COMS-E6998: Information Theory in Computer Science **Fall 2017**

Lecture 1: Entropy and Noiseless Compression

Lecturer: Omri Weinstein *Scribes: Rex Lei*

This lecture is an introduction to the most fundamental concept(s) in Information Theory and to its most classical application – *data compression*. This lecture focuses entropy and its properties, and concludes with a brief introduction to next lecture's discussion on prefix-free codes and Kraft's inequality.

The first part of the lecture was a review of the course syllabus and class logistics. The information is available here: `http://infotheorytcs.wikischolars.columbia.edu/`

As will become clear throughout the entire course, we shall see why information theory is the "right" language (and clean calculus) to reason about communication problems and other computational models (when it comes to proving LBs). In his classical 1948 paper, Shannon was interested in understanding the *one-way data transmission* problem: Suppose Alice has a random message (say, picture) $X \sim \mu$ in mind, and she wishes to describe it to Bob who (for now) has no extra information beyond the distribution μ.

1.1 Entropy

Consider the random variable[1]

$$X \sim \begin{cases} a, & \text{w.p } \frac{2}{3} \\ b, & \text{w.p } \frac{1}{6} \\ c, & \text{w.p } \frac{1}{6} \end{cases} \tag{1.1}$$

Intuitively, we'd like the *entropy* of X, denoted $H(X)$, to measure the amount of "information conveyed by the value of the random variable X", where the amount of information is, roughly speaking, the amount of "surprise" we get from observing a sample $X \sim \mu$. [2]

Suppose we had such a function $g : [0, 1] \mapsto \mathbb{R}^+$ that takes a probability and maps it to a non-negative real. Let's think about some natural properties of g. What if someone told you that the value of X is a, b, or c. Are you surprised? Of course not – because this occurs with probability 1 and we already know this. Therefore, we'd like $g(1) = 0$: flipping a two-headed coin doesn't provide any surprise.

Now suppose you were told that $X = a$. This is a little bit surprising; however, $X = b$ is more surprising because this is a rarer event. So, when $p < q$, we'd like S to satisfy $g(p) > g(q)$. We'd also like $g(x)$ to be a continuous function of x, since we'd like the surprise to be smooth (perturbing X by ε shouldn't make much difference).

Third, it's natural to require our surprise to be *additive* under *independent* samples – e.g., that the surprise from observing the events "$X_1 = a$", "$X_2 = b$" will be additive when X_1, X_2 are sampled independently (here we think of X_1, X_2 as indicator r.v's of arbitrary events, e.g., $X_1 = a, X_2 = b$, which allows to define entropy only w.r.t *binary* r.v's). So we want $g(p, q) = g(p) + g(q)$.

In summary, we have the following axioms for our entropy function g:

[1] We shall distinguish between the r.v X and its underlying distribution μ – whenever clear from context, we will use $H(X)$ or $H_\mu(X)$ to mean $H(\mu)$ (indeed, entropy is a measure on distributions, not on the random variable itself, as it completely discards the values of the support of X).

[2] Remember, surprise is always measured with respect to what we expect to happen, i.e. wrt our prior μ.

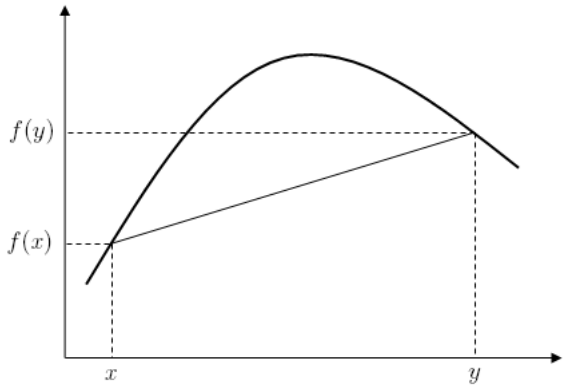

Figure 1.1: The depicted function is a *concave function*. The value $\frac{f(x)+f(y)}{2}$ is the height of the chord at the midpoint of x and y. This is less than the value $f\left(\frac{x+y}{2}\right)$, the height of f at the midpoint of x and y. Note: the function $\log(x)$ is concave. Image from `https://en.wikipedia.org/wiki/File:ConcaveDef.png`.

1. $g(1) = 0$
2. $g(p) > g(q)$ for probabilities p, q such that $p < q$. Furthermore, g is continuous.
3. $g(p, q) = g(p) + g(q)$.

Exercise 1.1.1. *Show that the only function satisfying all 3 axioms, up to normalization, is* $g(p) := c \lg_2 1/p$.

A convenient normalization is $g(1/2) = 1$ ("max surprise"), which fixed c to 1. To see why probability $1/2$ should be the max surprise, consider the following example: Say we have a coin that comes up heads with probability p and tails with probability $1 - p$. If $p = .01$, the coin almost always comes up tails, so the coin is not very surprising – without looking at the result of a flip, we can confidently guess that it is a tails. However, if $p = 1/2$, then it is very difficult to guess the outcome of a single flip, and finding out if the coin landed heads or tails is very "surprising".

Definition 1.1.2 (Entropy)**.** *The (Shannon) entropy of random variable* $X \sim \mu$ *is defined as*

$$H_\mu(X) := \sum_{x \in Supp(X)} \mu(x) \lg \frac{1}{\mu(x)} = \mathbb{E}_{x \sim \mu}\left[\lg \frac{1}{\mu(x)}\right],$$

where we define $0 \lg \frac{1}{0} := 0$.

For example, the above distribution has entropy $(1/3) \lg 3 + 2((1/6) \lg 6) \approx 1.38$.

Some facts and properties:

- Note that the definition depends entirely on the uncertainty of the distribution, not the *values* (i.e. support) that X takes. Indeed, what matters to "unpredictability" are not the values themselves (a random variable with renamed values will be as predictable as its former).
- $0 \leq H(X) \leq \lg |Supp(X)|$, with LHS equality iff X is deterministic and RHS equality iff X is uniform. (proof LHS : $0 \leq -\lg(x)$ for $x \in [0, 1]$. proof RHS: Jensen's inequality)

  A *concave* function is a function that satisfies $\frac{f(x)+f(y)}{2} \leq f\left(\frac{x+y}{2}\right)$ for all x, y. (See Figure 1.1.) Jensen's inequality states that for all concave functions, $f(\mathbb{E}[X]) \geq \mathbb{E}[f(X)]$. (The inequality sign is reversed for convex functions.)

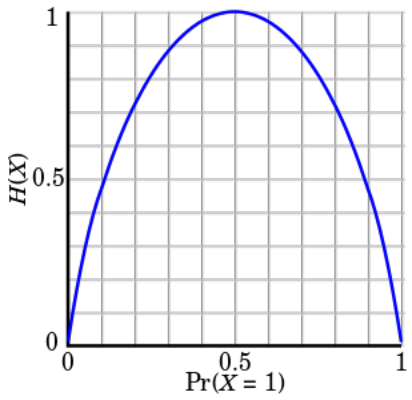

Figure 1.2: Plot of Binary Entropy Function, image from `http://en.wikipedia.org/wiki/Binary_entropy_function#/media/File:Binary_entropy_plot.svg`

Proof. (RHS) Given a random variable $X \sim \mu$, define random variable Y so that $Y \sim 1/\mu_i$ with probability μ_i.

$$\begin{aligned} H_\mu(X) &= \sum_{\mu_i} \lg \frac{1}{\mu_i} = \mathbb{E}_Y[\lg Y] \\ &\leq \lg \mathbb{E}_Y[Y] = \lg \sum_i \mu_i \cdot \frac{1}{\mu_i} = \lg \sum_i 1 = \lg |Supp(X)| \end{aligned} \tag{1.2}$$

□

- $H(p) \approx p \lg 1/p + o(p) = \Theta(p \lg 1/p)$ for small p (Taylor expansion, direct calculus).
- $H(p)$ has has heavy ("quadratic") tails. For example, binary entropy $H(0.9) = 0.4689 \approx 1 - (0.4^2)/2$. (See Figure 1.2.) In general, $H(1/2 + \varepsilon) \approx 1 - \varepsilon^2/8$.
- For the purposes of this class, we will almost exclusively discuss entropy with log base 2.
- **Caution.** We'll make an important distinction b/w r.v's and events.

Operational interpretation of Entropy. Let us now try to derive the entropy function of X in an "operational" manner (this will also give a more "combinatorial" interpretation of entropy).

The question Shannon asked himself: **Lossless compression / source coding problem :** Suppose Alice receives $X_1, \ldots, X_n \sim \mu^n$ and wishes to transmit it to Bob (who knows nothing but the prior μ). What, asymptotically, is the *amortized* amount of communication required to transmit this sequence?

For simplicity, we deal with the *binary* case: Suppose Alice wants to transmit $\overline{X} := X_1, \ldots, X_n \sim Ber(p)^n$ to Bob. How would you convey this information? A natural way to communicate $\overline{X}$ is to first send the number of 1s in the string, k, followed by $\lceil \lg \binom{n}{k} \rceil$ bits which specify the actual indices. This would cost (in expectation over X_i's) at most

$$\lceil \lg n \rceil + \sum_k \binom{n}{k} p^k (1-p)^{n-k} \left\lceil \lg \binom{n}{k} \right\rceil$$

When $\lim_{n\to\infty}$, the average number of bits can be shown (using Sterling's approximation) to be precisely $H(X)$. An even more intuitive way of saying this is using the *equipartition* property of typical sequences: We expect pn "1"s in $\overline{X}$, and in fact, by standard concentration bounds, $\overline{X}$ will be essentially *uniformly* distributed among all $\binom{n}{pn}$ stings of Hamming weight $pn + o(pn)$. So, we can ignore cases in which Alice's string has a Hamming weight much larger or smaller than pn.

How many such strings are there in $B_{pn}(0)$? Answer: $\lim_{n\to\infty} \frac{|B_{pn}(0)|}{n} = H(p) \approx O(p \lg 1/p)$.

Therefore, not only is entropy a mathematical function, but it also has an operational meaning as the amortized communication cost for communicating a sample taken from a binary distribution.

The non-binary case. It is easy extend the above to arbitrary alphabets/supports using just the the binary entropy definition. Indeed, we can "break" an arbitrary r.v. $X \in \{a, b, \ldots, z\}$ into indicator r.v's $W_1 := \mathbf{1}_{X=a}$ etc (and normalize). Then $H(X) = H(W_1) + \Pr[W_1 = 0] \cdot H(W_2)$ and we can continue by induction, using the fact that we can compute the *binary entropy*.

Conclusion. Asymptotically,

$$H_\mu(X) = \lim_{n\to\infty} \frac{C(X^n)}{n}$$

where $C(X^n)$ is the average cost (number of bits) of a 1-1 map between X^n and its transmitted representation.

One-shot Compression. We saw that entropy captures communication cost in an *asymptotic* sense. We will soon see that a much stronger claim is true – it captures even a *"single-shot"* transmission, as long as we're willing to measure things in *expectation*. That is, even if Alice gets a single sample $X \sim \mu$, she can spend at most $H_\mu(X) + 1$ bits and this is essentially optimal.

1.2 Lossless compression, prefix-free codes, and Kraft's inequality

We now introduce the concept of a *prefix-free code* or *instantaneous* code. The simple rule for such a code is that no codeword is a prefix of another codeword. Suppose $X \in \{a, b, c, \ldots, z\}$. We can communicate this with $\lceil \lg_2 26 \rceil = 5$ bits or use ASCII which would entail 7 bits. But suppose the distribution of X is very far from uniform – in this case we might be able to leverage that for a more efficient bit representation.

Suppose we need to encode A,B, and C. Say we have A map to 0, B maps to 1, and C map with 10. When we go to decode our message and see a 10, we cannot distinguish between BA and C. However, if no codeword is a prefix of another, there is never this ambiguity and you can decode as soon as your buffer matches a codeword. A potential prefix-free code for A, B, and C is $A \mapsto 0$, $B \mapsto 10$ and $C \mapsto 11$. (See figure 1.3.)

Our question now becomes: what are the lengths of codewords for which prefix-free codes exist? A prefix-free code is nothing but a *tree* where the letters sit at the leaves.

A naive approach would be to assign the same depth to each character/message. However, this could be wasteful. Suppose that $\Pr[X = a_i] = \mu_i$ for $i \in [n]$. Our goal is to minimize the *expected depth* of the tree, i.e., to minimize $\sum_i \mu_i \ell_i$ (where ℓ_i is the length of the root-to-leaf path of the encoding of a_i).

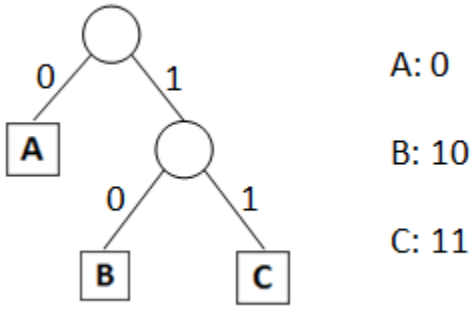

Figure 1.3: A prefix-free code for A, B, C. Image from `http://i.cs.hku.hk/~hkual/Notes/Greedy/HuffmanCoding.html`

A greedy approach to build a prefix code (tree) for X is to merge, in each iteration, the *least* likely two letters and put them in the bottom of the tree, then recurse on a support of size $n-1$ (draw figure). This is the famous *Huffman code*.

Fact 1.2.1. *The Huffman code is the minimizer of the expected depth $\sum_i \mu_i \ell_i$ among all (prefix) codes.*

Interestingly, the proof of optimality is a local one – one can show that any prefix code can be locally modified so that the two least likely symbols x, y differ only in their last bit of encoding, and this local swap can only decrease the encoding length. Note that the proof does not explicitly provide an analysis to the absolute expected code length. [3]

How does one prove the optimality of Huffman codes? In the next lecture, we will discuss *Kraft's inequality*.

Lemma 1.2.2 (Kraft's inequality)**.** *A prefix-free code*[4] *for $X \in_R [n]$ over alphabet Σ, with lengths $\ell_1, \ldots, \ell_n$ exists if and only if*

$$\sum_{i=1}^{n} |\Sigma|^{-\ell_i} \leq 1.$$

We will see a proof and continue this discussion in the next lecture.

References

Figure 1.1 from `https://en.wikipedia.org/wiki/File:ConcaveDef.png`

Figure 1.2 from `https://en.wikipedia.org/wiki/Binary_entropy_function#/media/File:Binary_entropy_plot.svg`

Figure 1.3 from `http://i.cs.hku.hk/~hkual/Notes/Greedy/HuffmanCoding.html`

[3] Huffman codes require storing the entire prefix tree in memory, and they have construction (and decoding) time which is polynomial in $|Supp(X)|$ which may be very large in case of product distributions ($X \sim \mu^k$). A more efficient and practical solution that addresses this drawback is *arithmetic codes*, which build the codeword "on the fly" and have very simple and efficient decoding time. The caveat is that they are only *asymptotically* optimal, i.e., they work for encoding a long stream ("block") of i.i.d smiles of X. We will discuss this difference many times in this course. This type of question is brings us to the realm of *algorithmic information theory*. we also note, as a curiosity, that a prefix-free code provides a solution to the "21 questions" game, and in particular, Huffman code is an optimal strategy for this game.

[4] In fact, Kraft's inequality applies to any *uniquely decodable code*, which has the property that for every n, if it is applied consecutively on n symbols, then the *concatenation* of compressed versions must still be decodable.