CS 229r Information Theory in Computer Science	Feb 7, 2019
Lecture 4	
Instructor: Madhu Sudan	*Scribe: Daniel Chiu*

1 Miscellaneous

1.1 Schedule for today

- Single Shot Compression
- Universal Compression
- Markovian Sources

1.2 Logistics

Problemset 1 is due at 8pm on Friday 2/8. You can use a total of 3 late days over the semester, but only at most 2 can be used on a single problemset.

2 Single Shot Compression

Up to now, we've been thinking of compression as measuring something. For instance, entropy is a measurement; information is a measurement. Now, we will think of compression as a problem - e.g. "you have a file, you want to compress it [map it to a more transmittable form] right away".

Definition 1 (Single Shot Compression)**.** In the *Single Shot Compression* problem, there are two parties, a sender and a receiver. They both know some distribution $P = (p_1, \cdots, p_m)$ over the possible inputs to the encoding. Sender additionally knows some $X \sim P$, and wants to transmit it using the encoder E. The encoder $E : [m] \to \{0,1\}^*$ should give rise to a prefix-free encoding, which implies the existence of a decoder $D : \{0,1\}^* \to [m] \cup \{?\}$ (where ? denotes an unknown input). The goal is to minimize the expected length, over the distribution P, of the encoding:

$$\min_E \{\mathbb{E}_{X\sim P}[|E(x)|]\} \tag{1}$$

It turns out that there exists an optimal algorithm for Single Shot Compression - giving an encoder that minimizes the expected length of the encoding for any distribution P. This is the **Huffman Encoding**.

How do we bring entropy into this? **Shannon Encoding** solves the problem using at most $H(X) + 1$ bits. Note that entropy by definition tells us that we need at least $H(X)$ bits - this is the **Shannon Lower Bound** (which we've already seen), so Shannon Encoding is within 1 bit of the [a priori] best possible solution.

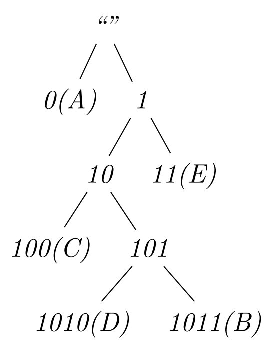
2.1 Huffman Coding

To understand Huffman coding, we first describe the **Encoding Tree**.

Example 2. *Suppose we have the mapping*

$$
\begin{aligned}
A &\to 0 \\
B &\to 1011 \\
C &\to 100 \\
D &\to 1010 \\
E &\to 11
\end{aligned}
$$

We can represent this as a binary tree, where a 0 representing taking a left edge, and 1 a right edge. If we mark each node which is the terminus of an output of the encoder, the prefix-free condition means that for any marked node, none of its ancestors are also marked.



Definition 3 (Huffman Encoding)**.** Given $P = (p_1, \cdots, p_m)$, the encoding function E for the *Huffman Encoding* is obtained by the following recursive algorithm:

1. If $m = 1$, encode $E(1) =$ “” (the empty string) and return. This is the base case.
2. Sort the p_i. For the remaining steps, assume $p_1 \geq \cdots \geq p_m$.
3. Merge p_m, p_{m-1} to get $Q = (q_1, \cdots, q_{m-1})$ where $q_i = p_i$ except $q_{m-1} = p_{m-1} + p_m$.
4. We build up the encoding for Q recursively. Given E' defined on $[m-1]$, define E which encodes $1, \cdots, m-2$ as E' does, but let $E(m-1) = E'(m-1) \circ 0, E(m) = E'(m-1) \circ 1$ (where $\circ$ denotes concatenation). Note that this preserves prefix-free-ness.

Here's a sketch of the proof of optimality:

Proof. Suppose E is some optimal encoding of $P = (p_1, \cdots, p_m)$. Without loss of generality, $p_1 \geq \cdots \geq p_m$, and define $\ell_i = |E(i)|$ for $1 \leq i \leq m$.

If we didn't have $\ell_1 \leq \ell_2 \leq \cdots \leq \ell_m$, then consider any pair (i, j) such that $p_i > p_j$ but $\ell_i > \ell_j$. Swapping the two encodings for i and j reduces the cost by $(p_i - p_j)(\ell_i - \ell_j)$, which is positive, contradiction. Thus, modulo equal probabilities, we have that the ℓ's are nondecreasing, and we can swap encodings for elements with equal probability without cost to get this in general.

Consider the encoding tree of E. Since l_m is maximal, the encoding $E(m)$ must be a leaf node N of the encoding tree. Unless $m = 1$, then N has a sibling N', and N' must be an encoding as well (why?). Thus, merge $m, m-1$ in the same way as the algorithm above does, and find the optimal tree for that string $(P_1, \cdots, P_{m-1} + P_m)$. Inductively, that tree must also be optimal, and we are basically done. $\square$

Exercise 4. *Complete and formalize the above proof.*

Solution: We omit most of the formalization since it follows from the sketch above, and just answer the "why?" above.

If N' above was not an encoding, then since N is a node with maximal depth, then N' has no children that are encodings either. This is impossible as m would've been encoded by node N instead.

When discussing Huffman codes, normal algorithm classes stop here. However, we'll go further to show that the expected encoding length that Huffman coding achieves is bounded by $H(x) + 1$. Note that this *is* surprisingly good, because this is for Single Shot Compression, whereas entropy is defined based on the limit of encoding more and more copies of the base text.

2.2 Shannon Encoding

To do so, we move on to...

Definition 5 (Shannon Encoding)**.** *Shannon Encoding* also takes in $P = (p_1, \cdots, p_m)$. We'll say upfront that to encode i, we will use $|E(i)| = \ell_i = \lceil \log \frac{1}{p_i} \rceil$ bits. Since $\sum_i p_i = 1$, we have $\sum_i 2^{-\ell_i} \leq \sum_i p_i \leq 1$ by our definition of ℓ_i. Thus, Kraft's inequality holds, and an encoding function exists with these $\ell_1, \cdots, \ell_m$.

Remark: Note that depending on how you prove Kraft's inequality, this might be entirely nonconstructive. However, it does have the interesting property that given any one p_i, we can immediately determine the length of its encoding $E(i)$ without knowing the other probabilities.

We can immediately analyze the performance (expected encoding length) of Shannon encoding:

$$\mathbb{E}_{X\sim P}[|E(x)|] = \sum_i p_i \ell_i = \sum_i p_i \lceil \log \frac{1}{p_i} \rceil \leq \sum_i p_i \left(\log \frac{1}{p_i} + 1 \right) = H(X) + 1$$

Remark: Since Huffman encoding is optimal and thus at least as good as Shannon, Huffman (which is harder to analyze) achieves at most $H(X) + 1$ as well. It's quite remarkable that entropy captures optimal encoding length so well.

Further Exploration: Is the gap of 1 between entropy and Shannon/Huffman tight? We know

$$H(X) \leq \text{Huffman length} \leq \text{Shannon length} \leq H(X) + 1$$

There's a total gap of 1 between $H(X)$ and $H(X) + 1$. Try to find distributions X that maximize the gap between each pair of adjacent quantities (one gap at a time - maximize Huffman length $- H(X)$, and so on).

Interlude: A long time ago, people we trying to build the first fax machine, and thought about compression. To compress, they had to have a distribution of the input, and they found frequencies of small strings manually. This was the state of the art in fax machines for 20 years.

3 Universal Compression

Unfortunately, uses of Single Shot Compression are uncommon in the real world. For instance, if you use gzip and feed it a new file, it'll work regardless of language or of having some prior on the distribution you're feeding in. This leads to the idea of **Universal Compression** - compression that works for any distribution and any source of information.

Definition 6 (Universal Compression)**.** The *Universal Compression* problem takes an input string $w \in \Sigma^n$ (Σ is the alphabet) and compresses w to $\{0,1\}^*$. The result should be invertible and prefix-free. Similar to the single shot version, we can define an expected length of encoding which should be minimized.

3.1 Lempel-Ziv

Lempel-Ziv gave an algorithm that was relatively effective empirically. There are some theorems for certain classes of probabilistic sources of w, which we will investigate more later in the semester. Today, we will describe the algorithm and some potential probabilistic sources.

What are we hoping for? We wish to find some repetitive structure; some self-similarity to exploit.

Definition 7 (Lempel-Ziv)**.** *Lempel-Ziv* compression begins by splitting the input string into encode-able pieces. Given w, we desire to split it into m small chunks "" $= s_0, s_1, s_2, \cdots, s_m$, so that w is the concatenation $s_0 \circ s_1 \circ \ldots \circ s_m$. For all i, let $s_i = s_{j_i} \cdot b_i$ for some $j_i < i$ and $b_i \in \Sigma$. In other words, each chunk should be a previous chunk with some extra character (except the last chunk). Furthermore, all chunks should be unique.

Exercise 8. *The above uniquely determines the chunking. Why?*

Solution: One observation is that if c is a chunk, then every prefix of c must have been a previous chunk.

We prove the above observation and the uniqueness of chunking by induction on the number of chunks so far. For the base case, the only possible first chunk is the first character, since each chunk must be a previous chunk with one extra character. For future chunks, consider the longest substring s that is equal to a previous chunk. No chunk shorter than $|s| + 1$ is legal, since all such substrings are prefixes of s and thus previous chunks. No chunk longer than $|s| + 1$ is legal, because it wouldn't be a previous chunk plus one character. Thus, the only legal next chunk is the substring of length $|s| + 1$, and each prefix of this chunk is a previous chunk, as desired.

Example 9.

$$w = 010111001101111101$$
$$w = 0|1|01|11|00|110|111|1101$$

Definition 10 (Lempel-Ziv (continued))**.** Finally, the encoding is simply $E(w) = PF((j_1, b_i), \cdots, (j_m, b_m))$, where the j's and b's are encoded in some prefix-free manner (denoted PF above). Each j will encode to about $\log(n)$ bits and each b to about $\log(\Sigma)$ bits long, so the total encoding is of length approximately $m(\log(n) + \log(\Sigma) \leq n(\log(n) + \log(\Sigma))$.

Exercise 11. *Apply Lempel Ziv to the following sequence:*

$$W = 010011000111000011110000011111$$

In addition, what would you conjecture are the worst and best case strings for the effectiveness of Lempel Ziv compression?

Solution: We decompose the string as follows:

$$W = 010011000111000011110000011111$$

$$= [0]\ \ [1]\ \ [00]\ \ [11]\ \ [000]\ \ [111]\ \ [0000]\ \ [1111]\ \ [00000]\ \ [11111]$$
$$\Rightarrow (\lambda, 0), (\lambda, 1), (A, 0), (B, 1), (C, 0), (D, 1), (E, 0), (F, 1), (G, 0), (H, 1)$$

From this, we can observe that, for every length k, the more substrings S_i of that length k, the longer our resulting encoding must be, since we must send along more combinations of length k, rather than strings of length greater than k that are built off of k. Thus, we conjecture that the worst case strings are those which contain (in increasing k order) all 2^k substrings of each length k, while the best case strings are those which contain exactly one substring of each length k, all using the same single character.

Exercise 12. *Find a prefix-free encoding of $\mathbb{Z}^+$ that encodes n using $\log n + O(\log\log n)$ bits.*

Solution: The prefix free encoding will look like

$$[\text{bits of length}]\ 01\ [\text{bits of } n]$$

To make the encoding decodable, the first segment will be encoded with pairs of bits. In other words, for each bit in $\lceil \log n \rceil$, the encoding will have two copies of that bit. Then, the "01" indicates the end of the first segment, and the length can be recovered from the bits read. Then, the next length bits are just n in binary.

This is prefix-free since otherwise one encoding is a prefix of another. However, that first encoding specifies how long the rest of the encoding after the "01" can be, contradiction.

The overall length is $2\lceil \log n \rceil + 2 + \lceil \log n \rceil$ which is $\log n + O(\log\log n)$ bits.

Example 13. *Continuing Example 10, we have that the pairs (j, b) are*

$$(0,0),(0,1),(1,1),(2,1),(1,0),(4,0),(4,1),(6,1)$$

Remark: Lempel-Ziv can often actually *expand* short strings. It doesn't "get going" until it builds up enough structure in the beginning of the string.

Remark: Can we iterate compression? Generally, no, because compression schemes usually aim to be approximately uniformly distributed on their output length (which is a consequence of being of length approximately equal to the entropy).

3.2 Markovian sources

Now, we aim to analyze the performance of Lempel-Ziv. To do so, we let the strings be drawn from some distribution P_X.

Theorem 14. *If $W = w_1 \circ \ldots \circ w_n$ where each is i.i.d. sampled from $w_i \sim P_X$, then as $n \to \infty$, with high probability the length of the compression is $(H(X) + o(1))n$.*

Note that another approach to this is Huffman coding - finding the sample frequency of each alphabet character, sending this distribution information so the decoder can be constructed, and then performing Huffman using this distribution. More surprisingly, Lempel-Ziv can effectively compress Markovian sources.

Definition 15 ((Time-invariant) Markov Chain)**.** A sequence $Z_1, \cdots, Z_n$ is a *Markov chain* if

$$\forall n : Z_n | Z_1, \cdots, Z_{n-1} \sim Z_n | Z_{n-1}.$$

By the $\sim$ notation, we mean the conditional distributions are the same. It is additionally *[*time-invariant] if

$$\forall n, m : Z_n | Z_{n-1} \sim Z_m | Z_{m-1}.$$

One piece of terminology - Z_i is called the "state" at time i. Furthermore, we can classify Markov chains where each state comes from a finite set:

Definition 16 (k-state Markov chain)**.** Suppose that for all i, $Z_i \in \Gamma = \{1, \cdots, k\}$.

Then, a k-*state Markov chain* is given by a $k \times k$ matrix M where $M_{ij} = Pr[Z_2 = j | Z_1 = i]$. In essence, this is a finite automaton.

We will only consider k-state Markov chains that are

1. Irreducible (strongly connected): there's a path from every state to every other state.

2. Aperiodic: the greatest common divisor of all cycle lengths is 1.

This implies the existence of the stationary distribution Π, such that $Z_i \sim \Pi \implies Z_{i+1} \sim \Pi$ if the initial distribution is Π.

Definition 17 (Entropy of Markov chain)**.** We can simplify the definition of entropy using properties of the Markov chains we're considering:

$$\begin{aligned} H(M) &= \lim_{n\to\infty} H(Z_n|Z_1,\cdots,Z_{n-1}) \\ &= \lim_{n\to\infty} H(Z_n|Z_{n-1}) \\ &= H(Z_2|Z_1) \end{aligned}$$

where the last equality holds when Z_1 is drawn from the stationary distribution Π.

Exercise 18. *Given the above, find the entropy of a k-state time-invariant Markov chain given the transition matrix M and the stationary distribution Π.*

Solution: Use the formula for conditional entropy.

Furthermore, we can hide the Markov chain in the background:

Definition 19 (Hidden Markov Model)**.** A *Hidden Markov Model* (HMM) has an underlying Markov chain $Z_1,\cdots,Z_n$. Given a distribution P_σ for each of the possible states $\sigma \in \Gamma$, this induces a second sequence $X_1,\cdots,X_n$ drawn from the first, where $X_i \sim P_{Z_i}$. This sequence $\{X_i\}$ is the observed output of the model.

It turns out that Lempel-Ziv can compress HMMs, and this is one of the nicest classes Lempel-Ziv can compress. We will see this later.