Intro to Information Theory

Math Circle

February 11, 2018

1. **Random variables** Let us review discrete random variables and some notation. A random variable X takes value $a \in \mathcal{A}$ with probability $P(a) \geq 0$. Here the "sample space" $\mathcal{A}$ is a countable set (finite or infinite), and $\sum_{a \in \mathcal{A}} P(a) = 1$. An example of a random variable is a coin toss. The set of outcomes is $\mathcal{A} = \{\text{heads}, \text{tails}\}$ with $P(\text{heads}) = p$ and $P(\text{tails}) = 1 - p$ (of course, $0 \leq p \leq 1$).

   (a) **Independent random variables** Random variables are independent if the outcome of one does not affect the outcome of the other. If coin tosses X_1 and X_2 are independent, then

   $$P\{X_1 = H \textbf{ and } X_2 = H\} = P\{X_1 = H\} \times P\{X_2 = H\}$$

   For n independent, fair (probability of heads is 0.5) coin tosses:

   i. What is the probability that all tosses are tails?

   ii. What is the probability of any sequence of heads and tails?

   (b) **Uniform random variables** Let $\mathcal{A} = \{a, b, \ldots, z, 1, 2, 3, 4, 5, 6\}$ (the set of all lower case letters plus the first six digits). Let X be uniformly distributed over $\mathcal{A}$ so that the probability that X is any symbol in $\mathcal{A}$ is $\frac{1}{32}$. (Such a random variable could be helpful in generating a one-time pad for cryptography.)

   i. What is the probability that X is a letter?

   ii. We would like to generate X using independent fair coin tosses. That is, we would like to toss some coins and based on the outcome give a symbol from $\mathcal{A}$ such that the probability of each symbol is exactly $\frac{1}{32}$. How many indpendent fair coin tosses are needed to generate X?

(c) **Generating fair coins** Let's say we have an unfair coin with the probability of heads equal 0.8. We would like to use independent tosses of the unfair coin to generate independent tosses of a fair coin. We will use the following mapping:

Unfair coins	Probability	"Fair" coins
HHH	$0.8^3 = 0.512 \approx \frac{1}{2}$	H
HHT	$0.8^2 \times 0.2 = 0.128 \approx \frac{1}{8}$	THH
HTH	$0.8^2 \times 0.2 = 0.128 \approx \frac{1}{8}$	THT
THH	$0.8^2 \times 0.2 = 0.128 \approx \frac{1}{8}$	TTH
HTT	$0.8 \times 0.2^2 = 0.032 \approx \frac{1}{32}$	TTTHH
THT	$0.8 \times 0.2^2 = 0.032 \approx \frac{1}{32}$	TTTHT
TTH	$0.8 \times 0.2^2 = 0.032 \approx \frac{1}{32}$	TTTTH
TTT	$0.2^3 = 0.008 \approx \frac{1}{128}$	TTTTT

i. Assuming the approximations given in the probabilities are exact, what is the **only** sequence of generated fair coins that does not have the right probability?

ii. For every 3 unfair coins, what is the expected* number of fair** coins that are generated? (*: Expected number is the sum of the number of coins times the probability of that outcome. **: The generated coins are not exactly fair, but are pretty close.)

(d) **First definition of information** What is a good definition of the information contained in a random variable? We could start with the reasonable assertion that a fair coin has exactly 1 bit of information. We could then build on that to say that the information in a random variable is how many fair coins are needed to generate it or how many fair coins can be generated from it.

i. What is the information in the random variable uniformly distributed over 32 elements?

ii. What is the approximate information in an unfair coin with probability of heads equal to 0.8? Hint: What is the information is 3 independent unfair coins?

2. **Logarithms** In order to introduce a more formal definition of information, let us review the definition of base-b logarithms:

$$\log_b x = y \iff b^y = x.$$

Another way to rewrite this definition is $b^{\log_b x} = x$. There are many interesting properties of logarithms. Let us go over a few of them.

(a) What is $\log_2(2^n)$?

(b) What is $\log_2\left(\left(\frac{1}{2}\right)^n\right)$?

(c) **Logs of products** For $x > 0$, $y > 0$, $b > 0$,

$$\begin{aligned} b^{(\log_b(x)+\log_b(y))} &= b^{\log_b(x)} \times b^{\log_b(y)} \\ &= x \times y \\ &= b^{\log_b(x\times y)} \end{aligned}$$

i. Write $\log_b(x \times y)$ in terms of $\log_b(x)$ and $\log_b(y)$.

ii. If $\log_2(3) \approx 1.585$, what is $\log_2(72)$?

iii. Extra credit: Justify the steps in the equations above.

3. **Entropy** We now have the tools to define entropy of a random variable, which is one way of describing the information in that random variable. Entropy is a function of the probability distribution and not on the exact outcomes. The entropy of a random variable X is given by:

$$H(X) = \sum_{a \in \mathcal{A}} P(a) \log \left(\frac{1}{P(a)} \right)$$

(We assert that $0 \log \frac{1}{0} = 0$ since $p \log \frac{1}{p}$ approaches zero as p approaches 0.) Unless otherwise noted, we will use base-2 logarithm and the units of entropy is in binary bits.

(a) **Intuition: Entropy = Average Surprise** One way to think about entropy is the average "surprise" of a random variable. For two possible outcomes a and b, observing a is more suprising than observing b if $P(a) < P(b)$. For example, the Patriots were given a 60% chance to win the Super Bowl, while the Eagles had a 40% chance to win. The fact that the Eagles won was more suprising than a Patriots victory would have been.

i. Let us define the surprise of an outcome as $s(a) = \log \frac{1}{P(a)}$. Clearly $P(a) < P(b)$ implies $s(a) > s(b)$. Show that entropy is the average surprise.

ii. What if surprise was defined as $s'(a) = \frac{1}{P(a)}$? This also satisfies $P(a) < P(b)$ implies $s'(a) > s'(b)$. Compute the average surprise under this definition for any random variable. Why is this not a good definition?

(b) **Examples**

i. Let X be a fair coin toss, then

$$H(X) = P(\mathrm{H}) \log_2 \frac{1}{P(\mathrm{H})} + P(\mathrm{T}) \log_2 \frac{1}{P(\mathrm{T})} = \frac{1}{2} \log_2 2 + \frac{1}{2} \log_2 2 = 1 \,\text{bit}.$$

ii. Let X be uniform over $\mathcal{A} = \{a, b, \ldots, z, 1, 2, 3, 4, 5, 6\}$, then

$$H(X) = 32 \times \frac{1}{32} \log_2 32 = 5 \,\text{bits}$$

iii. Let X be an unfair coin toss with probability of heads equal to $\frac{4}{5}$, then

$$H(X) = \frac{4}{5} \log_2 \frac{5}{4} + \frac{1}{5} \log_2 5 \approx 0.722 \,\text{bits}.$$

Do these numbers agree with our first definition of information in Problem 1d.

(c) **Entropy of independent coin tosses** Let $X_1, X_2, \ldots, X_n$ be independent, fair coin tosses and let $Y = \{X_1, X_2, \ldots, X_n\}$ be the collection of the n coin tosses. What is the extropy of the collection of coin tosses, $H(Y)$? Hint: How many possible outcomes can Y take? What is the probability of each outcome? See Problem 1a.

(d) A Bernoulli random variable is just like a coin toss, but with heads replaced by 1 and tails replaced by 0. These are often used in statistics to describe the outcome of experiments, with $p = P(1)$ called the probability of success. If $X_1, \ldots, X_n$ are Bernoulli random variables describing experiments then $\sum X_i$ concisely gives the number of successful experiments.

Let X_1 and X_2 be independent Bernoulli($\frac{1}{2}$) random variables.

i. What is $H(X_1 + X_2)$?

ii. What is $H(X_1 + 2 \times X_2)$?

iii. What is $H(\{X_1, X_2\})$? See Problem 3c.

(e) Let X be a coin toss with probability of heads equal to p (or a Bernoulli(p) random variable). Graph the entropy $H(X)$ as a function of p. Hint: Compute $H(X)$ for p equal to 0, $\frac{1}{5}$, $\frac{1}{4}$, $\frac{1}{2}$, $\frac{3}{4}$, $\frac{4}{5}$, and 1 and draw a smooth curve between the points (again, use $\log_2(3) \approx 1.585$). The curve should be **concave** - a line connecting any two points in the curve should be below the curve.

(f) Let X_1 be a Bernoulli(p) random variable and let X_2 be a Bernoulli($1-p$) random variable. Let

$$X = \begin{cases} X_1, \text{with probability } \frac{1}{2} \\ X_2, \text{with probability } \frac{1}{2}. \end{cases}$$

i. What is the probability that X is 1?

ii. What is $H(X)$?

iii. How does $H(X)$ compare with $H(X_1)$ and $H(X_2)$?

(g) **Properties of Entropy** These are suggested by the problems above, but we won't prove them.

i. $0 \leq H(X) \leq \log |\mathcal{A}|$

ii. For independent random variables, $H(\{X_1, \ldots, X_n\}) = \sum H(X_i)$.

iii. For a deterministic function f, $H(f(X)) \leq H(X)$.

iv. Entropy is concave in the probability distribution. That is, if Z is distributed like X if a coin toss is heads and like Y if a coin toss is tails, then $H(Z) \geq pH(X) + (1-p)H(Y)$.

4. **Source codes**. A source code C for a random variable X is a mapping from $\mathcal{A}$ to $\mathcal{D}^*$, the set of finite length strings of symbols from a D-ary alphabet. (Usually $\mathcal{D} = \{0, 1, \ldots, D-1\}$.)

(a) **Morse Code** In the Morse Code, letters are represented by a series of dots and dashes. Examples include S="dot dot dot", O="dash dash dash", E="dot" and Q="dash dash dot dash". Now you know how to signal SOS (="Save Our Ship"). You also see that letters that are more common in the English language are assigned shorter codewords and letters that are less common are assigned longer codewords. This means that the Morse Code (somewhat) efficiently encodes the English language. Also, since each letter has a different codeword (and breaks are placed between letters), a trained technician can perfectly recover a message sent in Morse Code.

i. How long do the Morse Code codewords need to be to represent all 26 letters in the alphabet? Hint: How many different codewords of length 1, length 2, etc. are there?

ii. What if we also need to represent the 10 digits plus 16 special symbols (\$, #, etc.)? How long do the codewords need to be now?

(b) **Example** Let $\mathcal{A} = \{\text{rock}, \text{paper}, \text{scissors}\}$ and let X be uniformly distributed over $\mathcal{A}$. A binary source code ($D = 2$) could be

$$\begin{aligned} C(\text{rock}) &= 0 \\ C(\text{paper}) &= 10 \\ C(\text{scissors}) &= 11. \end{aligned}$$

A ternary source code ($D = 3$) could be (boringly):

$$\begin{aligned} C(\text{rock}) &= 0 \\ C(\text{paper}) &= 1 \\ C(\text{scissors}) &= 2. \end{aligned}$$

(c) **Prefix-free codes** A prefix-free code is one in which no codeword is the prefix of another codeword. The two rock/paper/scissors codes are prefix-free. However, the Morse Code is not (the codeword for E is a prefix for the codeword for S). A sequence of codewords from a prefix-free code can be easily decoded. Simply start at the beginning of the string and keep going until the end of a codeword is reached.

For the binary rock/paper/scissors code, what sequence of symbols generates the sequence of codewords 001011110?

(d) **Prefix-free codes as trees** A prefix-free code can be represented in a tree structure with the following properties: each node has a maximum of D branches and each codeword is on a leaf of a tree. This follows from the fact that no codeword is the prefix of another codeword. Draw trees to represent the rock/paper/scissors codes:

(e) **Non-prefix-free codes** Consider the binary code:

$$C(\text{rock}) = 0$$
$$C(\text{paper}) = 01$$
$$C(\text{scissors}) = 11.$$

This is not a prefix-free code since C(rock) is a prefix for C(paper). Can you decode the sequence of codewords 001011110? If so, what was the original sequence and how did you find it? If not, why not?

(f) **Expected length of a code** The length of a codeword is simply the number of symbols it uses. For the binary rock/paper/scissors code, $l(\text{rock})=1$ and $l(\text{paper})=l(\text{scissors})=2$. The expected length of a code is the expectation over the distribution

$$L = \sum_{a \in \mathcal{A}} l(a)P(a).$$

What is the expected lengths of the different rock/paper/scissors codes?

(g) **Bounds on expected codeword length** Let us now compare the average codeword length for a code to the entropy of the random variable. Here the codes are D-ary prefix-free codes and the entropy is computed in base-D. See the appendix for sketches of proofs of these statements.

i. For any prefix-free code, the average codeword length satisfies:

$$L \geq H(X).$$

ii. There exists a prefix-free code such that the average codeword length satisfies:

$$L < H(X) + 1.$$

iii. The optimal average codeword length L^* prefix-free code satisfies:

$$H(X) \leq L^* < H(X) + 1.$$

(h) **Back to example** What is the entropy of the uniform rock/paper/scissors random variable in both base-2 and base-3? (Recall: $\log_2 3 \approx 1.585$.) How do these numbers compare to the average codeword lengths from Problem 4f?

(i) **Tightness of upper bound** Show that the bound $L < H(X) + 1$ cannot be improved. In particular, for any $\epsilon > 0$ show that there exists a random variable X with $L > H(X) + 1 - \epsilon$ for any prefix-free code. Hint: You can use a random variable that has been discussed in this packet.

(j) **Block coding** Consider independent, identically distributed random variables $X_1, X_2, \ldots, X_n$. Instead of encoding each X_i separately, we can encode them as a block. That is, our code is designed for the random variable $Y = \{X_1, \ldots, X_n\}$. Find upper and lower bounds on the optimal average codeword length L^* as a function of $H(X_1)$ and n. (Hint: See Problem 3(g)ii.) What happens to the codeword length per symbol $\frac{L^*}{n}$ as n goes to infinity?

5. **Huffman Codes - shortest expected length prefix-free code for a given source.** We will describe how to design a code for a given probability distribution that has the shorted expected codeword length.

   (a) **Graphical definition** We will go over how to construct Huffman codes on the board. The key step is to merge the D smallest probabilities into a new symbol.

   (b) **Recursive Definition** The following is a fairly thorough definition of how Huffman code construction works. This is included for reference and for anyone who prefers reading the gory details. We would like to create a D-ary code for a source X with sample space $\mathcal{A}$ where $n = |\mathcal{A}|$.

      i. If $n \leq D$, then we assign each member in the sample space a code of length 1.

      ii. If $n > D$, then we create a new random variable Y with $|\mathcal{Y}| = n - D + 1$, construct an optimal code for Y and use that to create code for X. To construct Y, we find the D least likely outcomes of X. That is, let $\mathcal{L} = \{x_1, \ldots, x_D\} \subset \mathcal{A}$ such that $P(a) \geq P(b)$ for all $a \notin \mathcal{L}$ and for all $b \in \mathcal{L}$. Let $\mathcal{Y}$ be the elements of $\mathcal{A}$ that are not in $\mathcal{L}$ plus a new element y_{merged}. Then,

$$P_Y(y) = \begin{cases} P_X(y), \text{ if } y \notin \mathcal{L} \\ \sum_{a \in \mathcal{L}} P_X(a), \text{ if } y = y_{\text{merged}}. \end{cases}$$

      Given a code for Y, we can form a code for X. If $x \notin \mathcal{L}$, use the same codeword. If $x \in \mathcal{L}$ use the codeword for y_{merged} plus an additional digit. This can be done since there are D elements in $\mathcal{L}$.

   (c) **Rock/Paper/Scissors** Confirm that the codes given in Problem 4b are Huffman Codes for $D = 2$ and $D = 3$.

(d) **Example** Let $\mathcal{A} = \{a, b, c, d\}$ and let

$$P(a) = 0.4$$
$$P(b) = 0.3$$
$$P(c) = 0.2$$
$$P(d) = 0.1$$

i. Binary code ($D = 2$): First, merge c and d into e with e having probability 0.3. Then, merge b and e into f with f having probability 0.6. There are only two symbols left and we can stop. The resuling codewords are $C(a) = 0$, $C(b) = 10$, $C(c) = 110$ and $C(d) = 111$.

ii. Ternary code ($D = 3$): First, merge b, c and d into e with e having probability 0.6. There are only two symbols left and we can stop. The resulting codewords are $C(a) = 0$, $C(b) = 10$, $C(c) = 11$ and $C(d) = 12$.

What are the expected lengths of the two codes?

(e) **Expanded Example** Let $\mathcal{L} = \{a, b, c, d, e\}$ and let

$$P(a) = 0.4$$
$$P(b) = 0.3$$
$$P(c) = 0.2$$
$$P(d) = 0.05$$
$$P(e) = 0.05$$

Find binary and ternary Huffman codes for this source. What are the expected lengths of the two codes?

(f) **Inserting dummy variables** For the ternary ($D = 3$) codes for the previous two examples, which has the higher expected codeword length?

i. How can we use the code from the expanded example to construct a better code for the first example? Hint: Add a "dummy" symbol to $\mathcal{A}$ with probability zero. This will not effect the average codeword length.

ii. Given D, for what sample space sizes $n = |\mathcal{A}|$ does the Huffman Code algorithm terminate with exactly D elements (not less than D)?

iii. Given D and $n = |\mathcal{A}|$, how many dummy symbols of probablity 0 should we insert so that the Huffman Code algorithm terminates with exactly D elements.

6. **Questions for further thought.** Below are just a small subset of questions that can be answered using information theory. There are also many areas of open research.

   (a) What can we say about codes that can be decoded but are not prefix-free?

   (b) What happens if we don't know the probability distribution of a source? Can we design a **universal** source code that does as well as one designed for the distribution?

   (c) What if we are happy with a degraded version of the original message (for example, JPEG compression)? How many bits do we need for that?

   (d) Another kind of coding studied in information theory is **channel coding**. Instead of removing redundancy to represent a source with the fewest bits possible (like in source coding), redundancy is added so that a message can be reliably received when sent over a noisy channel. One simple example of a noisy channel is one in which bits are sent and an error occurs with probabilty p. The question now becomes what is the greatest data rate (bits/channel use) that can be reliably sent over a given channel?

   (e) How do we design practical codes that achieve the theoretical limits?

   (f) Further reading:

      i. "Elements of Information Theory" by Cover and Thomas. Common textbook for graduate level information theory course.

      ii. "A Mind at Play" by Soni and Goodman. Biography of Claude Shannon, the inventor of information theory. A lot of stuff about Shannon's life but also talks about some of the mathematical concepts.

      iii. "A Mathematical Theory of Communication" by Claude Shannon. Written in 1948 for the Bell System Technical Journal, this is the article that started information theory.

7. **Appendix: Proofs** In this section, we sketch the proof of the bounds given in Section 4g. Several steps are left for the reader to justify.

   (a) **Lemma: Kraft inequality** For any D-ary prefix-free code, the codeword lengths $l_i = l(x_i)$ satisfy

   $$\sum_i D^{-l_i} \leq 1.$$

   Conversely, given a set of codeword lengths $\{l_i\}$ that satisfy the inequality above, one can construct a D-ary prefix-free code with those codeword lengths. To prove the direct part, think of the prefix-free code as tree with at most D branches at every node. The details are left to the reader.

   (b) **Kraft Inequality practice** In a certain D-ary prefix-free code, there are 6 codewords. Two codewords have length 1, two codewords have length 2, and two codewords have length 3. What values can D be?

(c) **Upper bound on expected length of optimal code** For a given random variable X distributed over $\mathcal{A}$, let

$$l(a) = \left\lceil \log_D \left(\frac{1}{P(a)} \right) \right\rceil,$$

for all $a \in \mathcal{A}$. Here $\lceil y \rceil$ is the "ceiling" of y, which denotes the smallest integer greater than or equal to y. Show that $\sum_{a \in \mathcal{A}} D^{-l(a)} \leq 1$:

By the converse to the Kraft inequality, we can construct a code with these lengths. Justify the following steps to upper bound the expected codeword length of the code:

$$\begin{aligned} L &= \sum_{a \in \mathcal{A}} P(a) l(a) \\ &= \sum_{a \in \mathcal{A}} P(a) \left\lceil \log_D \left(\frac{1}{P(a)} \right) \right\rceil \\ &< \sum_{a \in \mathcal{A}} P(a) \left(\log_D \left(\frac{1}{P(a)} \right) + 1 \right) \\ &= H(X) + 1 \end{aligned}$$

The best code can only be better than this one, thus the smallest expected length L^* is upper bounded by $H(X) + 1$.

(d) **Lower bound on expected length** For any code that satisfies the Kraft inequality, we have

$$\begin{aligned} L - H_D(X) &= \sum P(a) l(a) - \sum P(a) \log_D \frac{1}{P(a)} \\ &= -\sum P(a) \log_D D^{-l(a)} + \sum P(a) \log_D P(a) \\ &= \sum P(a) \log_D \frac{P(a)}{D^{-l(a)}} \\ &\geq \left(\sum P(a) \right) \log_D \frac{\sum P(a)}{\sum D^{-l(a)}} \\ &\geq 0. \end{aligned}$$

The first inequality is known as the log-sum inequality. (We will not prove that here.) The last step follows from the Kraft inequality.

Combining this with the upper bound, we see that the smallest expected codelength for a given source X satisfies:

$$H(X) \leq L^* < H(X) + 1$$