemes. Such refinements are unnecessary for the asymptotics considered by the theory.

3 The source coding theorem

MacKay pp66–73 gives more detail for the intuitions behind information content. Intuition is important: it will help you apply this stuff, get answers more quickly, and avoid mistakes. For now, to follow the proof, you just need to know $h(x) = -\log P(x)$, and $H(X) = \mathbb{E}[h(x)]$. For brevity we write H instead of $H(X)$.

3.1 Extended ensembles

An ensemble (p22 MacKay), X, defines the possible outcomes we are interested in, and their probabilities. An *extended ensemble*, X^N, defines all possible blocks of N independent outcomes, each drawn from X.

Motivation 1: If X defines a $\mathrm{Bernoulli}(p\!=\!0.1)$ distribution, the possible outcomes are `0` and `1`. These outcomes can't be compressed. The two best possible codes are: 1) represent `0` with `0`, and `1` with `1`, and 2) represent `0` with `1`, and `1` with `0`. However, outcomes from *extended ensembles* can be compressed. The lecture notes contain an explicit example for $N\!=\!5$.

Motivation 2: We are interested in general ensembles, with arbitrary alphabets and probability distributions. A single outcome might be a file representing a document, an image, sound, or anything. By considering large bags of N outcomes, the details of the ensemble wash away due to the law of large numbers. It turns out we only need to know the ensemble's entropy.

We'll call the block of N outcomes from the extended ensemble $\mathbf{x} = [x_1, x_2, \ldots, x_n, \ldots, x_N]$.

The information content of a block is $h(\mathbf{x}) = \sum_{n=1}^{N} h(x_n)$. (Why?)

The average information content of a block is $\mathbb{E}[h(\mathbf{x})] = NH$. (Why?)

The variance of the information content is $N\sigma^2$, where in this document σ^2 is the variance of the information content under ensemble X. In principle we could compute the actual value of $\sigma^2 = \mathbb{E}[(h(x) - H)^2]$ for a given ensemble, but we won't need to here.

3.2 The typical set

We imagine listing all possible extended outcomes $\mathbf{x}$ in order of probability. (As we did for $N\!=\!5$ Bernoulli outcomes in class.) This list is in reverse order of information content.

The information content is the sum of a large number of independent items so, by the law of large numbers, it will usually be close to the mean, NH. That is, we'll usually see items in our list around those with information content NH, plus or minus a few standard deviations. Example: if our ensemble is a $\mathrm{Bernoulli}(p = 0.1)$ ensemble, we'll usually see blocks with close to 10% `1`'s, and 90% `0`'s. Or equivalently, blocks with information content close to $NH_2(p) = -Np\log p - N(1\!-\!p)\log(1\!-\!p)$.

The typical set containing all blocks with information contents within plus or minus m standard deviations of the mean is:

$$T_m = \left\{\mathbf{x} : h(\mathbf{x}) \in [NH\!-\!m\sqrt{N}\sigma,\; NH\!+\!m\sqrt{N}\sigma]\right\}.$$

Because $h = -\log p$, the bags of N items with smallest probability in this set have total information content $NH\!+\!m\sqrt{N}$, and probability:

$$p_{\min} = 2^{-NH - m\sqrt{N}\sigma}.$$

Similarly, the bags of N items with the largest probability in this set have probability:

$$p_{\max} = 2^{-NH + m\sqrt{N}\sigma}.$$

There are possible outcomes, outside the typical set, that have probability much larger than $p_{\max}$. In our Bernoulli example, the all-zero block `000...000` is much more probable than any individual typical block with around 10% `1`'s. However, the total probability mass associated with these most probable items is small, because (relatively speaking) there aren't many of them. Excluding these top-most probable items from T_m, substantially reducing $p_{\max}$, helps the second part of the proof go through.

The larger we pick m, the more probable it is that blocks will fall in the typical set T_m. Chebyshev's inequality gives us a bound (which is often very loose, but true for *all* ensembles):

$$P(\mathbf{x} \notin T_m) \le \frac{1}{m^2}.$$

We set m large enough that the encoding cost will be dominated by $\log |T_m|$ (see Section 1). Now we just need to find (well, bound) the size of the typical set.

3.3 Upper-bounding the number of bits required

If all the possible outcomes in an ensemble have probability $1/5$, you know there are 5 possible outcomes. If the *minimum* probability of outcomes is $1/5$, you know there are *at most* 5 possible outcomes. Similarly

$$|T_m| \le \frac{1}{p_{\min}}, \quad \Rightarrow \quad |T_m| \le 2^{NH + m\sqrt{N}\sigma}.$$

Therefore we can encode blocks in the typical set with $NH + m\sqrt{N}\sigma$ bits. For large m we can encode any block with expected length close to this value, because we'll rarely see anything outside the typical set. The expected length per outcome from the original ensemble is

$$H + m\sigma/\sqrt{N}\ \text{bits/symbol}, \rightarrow H\ \text{bits/symbol as}\ N \rightarrow \infty.$$

As long as we're encoding sufficiently large bags of symbols at once, we can compress them to H bits/symbol on average.

3.4 Lower-bounding the number of bits required

Could we do better? Maybe we could identify a set S, smaller than the typical set, and give those items shorter codes? Let

$$|S| = 2^{NH(1-\epsilon)}, \quad \epsilon \in [0, 1].$$

For example, if $\epsilon = 0.01$, we'd be asking for a set that we could index with 1% fewer bits than using the typical set. If blocks fall in this set frequently enough, we'd notice a small increase in compression.

A block in the new set might also be in the typical set ($\mathbf{x} \in T_m$), otherwise it will be outside the typical set ($\mathbf{x} \in \bar{T}_m$). We split the probability that a block falls in our new set into those two possibilities:

$$P(\mathbf{x} \in S) = P(\mathbf{x} \in S \cap T_m) + P(\mathbf{x} \in S \cap \bar{T}_m).$$

For large m, we've already established that the probability of falling outside the typical set is negligible, so we can ignore the second term. We now bound the first term. There are at most $|S|$ elements in $S \cap T_m$, and their probability is at most $p_{\max}$ each, therefore:

$$\begin{aligned} P(\mathbf{x} \in S \cap T_m) &\le |S| \cdot p_{\max} \\ &\le 2^{NH(1-\epsilon)} \cdot 2^{-NH + m\sqrt{N}\sigma} \\ &\le 2^{N(-\epsilon H + m\sigma/\sqrt{N})} \\ \Rightarrow P(\mathbf{x} \in S \cap T_m) &\rightarrow 0, \ \text{as}\ N \rightarrow \infty. \end{aligned}$$

In other words, any set S significantly smaller than the typical set is useless. For large extended ensembles, we'll never see a block land in S. It doesn't matter if we give those items shorter encodings, the effect on the expected coding length is negligible.

Therefore, the expected encoding length for large extended ensembles cannot be less than H bits/symbol. Our original ensemble, and small extended ensembles must also have this limit: otherwise we could just split our large extended ensemble into smaller parts, and then encode it with fewer than H bits/symbol.

Many outcomes can be compressed together in H bits/symbol on average, but no less.