CS 761 : Randomized Algorithms , Spring 2018 , Waterloo

Lecture 12 : Entropy

We study the notion of entropy and see its role in randomness extraction, compression, and error correcting codes.

Entropy

Entropy is an important definition that measures the amount of "information" in a random variable.

We will see that it arises naturally in randomness extraction, compression, and error-correcting codes.

Definition $H(X) = -\sum_x \Pr(X=x) \log_2 \Pr(X=x)$.

Equivalenty, $H(X) = E[\log_2(1/\Pr(X=x))]$..

For a binary random variable, $H(p) = -p\log_2 p - (1-p)\log_2(1-p)$.

Some basic facts about the entropy function:

- $H(X) \leq \log_2 |S|$, where S is the set of outcomes of X.
- $H(X,Y) = H(X|Y) + H(Y) \leq H(X) + H(Y)$, and equality achieves when X and Y are independent.

Entropy and Binomial Coefficients

The following is a basic relation between entropy and binomial coefficients that will be used later.

Lemma Suppose nq is an integer where $0 < q < 1$. Then $\frac{2^{nH(q)}}{n+1} \leq \binom{n}{nq} \leq 2^{nH(q)}$.

proof Upper bound: $\binom{n}{nq} q^{qn} (1-q)^{(1-q)n} \leq \sum_{k=0}^{n} \binom{n}{k} q^k (1-q)^{n-k} = (q + (1-q))^n = 1$.

$$\Rightarrow \binom{n}{nq} \leq q^{-qn} (1-q)^{-(1-q)n} = 2^{-q\log_2 q\, n - (1-q)\log_2(1-q)n} = 2^{nH(q)}.$$

Lower bound: We will argue that $\binom{n}{nq} q^{qn} (1-q)^{(1-q)n}$ is the largest term in the binomial expansion and thus $\binom{n}{nq} q^{qn}(1-q)^{(1-q)n} \geq \frac{1}{n+1}$, and so $\binom{n}{nq} \geq \frac{2^{nH(q)}}{n+1}$.

To see that this is the largest term in the binomial expansion, we just need to compare two consecutive terms $\binom{n}{k} q^k (1-q)^{n-k} - \binom{n}{k+1} q^{k+1} (1-q)^{n-k-1}$

$$= \binom{n}{k} q^k (1-q)^{n-k} \left(1 - \frac{n-k}{k+1} \cdot \frac{q}{1-q}\right)$$

This difference is nonnegative iff $1 - \frac{n-k}{k+1} \cdot \frac{q}{1-q} \geq 0$ iff $k \geq qn - 1 - q$.

Thus $k = qn$ gives the largest term in the expansion. $\square$

Randomness Extraction

The goal here is to extract uniform random bits from a random variable.

<u>Definition</u>: Given x, an extraction function Ext should satisfy: $\Pr(\text{Ext}(X) = y \mid |y| = k) = 1/2^k$.

In other words, whenever its output is k bits, all k-bit strings are equally likely.

We will show that, on average, we can extract $H(p)$ bit from a coin flip with probability p head.

The proof will consist of two steps:

① If X is a random variable chosen uniformly from $\{0, \dots, m-1\}$, then we can extract $\sim \log_2 m$ bits.

② If we have n coin flips, then we can extract a random variable from $\{0, \dots, m-1\}$ for a large m.

The first step is easier.

<u>Theorem</u> If X is chosen uniformly randomly from $\{0, \dots, m-1\}$, then it is possible to extract $\lfloor \log_2 m \rfloor - 1$ independent and unbiased bits.

<u>Proof</u> (sketch) Let $\alpha = \lfloor \log_2 m \rfloor$.

The best case is when $m = 2^\alpha$, in which we just output the α-bit representation of X.

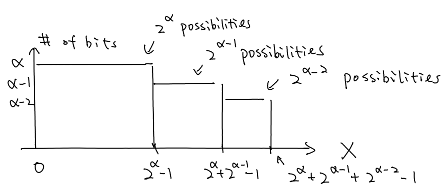
The worst case is when $m = 2^{\alpha+1} - 1$. What we will do is the following.

If $X \in \{0, \dots, 2^\alpha - 1\}$, then output the α-bit representation of X.

Otherwise if $X \in \{2^\alpha, \dots, 2^\alpha + (2^{\alpha-1} - 1)\}$, then output the $(\alpha-1)$-bit representation of $(X - 2^\alpha)$.

Otherwise if $X \in \{2^\alpha + 2^{\alpha-1}, \dots, 2^\alpha + 2^{\alpha-1} + (2^{\alpha-2} - 1)\}$, then output the $(\alpha-2)$-bit representation of $(X - 2^\alpha - 2^{\alpha-1})$, and so on.

Pictorically,



It is clear that this will satisfy the requirement of the extraction function.

It can be shown that the expected number of bits output is at least $\lfloor \log_2 m \rfloor - 1$, by a simple induction (See Theorem 9.4 of MU). $\square$

Now, suppose we have a biased coin with probability $p > 1/2$ being head.

We cannot extract < 1 bit from one coin flip.

So, instead, we flip the coins many times and hope to extract $H(p)$ bit per coin flip.

<u>Theorem</u> Given a coin which comes up head with probability $p > \frac{1}{2}$. For any δ and any sufficiently large n, there exists an extraction function, given n independent coin flips as input, outputs an average of at least $(1-\delta) n H(p)$ random bits.

<u>proof:</u> Algorithm: If there are j heads, map it to a number in $\{0, \dots, \binom{n}{j} - 1\}$.

Then use the previous algorithm to extract $\lfloor \log_2 \binom{n}{j} \rfloor - 1$ random bits.

Clearly, this algorithm satisfies the definition of an extraction function. We analyze its performance.

Let B be the number of bits output by the extraction function.

Let Z be the number of heads in the input.

Then $E[B] = \sum_{k=0}^{n} E[B \mid Z=k] \Pr(Z=k) \geq \sum_{k=\lfloor n(p-\varepsilon) \rfloor}^{\lceil n(p+\varepsilon) \rceil} E[B \mid Z=k] \Pr(Z=k)$ where ε is small s.t. $p - \varepsilon > \frac{1}{2}$

$$\geq \sum_{k=\lfloor n(p-\varepsilon) \rfloor}^{\lceil n(p+\varepsilon) \rceil} \left(\lfloor \log_2 \binom{n}{k} \rfloor - 1 \right) \Pr(Z=k) \quad \text{by previous theorem}$$

$$\geq \left(\log_2 \frac{2^{nH(p+\varepsilon)}}{n+1} - 2 \right) \sum_{k=\lfloor n(p-\varepsilon) \rfloor}^{\lceil n(p+\varepsilon) \rceil} \Pr(Z=k) \quad \text{by lemma}$$

The sum is just $\Pr(|Z - np| \leq \varepsilon n)$, which is at least $1 - 2e^{-(np)(\varepsilon/p)^2/3} = 1 - 2e^{-n\varepsilon^2/3p}$ by Chernoff.

Hence, $E[B] \geq \left(nH(p+\varepsilon) - \log_2(n+1) - 2 \right) \left(1 - 2e^{-n\varepsilon^2/3p} \right)$.

By choosing ε small enough, we have $nH(p+\varepsilon) \geq (1 - \delta/3) n H(p)$.

By setting n large enough, we have $\left(1 - 2e^{-n\varepsilon^2/3p} \right) \geq 1 - \delta/3$.

So, $E[B] \geq \left((1-\delta/3) n H(p) - \log_2(n+1) - 2 \right) (1 - \delta/3)$.

By setting n large enough, $(\delta/3) n H(p) > \log_2(n+1) - 2$, and thus

$$E[B] \geq \left(1 - (2\delta/3) n H(p) \right) (1 - \delta/3) \geq (1-\delta) n H(p). \quad \square$$

We cannot do better than this.

<u>Theorem</u> Under the same setting, the average number of bits output by an extraction function Ext on an input sequence of n independent flips is at most $nH(p)$.

<u>Proof</u> The main observation is: If an input sequence x occurs with probability q, then the corresponding output sequence $\mathrm{Ext}(x)$ can have at most $|\mathrm{Ext}(x)| \leq \log_2 (1/q)$ bits.

This is because all sequences with $|\mathrm{Ext}(x)|$ bits would have probability q by the requirement of an extraction function, and thus we must have $2^{|\mathrm{Ext}(x)|} \cdot q \le 1 \Rightarrow |\mathrm{Ext}(x)| \le \log_2(1/q)$.

Therefore, $E[B] = \sum_x \Pr(X=x)\,|\mathrm{Ext}(x)| \le \sum_x \Pr(X=x) \log_2\left(1/\Pr(X=x)\right) = H(X) = nH(p)$. $\square$

Compression

The goal here is to compress the output of a random variable.

We will show that n coin flips can be compressed as roughly $nH(p)$ bits.

Definition : A compression function takes as input a sequence of n coin flips, and outputs a sequence of bits such that each input yields a distinct output.

The following theorem is also known as the Source Coding theorem.

Theorem Consider a coin with probability $p > 1/2$ head. For any $\delta > 0$ and any sufficiently large n, there exists a compression function such that the expected number of bits used is at most $(1+\delta)nH(p)$.

Proof Let $\varepsilon > 0$ be a small enough constant with $p - \varepsilon > 1/2$.

Algorithm: When there are less than $n(p-\varepsilon)$ heads, then set the first bit to be one and use the naive scheme (i.e. one bit for each flip).

When there are at least $n(p-\varepsilon)$ heads, then there are at most

$$\sum_{j=\lceil n(p-\varepsilon)\rceil}^{n} \binom{n}{j} \le \sum_{j=\lceil n(p-\varepsilon)\rceil}^{n} \binom{n}{\lceil n(p-\varepsilon)\rceil} \le \frac{n}{2}\, 2^{nH(p-\varepsilon)}$$ such sequences (by Lemma).

For each such sequence, assign a unique sequence of at most $nH(p-\varepsilon) + \log_2 n$ bits to represent it, and add a zero to the first bit.

By Chernoff bound, $\Pr(np - X > n\varepsilon) < e^{-n\varepsilon^2/2p}$.

Therefore, the expected number of bits required is at most

$$e^{-n\varepsilon^2/2p}(n+1) + \left(1 - e^{-n\varepsilon^2/2p}\right)\left(nH(p-\varepsilon) + \log_2 n + 1\right).$$

By setting ε small enough and n large enough, similar argument as in above theorem can be used to show that this is at most $(1+\delta)nH(p)$. $\square$

There is an almost matching lower bound.

Theorem Under the same setting, the expected number of bits for any compression function $\geq (1-\delta) n H(p)$.

proof Observation: If a sequence S_1 is more likely than S_2, then an optimal compression function would assign fewer bits to S_1 than to S_2. Since $p > 1/2$, a sequence with more heads should get fewer bits than sequences with fewer heads.

Consider those sequences with $\lfloor n(p+\varepsilon) \rfloor$ heads. There are $\binom{n}{\lfloor n(p+\varepsilon)\rfloor} \geq 2^{nH(p+\varepsilon)}/(n+1)$ of them.

Therefore, one of them must need $\log_2 \left(2^{nH(p+\varepsilon)}/(n+1) \right) - 1 = nH(p+\varepsilon) - \log_2(n+1) - 1$ bits, as there are not enough shorter strings.

By Chernoff bound, the number of heads is at most $\lfloor n(p+\varepsilon) \rfloor$ with probability $\geq 1 - e^{-n\varepsilon^2/12p}$.

Therefore, by the above observation, the expected number of bits output $\geq (1-e^{-n\varepsilon^2/12p})(nH(p+\varepsilon) - \log_2(n+1) - 1)$.

By setting ε small enough and n large enough, this is at least $(1-\delta) n H(p)$ bits. $\square$

Error Correcting Code

Alice wants to send a message to Bob through a noisy channel, where each bit is flipped independently with probability p.

Question: Can they come up with a scheme so that Bob can figure out what Alice wanted to send?

Definition: A (k,n) encoding function maps k bits to n bits.

A (k,n) decoding function maps n bits to k bits.

Shannon solved this problem with matching upper and lower bounds.

This is the founding result of information theory.

Theorem For $p < 1/2$, for any $\delta, \gamma > 0$, for sufficiently large n, for any $k \leq n(1-H(p)-\delta)$, there are (k,n) encoding and decoding functions such that the receiver can obtain the correct message with probability at least $1-\gamma$.

proof First, we prove a weaker result that there exist appropriate coding functions when the input is chosen uniformly at random from all k-bit inputs. Then, we will modify to ensure that the error probability is at most γ on every possible input.

Encoding: For each k-bit string, assign a random n-bit string. Let the codewords be $X_0, X_1, \ldots X_{2^k-1}$.

Decoding: Assume the receiver know the codewords $X_0, \dots, X_{2^k-1}$. Let the received string be R.

Check each X_i if the number of different bits between X_i and R is between $n(p-\varepsilon)$ and $n(p+\varepsilon)$. If there is a unique such X_i, then output the k-bit string corresponding to X_i as the message sent (i.e output i).

Error possibilities: ① The original codeword differs from R in less than $n(p-\varepsilon)$ bits or more than $n(p+\varepsilon)$ bits.

② There are more than one codewords differ from R in between $n(p-\varepsilon)$ and $n(p+\varepsilon)$ bits.

① By Chernoff bound, this is at most $2e^{-\varepsilon^2 n/3p} < \gamma/2$ when n is large enough.

② Suppose X_0 was sent and R is received. Since $X_1, \dots X_{2^k-1}$ are chosen randomly,

$$
\begin{aligned}
&\Pr(\,R \text{ and some } X_i \text{ differ in between } n(p-\varepsilon) \text{ and } n(p+\varepsilon) \text{ bits}\,)\\
\le\ & 2^k \sum_{j=n(p-\varepsilon)}^{n(p+\varepsilon)} \binom{n}{j} \Big/ 2^n \\
\le\ & 2^k \cdot n \cdot \binom{n}{n(p+\varepsilon)} \Big/ 2^n \\
\le\ & 2^k \cdot n \cdot 2^{nH(p+\varepsilon)} / 2^n \\
=\ & n \cdot 2^{n(H(p+\varepsilon) - H(p) - \delta)} \qquad \text{since } k \le n(1-H(p)-\delta).
\end{aligned}
$$

By setting ε small enough, $H(p+\varepsilon) - H(p) - \delta$ is negative.

By setting n large enough, this probability is at most $\gamma/2$.

Therefore, the error probability is at most γ when X_0 was sent.

Thus, the total error probability is at most $2^k\gamma$.

There exists a specific $X_0, \dots, X_{2^k-1}$ with total error probability at most $2^k\gamma$.

If each message was sent with probability $1/2^k$, then the average error probability is $\le \gamma$.

<u>Worst case error probability</u>: This is similar to the deletion method we have seen before.

By Markov's inequality, there are at most half the codewords with error probability $> 2\gamma$.

We just remove those codewords. It becomes a $(k-1, n)$ encoding and decoding scheme. □

The above result is best possible.

<u>Theorem</u> Under the same setting, for any $k \geq n(1-H(p)+\delta)$, there are no (k,n) encoding and decoding functions such that the receiver can obtain the correct message with probability at least γ.

<u>proof</u> (idea). The channel error is between $n(p-\varepsilon)$ and $n(p+\varepsilon)$ with high probability.

Consider the simpler setting where the decoding function is always correct in this range.

Then each codeword is associated with

$$\sum_{k=n(p-\varepsilon)}^{n(p+\varepsilon)} \binom{n}{k} \geq \binom{n}{np} \geq 2^{nH(p)}/n+1 \text{ bit sequences.}$$

But there are 2^k codewords.

So, $2^k \cdot \left(\frac{2^{nH(p)}}{n+1}\right) \geq 2^{n(1-H(p)+\delta)}\left(\frac{2^{nH(p)}}{n+1}\right) > 2^n$ for large enough n.

The associated sequences are more than 2^n, a contradiction.

The argument can be modified to allow for small decoding error in this range. □

We note that the above decoding algorithm is inefficient, with running time exponential on the the message size.

Finding a code with efficient encoding and decoding algorithms (ideally linear time algorithms) achieving the rate of the Shannon's coding theorem is still an active research area, fifty years after Shannon's result.

<u>Reference</u> : The material is from chapter 9 of "Probability and Computing" by Mitzenmacher and Upfal.