1 Class Administrivia

15-859 "ITCS" is Information Theory + Application in TCS. The instructors are Mahdi Cheraghchi and Venkat Guruswami. The requirements for the course are:

- 3-5 problem sets (every 2-3 weeks, "on the hardish side")
- Some amount of scribe notes, depending on how many people are actually enrolled

The impetus for this class is Shannon's seminal 1948 paper and the increasing role of information-theoretic mindset and tools in the theory of computing.

Essentially, information theory gives a clean calculus in various settings, especially when it comes to offerings of lower bounds.

The text is Cover and Thomas's *Elements of Information Theory*, but it's **not** required. Hopefully the scribe notes will be very good so those will offer a good reference.

The prerequisite for the class is to be comfortable with discrete probability and random variables.

2 Entropy

Let X be a discrete random variable, say,

$$X \sim \begin{cases} a & \frac{2}{3} \\ b & \frac{1}{6} \\ c & \frac{1}{6} \end{cases}$$

and define $H(X)$ to be the entropy of the random variable X. We'd like the entropy 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.

Suppose we had such a function $S : [0, 1] \to \mathbb{R}^+$ that takes a probability and maps it to a non-negative real. Let's think about some natural properties of S.

Suppose we told you that X is a, b, or c. Are you surprised? No—because this occurs with probability 1 and we already know this. So therefore we'd like $S(1) = 0$: flipping a two-headed coin doesn't give us 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 we'd like S to satisfy $S(p) > S(q)$ if $p < q$.

We'd also like $S(x)$ to be a continuous function of x, because we'd like the surprise to not exhibit jumps (if we had instead

$$X \sim \begin{cases} a & \frac{2}{3} + 10^{-6} \\ b & \frac{1}{6} - 10^{-6} \\ c & \frac{1}{6} \end{cases}$$

our impression of the "surprise" of a should not be much different than the original X).

Now suppose X_1 and X_2 are two independent instantiations of X. It's natural for our surprise from $X_1 = a$ and $X_2 = b$ to be additive (as each event "adds" to our surprise):

$$S\left(\frac{2}{3}\frac{1}{3}\right) = S\left(\frac{2}{3}\right) + S\left(\frac{1}{3}\right)$$

which means we must have $S(pq) = S(p) + S(q)$.

Exercise. *The only S that satisfies the above requirements is $S(p) = c \log_2 \left(\frac{1}{p}\right), c > 0$. A convenient normalization constant is that we are unit surprised by a fair coin flip ($S(1/2) = 1$), which sets $c = 1$.*

Definition 1 (Entropy)**.** *The entropy of a discrete random variable X, denoted $H(X)$, is the average surprise for an outcome of X:*

$$\mathop{\mathbb{E}}_{x}\left(\log_2 \frac{1}{P(x=x)}\right) = \sum_{x\in \operatorname{supp}(X)} p(x)\log_2 \frac{1}{p(x)},$$

where we define $0\log_2 \frac{1}{0} = 0$.

It is simple to derive the following facts and examples:

Fact. $H(X) \geq 0; H(X) = 0$ *if and only if X is deterministic.*

Example. *Let* $X \sim \begin{cases} a & \frac{2}{3} \\ b & \frac{1}{6} \\ c & \frac{1}{6} \end{cases}$ *as before. Then* $H(X) = \frac{2}{3}\log_2\frac{3}{2} + \frac{1}{6}\log_2 6 + \frac{1}{6}\log_2 6$.

Example. *Let X be uniform on $\{a_1,\dots,a_n\}$. Then $H(X) = \log_2 n$.*

Fact. *If $|\operatorname{supp}(x)| = n$, then $H(X) \leq \log_2 n$.*

Communication. Let's see if we can derive the binary entropy function in a different setting. Suppose $X \leftarrow \mathrm{Ber}(1/2)$, then $H(X) = 1$. And if $X \leftarrow \mathrm{Ber}(1)$, then $H(X) = 0$. Now suppose $X \leftarrow \mathrm{Ber}(p)$, and we have $X_1, X_2, \dots, X_n$ iid samples from this distribution, represented as a bitstring $\{0,1\}^n$. A way to communicate this in an efficient way is that we first send k, the number of 1 bits in the string, and then use $\lceil \log_2 \binom{n}{k} \rceil$ bits to reveal which subset those 1s go in. The expected number of bits communicated with this scheme is

$$\mathbb{E}(\text{number of bits}) = \underbrace{\lceil \log_2 n \rceil}_{\text{sending } k} + \underbrace{\sum_{k=0}^{n} \binom{n}{k} p^k (1-p)^{n-k} \left\lceil \log_2 \binom{n}{k} \right\rceil}_{\text{number of bits required for each } k}.$$

What we are interested is the average fraction of bits we need as the number of bits as n grows large: $\lim_{n\to\infty} \frac{1}{n}\mathbb{E}(\text{number of bits})$. Dividing our expression by n, we can omit the first term ($\log n/n \to 0$ as $n \to \infty$) and we can use Stirling's approximation (or the intuitive concept that if k is far from pn the term in the sum is close to zero) to obtain that

$$\lim_{n\to\infty} \frac{1}{n}\mathbb{E}(\text{number of bits}) = h(p) = p\log_2\frac{1}{p} + (1-p)\log_2\frac{1}{1-p},$$

which tells us that we can take $H(\mathrm{Ber}(p))$ bits, on average, to communicate a $\mathrm{Ber}(p)$ random variable.

Non-binary random variable. We can extend this definition of entropy in the communications context to non-binary random variables. Suppose $X \leftarrow \{a_1,\dots,a_n\}$. Suppose we have related variables

$$Z_1 = \begin{cases} 1 & X = a_1 \\ 0 & \text{else} \end{cases}$$

and

$$Z_2 = \begin{cases} a_2 & \frac{p_2}{1-p_1} \\ \vdots & \\ a_n & \frac{p_n}{1-p_1} \end{cases}.$$

We can break

$$H(X) = H(Z_1) + \Pr(Z_1 = 0)H(Z_2),$$

as we need to communicate Z_1 and then, if Z_1 is 0, we need to communicate Z_2. Since we know how to compute $H(Z_1)$ from the binary case, we can recursively use this to obtain $H(X) = \sum_{i=1}^{n} p_i \log 1/p_i$, which is the same expression as we got before.

Lemma 1. *$|\text{supp}(x)| = n$ implies that $H(X) \le \log_2 n$ (presented as a fact earlier).*

Proof. We use Jensen's inequality: for convex functions f, we have $\mathbb{E}[f(x)] \le f(\mathbb{E}[x])$ (this implies $\mathbb{E}[X^2] \ge \mathbb{E}[X]^2$ and $\mathbb{E}[\sqrt{X}] \le \sqrt{\mathbb{E}[X]}$.) Since we have

$$\begin{aligned} H(X) &= \mathop{\mathbb{E}}_{x \leftarrow X}\left[\log \frac{1}{p(x)}\right] \\ &\le \log\left(\mathop{\mathbb{E}}_{x \leftarrow X} \frac{1}{p(x)}\right) \end{aligned}$$

where we have applied Jensen's inequality to the expression with $f(x) = \log x$ while using the random variable $Z \sim 1/p(x)$ w.p. $p(x)$:

$$H(X) = \mathop{\mathbb{E}}_{Z}[\log Z].$$

□

3 Lossless Data Compression aka Source Coding

Suppose $X \in \{a, b, c, \ldots, z\}$. We can communicate this with $\lceil \log_2(26) \rceil = 5$ bits or use ASCII and use 7 bits. But suppose the distribution of X is not uniform. Then 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. 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 can't 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 can't 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? We can visualize a prefix-free code as a binary tree where the letters sit at the leaves (see figure 1)

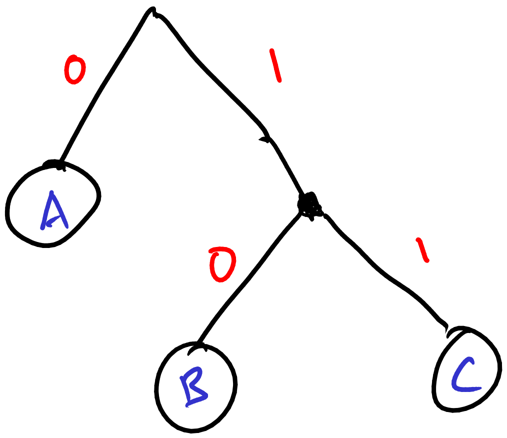

Figure 1: The tree for the example $A = 0$, $B = 10$, and $C = 11$

Now suppose we have a probability distribution on X:

$$X \leftarrow \begin{cases} a_1 & p_1 \\ a_2 & p_2 \\ \vdots & \vdots \\ a_n & p_n \end{cases}$$

and we can use a greedy construction (combine the least likely two letters and put them at the bottom of the tree, and then recurse on the $n-1$-size support, as seen in figure 2) to minimize $\sum_i p_i l_i$ where l_i is the length of the codeword that presents i. This construction is called *Huffman coding.*

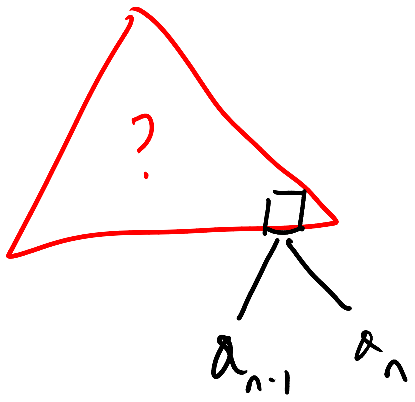

Figure 2: We combine the two least likely letters and recurse on the rest of the triangle

Fact. *Huffman codes minimize the expected depth $\sum_i p_i l_i$. However, they have time to construct that is polynomial in the size of the support of the underlying random variable, which may be very large in the case of product probability distributions.*

Fact (Kraft's Inequality)**.** *A prefix-free code for n symbols exists with lengths $\ell_1, \ell_2, \ldots, \ell_n$ if and only if*

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

where $|\Sigma|$ is the size of the alphabet for the code (2 for a binary alphabet, 3 if were coding over $\{0,1,2\}$, etc).

Proof (only if). This will proceed probabilistically. Take a random walk down the tree until you hit a leaf or fall off the tree. The

$$\begin{aligned} 1 &\ge \Pr(\text{we hit a leaf}) \\ &\overset{(a)}{=} \sum_{i=1}^{n} \Pr(\text{hit leaf } i) \\ &\overset{(b)}{=} \sum_{i=1}^{n} \frac{1}{2^{\ell_i}} \end{aligned}$$

where (a) follows because we can't hit multiple leaves and therefore the probability is over a disjoint union, and (b) follows because there is a unique path of length ℓ_i that leads to any particular leaf. This gives the result that we desire. □

Proof (if). Without loss of generality, we have $\ell_1 \le \ell_2 \le \ell_3 \ldots \le \ell_n = L$, where L is the total 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 can't have a prefix of the first codeword, we can argue the same for ℓ_2. This process is illustrated in 3. But if we have

$$\sum_{i=1}^{n} 2^{-\ell_i} \le 1$$

then we can't possibly "eat up" the entirety of the tree until the last step. □

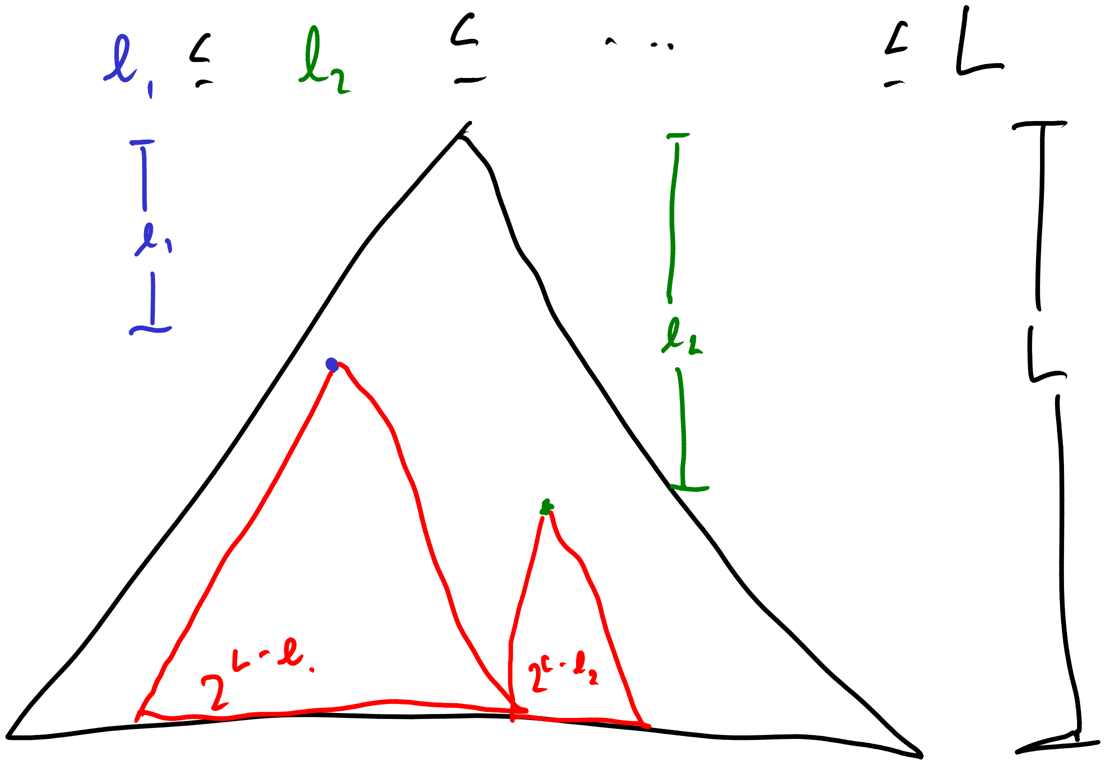

Figure 3: The codewords of length ℓ_1 and ℓ_2 are colored in their respective colors. The red triangles indicate the parts of the tree that have been cut. Due to the prefix-free nature of the code, they are non-overlapping.