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

Lecture 2: Entropy and Noiseless Compression

Lecturer: Omri Weinstein *Scribes: scribe-name1,2,3*

This lecture is an introduction to the most fundamental concept(s) in Information Theory and to its most classical application – *data compression*. As will become clear starting this lecture and throughout this entire course, we shall see why IT 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.

2.1 Entropy

Consider the random variable[1]

$$X \sim \begin{cases} a, & \text{w.p } \frac{2}{3} \\ b, & \text{w.p } \frac{1}{6} \\ c, & \text{w.p } \frac{1}{6} \end{cases} \tag{2.1}$$

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 this random variable (remember, surprise is always measured against what we expect to happen).

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. Suppose we told you that X is a, b, or c. Are you surprised ? Of course not – because this occurs with probability 1 and we already know this. So therefore wed like $g(1) = 0$: flipping a two-headed coin doesn't provide any surprise. Now suppose we told you that $X = a$. This is a little bit surprising, but $X = b$ is more surprising because this is a rarer event. So wed like S to satisfy $g(p) > g(q)$ if $p < q$. Wed 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 $g(p, q) = g(p) + g(q)$.

Exercise 2.1.1. *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*).*

Definition 2.1.2 (Entropy)**.** *The (Shannon) entropy of* $X \sim \mu$ *is defined as*

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

[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).

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

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

Some facts and properties:

- Note that the definition is completely oblivious to the *values* (i.e., support) that X takes. Indeed, what matters to "unpredictability" are not the values themselves (a 1-1 transformation 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 : $0 \leq -\lg(x) \leq 1$ for $x \in [0,1]$. proof RHS: Jensen's inequality [TODO: explain + draw convex/concave function]).
- $H(p) \approx 2p \lg 1/p + o(p) = \Theta(p \lg 1/p)$ (Taylor expansion, direct calculus).
- $H(p)$ has has heavy ("quadratic") tails. For example, binary entropy $H(0.9) = 0.4689 \approx 1 - (0.4^2)/2$ (draw graph). In general, $h(1/2 + \varepsilon) \approx 1 - \varepsilon^2/8$.
- **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 is the *amortized* amount of communication required to transmit this sequence (asymptotically)?

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. 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 starlings 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 $pn + o(pn)$. How many such strings are there in $B_{pn}(0)$? Answer: $\lim_{n\to\infty} |B_{pn}(0)| = 2^{H(p)\cdot n} \approx 2^{n \cdot p \lg 1/p}$.

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.*

We saw that entropy captures communication cost in an *asymptotic* sense. We will now 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.*

2.2 Lossless compression, Kraft's inequality and Shannon's noiseless coding theorem

Suppose $X \in \{a, b, c, \dots, z\}$. 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.

We introduce the concept of a *prefix-free code* or *instantaneous* code (we might mention later why this is essentially without loss of generality for coding purposes). The simple rule for such a code is that no codeword is a prefix of another codeword. Suppose we need to encode A,B, and C. We cant say A maps to 0 and B maps to 1 because then C can neither start with 0 or 1. This also makes decoding easy, because you can decode as soon as your buffer matches a codeword (it cant be a prefix of another code). A potential prefix-free code for A, B, and C is $A \mapsto 0$, $B \mapsto 10$ and $C \mapsto 11$. 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 (Draw tree figure).

Now 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). A greedy approach to built 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 2.2.1. *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.

Remark. Notwithstanding, 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.

How does one prove the optimality of Huffman codes?

Lemma 2.2.2 (Kraft's inequality). *A prefix-free code*[2] *for $X \in_R [n]$ over alphabet Σ, with lengths $\ell_1, \dots, \ell_n$ exists if and only if*

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

Proof. ($\rightarrow$) Take a random walk on the prefix tree from the root, until you either reach a leaf (symbol) or fall off the tree. It is easy to see that

[2] 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.

$$1 \geq \Pr[\text{Reach a leaf}] = \sum_{i=1}^{n} \Pr[\text{Reach leaf } i] = \sum_{i=1}^{n} |\Sigma|^{-\ell_i}.$$

where the equality follows since this is a union of disjoint events (only 1 leaf can be hit at termination).

($\Leftarrow$) Consider the case $\Sigma = 2$. Without loss of generality, we have $\ell_1 \leq \dots \ell_n := L$ where L is the depth of the tree. Since there is a codeword at ℓ_1, that eliminates $2^{L-\ell_1}$ possible leaves from the full binary tree. Since the second codeword *cannot have a prefix of the first codeword*, we can argue the same for ℓ_2 (Draw figure). Straightforward calculation shows that we can keep going until we have "eaten up" the entire 2^L leaves of the full binary tree, i.e., so long as $\sum_i 2^{L-\ell_i} \leq 2^L$.

□

2.2.1 The Shannon-Fano code

Kraft's inequality tells us that in order to build a good (prefix) code for compressing $X \in_R [n]$, it's enough to come up with a sequence of integers $\{\ell_i\}_{i=1}^n$ that satisfy Kraft's inequality inequality. Indeed, Shannon observed that the function

$$\ell_i := \left\lceil \lg \frac{1}{\mu_i} \right\rceil.$$

It is immediate to see that indeed $\sum_i 2^{-\ell_i} \leq \sum_i 2^{-\lg_2 \frac{1}{\mu_i}} = \sum_i \mu_i = 1$, hence a corresponding prefix code exists. This is the *Shannon-Fano* code. The expected length of this code is

$$H(X) \leq \sum_i \mu_i \left\lceil \lg \frac{1}{\mu_i} \right\rceil \leq \sum_i \mu_i (\lg \frac{1}{\mu_i} + 1) = H(X) + 1.$$

In some cases,the Shannon code does not perform optimally. Consider a Bernoulli random variable X with parameter 0.0001. An optimal encoding requires only one bit to encode the value of X. The Shannon code would encode 0 by 1 bit and encode 1 by log 104 bits. This is good on average but bad in the worst case. We can also compare the Shannon code to the Huffman code. The Huffman code always has shorter expected length, but there are examples for which a single value is encoded with more bits by a Huffman code than it is by a Shannon code. Consider a random variable X that takes values a, b, c, and d with probabilities 1/3, 1/3, 1/4, and 1/12, respectively. A Shannon code would encode a, b, c, and d with 2, 2, 2, and 4 bits, respectively. On the other hand, there is an optimal Huffman code encoding a, b, c, and d with 1, 2, 3, and 3 bits respectively. Note that c is encoded with more bits in the Huffman code than it is in the Shannon code, but the Huffman code has shorter expected length. Also note that the optimal code is not unique: We could also encode all values with 2 bits to get the same expected length.

Theorem 2.2.3. *The expected length of any prefix code for X is at least $H(X)$.*

Proof. We will show that $H(X) - \sum_i \mu_i \ell_i < 0$. To this end, we have

$$\begin{aligned} H(X) - \sum_i \mu_i \ell_i &= \sum_i \mu_i \lg \frac{1}{\mu_i 2^{\ell_i}} \\ &\leq \lg \left(\sum_i \mu_i \cdot \frac{1}{\mu_i 2^{\ell_i}} \right) \qquad \text{(By Jensen's inequality)} \\ &\leq \lg(1) = 0 \qquad \text{(By Kraft's inequality)} \end{aligned}$$

□

Recall that this direction of Kraft's inequality applies even to non-prefix codes, hence entropy is a lower bound on the expected length of any source code for X.

Shannon's noiseless coding theorem. If we're willing to believe that entropy is additive over *independent* random variables ($H(X^n) = nH(X)$, a fact we shall soon prove formally), then the Shannon-Fano "single-shot" compression result implies immediately that

$$\lim_{n\to\infty} \frac{C(X^n)}{n} \leq \lim_{n\to\infty} \frac{H(X^n)+1}{n} = \lim_{n\to\infty} \frac{nH(X)+1}{n} = H(X).$$

This is Shannon's noiseless coding theorem, one of the cornerstones of information theory. Note that we could prove this theorem more simply using the equipartition property of entropy (typical sequences) Chernoff bound, but the latter proof is more instructive for our future purposes – we showed an *amortized/asymptotic* compression result via *single-shot* compression. When we get to the *interactive* communication setup, this approach will be very useful.

References