CS 229r Information Theory in Computer Science Feb 12, 2019

Lecture 5

Instructor: Madhu Sudan *Scribe: Pranay Tankala*

1 Overview

A *universal compression* algorithm is a single compression scheme applicable to input sequences from an entire class of information sources. An example we've seen before is the Lempel-Ziv '78 algorithm, which doesn't assume knowledge of specific the probability distribution used to generate its input. In this lecture, we will be primarily interested in input sequences drawn from the commonly encountered class of *Markovian sources*, or *hidden Markov models*. Today's goal is to work toward a proof that the Lempel-Ziv algorithm is a universal compression algorithm for the class of Markovian sources.

1.1 Today's Schedule

Universal Compression

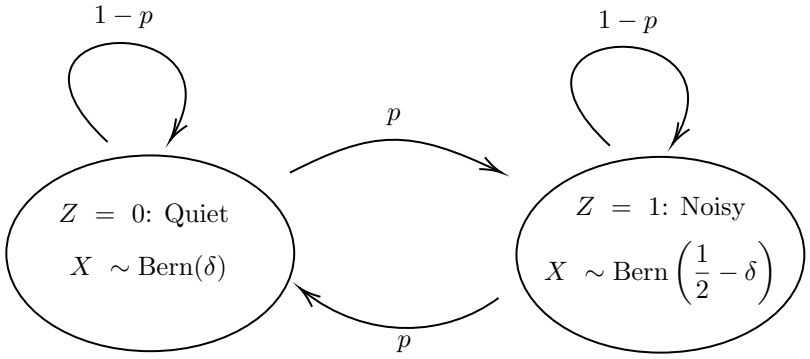
- Hidden Markov Models (HMM)
- A Universal Compressor
- Lempel-Ziv is Universal

2 Hidden Markov Models

We begin with some illustrative examples of hidden Markov models, which are closely related to Markov chains:

2.1 Outer Space

Consider a satellite listening for signals from outer space. Most of the time, there is hardly any information transmission, but occasionally, some highly entropic planetary activity is observed. We model the information source as a hidden Markov model with "quiet" and "noisy" states, controlled by some small positive parameters δ and p:



We denote the successive states of this information source as $Z_1, Z_2, \ldots, Z_n$, and the bit sequence produced as $X_1, X_2, \ldots, X_n$. In the quiet state ($Z_i = 0$) it's quite rare that $X_i = 1$, but in the noisy state ($Z_i = 1$), this

happens with probability nearly $\frac{1}{2}$. To be precise, we have $X_i \sim \text{Bern}(\delta)$ when $Z_i = 0$ and $X_i \sim \text{Bern}\left(\frac{1}{2} - \delta\right)$ when $Z_i = 1$. By the end of these notes, we will have the tools to compress the sequence $X_1, \ldots, X_n$.

2.2 Shannon's n-gram Model

A special case of hidden Markov models is the Markov chain. In Shannon's foundational paper, he discusses a Markov chain to model English, in which letters are drawn randomly according to a probability distribution conditional on the previous $n-1$ letters. Here is Shannon's example of a sequence produced using trigram frequencies:

```
IN NO IST LAT WHEY CRATICT FROURE BIRS GROCID PONDENOME OF DEMONSTURES
OF THE REPTAGIN IS REGOACTIONA OF CRE.
```

http://math.harvard.edu/~ctm/home/text/others/shannon/entropy/entropy.pdf

2.3 General Markovian Sources

We now provide a general definition of hidden Markov models that abstracts the key features of the previous two examples. Intuitively, a hidden Markov model consists of a Markov chain of "hidden" states $Z_1, Z_2, \ldots$, that can only be observed through a sequence of related variables $X_1, X_2, \ldots$ (which do not necessarily form a Markov chain). More formally:

Definition 1. A *hidden Markov model (HMM)* is given by

- A matrix $M \in \mathbb{R}^{k \times k}$ representing a Markov chain, where
$$M_{ij} = \Pr[\text{next state is } j \mid \text{current state is } i].$$
- A tuple of k distributions $P^{(1)}, \ldots, P^{(k)}$ supported on a set Ω.

When discussing a Markov chain M, we will always assume that M is irreducible and aperiodic, so that M has a unique stationary distribution $\Pi \in \mathbb{R}^k$ satisfying $\Pi M = \Pi$. In this case, the *states* $Z_1, Z_2, \ldots$ of the HMM are given by $Z_1 \sim \Pi$ and $Z_i \xrightarrow{M} Z_{i+1}$, so that $Z_i \sim \Pi$ for every i. The *output* of the HMM is a sequence $X_1, X_2, \ldots$ of random variables supported on Ω, generated according to $X_i \sim P^{(Z_i)}$

Definition 2. The *entropy rate* of a hidden Markov model is $\lim_{n\to\infty} H(X_n|X_1, \ldots, X_{n-1})$.

Under our assumptions on M, the above limit always exists. Indeed, we have

$$\begin{aligned} H(X_n|X_1, \ldots, X_{n-1}) &\le H(X_n|X_2, \ldots, X_{n-1}) && \text{(conditioning reduces entropy)} \\ &= H(X_{n-1}|X_1, \ldots, X_{n-2}) && \text{(we've assumed a stationary process),} \end{aligned}$$

so the values $H(X_n|X_1, \ldots, X_{n-1})$ are monotonically decreasing, and hence converge to a limit $H(M)$ from above.

Remark In Example 2.1, we have $k = 2$, $M = \begin{bmatrix} 1-p & p \\ p & 1-p \end{bmatrix}$ and distributions $\text{Bern}(\delta)$ and $\text{Bern}\left(\frac{1}{2} - \delta\right)$ for the quiet and noisy states, respectively. However, we lack a closed form expression for the entropy rate $H(M)$ in terms of p and δ. In fact, the best approximation algorithms we have at our disposal to compute $H(M)$ are still quite coarse.

Remark Without our assumptions on M, there would be no reason to expect the values $H(X_n|X_1, \ldots, X_{n-1})$ to converge to a limit. For example, if we were to take any Markov chain M and place an unavoidable "intermediate" node along each edge, the entropy $H(X_n|X_1, \ldots, X_{n-1})$ could be wildly different for even and odd n.

3 A Universal Compressor

In order to begin to describe universal compression algorithms for Markovian sources, we need

Theorem 3 (Asymptotic Equipartition Principle for Markovian Sources)**.** *For every hidden Markov model $(M, P^{(1)}, \ldots, P^{(k)})$ and $\varepsilon > 0$, for sufficiently large ℓ, there exists a set $S \subseteq \Omega^\ell$ such that for every start state Z_1,*

1. $\Pr[(X_1, \ldots, X_\ell) \notin S] \leq \varepsilon$
2. *For all $(\alpha_1, \ldots, \alpha_\ell) \in S$,*

$$\frac{1}{|S|^{1+\varepsilon}} \leq \Pr[(X_1, \ldots, X_\ell) = (\alpha_1, \ldots, \alpha_\ell)] \leq \frac{1}{|S|^{1-\varepsilon}}.$$

3. $|S| \approx 2^{(H(M)\pm\varepsilon)\ell}$

The set S is called the typical set*.*

Note that the AEP as stated above is more general than the version of the AEP we saw before, which dealt only with i.i.d. sequences of random variables.

Proof. We will only sketch some of the ideas that go into the proof of the general AEP. One approach is to first prove the theorem for the special case of Markov chains ($X_i = Z_i$), by reducing to the case of i.i.d. random variables. To do this, we would decouple the walk

$$\underbrace{X_1, X_2, \ldots, X_a}_{W_1}, \underbrace{X_{a+1}, \ldots, X_b}_{W_2}, \underbrace{X_{b+1}, \ldots, X_c}_{\cdots}, \ldots$$

into sequences W_i that end at state 1 *and* do not contain state 1 in between. The sequences W_i now have i.i.d. lengths and i.i.d. expected entropy. To reduce the case of a general HMM to that of a Markov chain, we could insert a handful of the state variables (say $Z_t, Z_{2t}, \ldots$) at periodic intervals. Doing so would make the entropy of the sequence higher, but not much higher, while giving us a Markov chain. □

Exercise 4. *Argue that the AEP, as stated above, is equivalent to the assertion that*

$$-\frac{1}{\ell} \log p(X_1, \ldots, X_\ell) \to H(M)$$

in probability as $\ell \to \infty$.

3.1 A Non-Universal Compressor

We'll first use the AEP to describe a remarkably simple compression algorithm for a *known* Markovian source M. Suppose we wish to encode a string $x_1 \cdots x_n$ produced by M. Take ℓ and S as in the statement of the theorem, and suppose that $n \gg \ell$. We first divide the string into blocks of length ℓ:

$$\underbrace{x_1 \cdots x_\ell}_{B_1} \underbrace{x_{\ell+1} \cdots x_{2\ell}}_{B_2} \cdots \underbrace{x_{n-\ell+1} \cdots x_n}_{B_{n/\ell}}.$$

To encode a block B_i, we first use a single bit to indicate whether or not B_i belongs to S, the set of typical sequences. If B_i is typical, the rest of the encoding consists of a mere $\log_2 |S|$ bits that distinguish B_i from the other members of S. If B_i is atypical, then we transmit an entire $\log |\Omega^\ell|$ bit sequence to specify B_i completely. By the AEP, the probability of the latter case is at most ε. Thus, the expected length of a single block's encoding is at most

$$(1 + \log_2 |S|) + \varepsilon \cdot (1 + \log_2 |\Omega^\ell|) \approx (H(M) + \varepsilon + \varepsilon \log_2 |\Omega|) \cdot \ell,$$

which is very close to the optimal rate of $H(M)$ bits per input symbol. To encode x, we simply concatenate the encodings of $B_1, \ldots, B_{n/\ell}$.

3.2 Universal Huffman Algorithm

The previous algorithm made use of the values ℓ, S. This is problematic for two reasons. First, it limits the algorithm to a *single* Markovian source. Second, we generally do not know how to compute ℓ or S. We attempt to circumvent these issues with what we will call the *universal Huffman algorithm*. Suppose we are given $x_1, \ldots, x_n$ to encode, and a parameter ℓ. The algorithm proceeds in stages:

1. Divide x into size ℓ blocks $B_1, \ldots, B_{n/\ell}$ as before:

$$\underbrace{x_1 \cdots x_\ell}_{B_1} \; \underbrace{x_{\ell+1} \cdots x_{2\ell}}_{B_2} \; \cdots \; \underbrace{x_{n-\ell+1} \cdots x_n}_{B_{n/\ell}}.$$

2. Compute the *empirical frequency*

$$f_w = \text{frequency of } w \text{ among } B_1, \ldots, B_{n/\ell}$$

   for each $w \in \Omega^\ell$.

3. For each $i = 1, \ldots, n$, compute

$$Z_i = \text{Huffman}(B_i; \{f_w\}),$$

   the compression of B_i under the Huffman code tree corresponding to the frequences $\{f_w\}$.

4. Finally, transmit some encoding of the empirical frequencies $\{f_w\}$, followed by the sequence $Z_1, Z_2, \ldots, Z_{n/\ell}$.

Notice that the Huffman code tree itself need not be transmitted, since the decoder can rebuild it from scratch using just the empirical frequencies $\{f_w\}$. Furthermore, using the AEP, one can prove:

Theorem 5. *For all Markovian sources M and $\varepsilon > 0$, as ℓ and n grow, we have*

$$\Pr[|\textit{Univ-Huffman}(x_1, \ldots, x_n)| > n \cdot (H(M) + \varepsilon)] \to 0$$

We don't know much about how large n must be, as a function of ℓ, in order for the above algorithm to consistently perform well. We know even less about how large ℓ must be, as a function of the number of states in underlying Markov chain producing the input. If you were surprised by the seemingly inexpensive computations needed for the universal Huffman algorithm, be warned that the real complexity is hidden in the size of n, which may need to be exponentially large for the compression to work well.

Exercise 6. *Prove, using principles of complexity theory, that the total compression time for a universal compression algorithm must be at least exponential in either $1/\varepsilon$ or k, the number of states of the Markov chain.*

4 Lempel-Ziv is Universal

We begin with a review of the Lempel-Ziv algorithm to compress a string $x = x_1 \cdots x_n$, $x_i \in \Omega$.

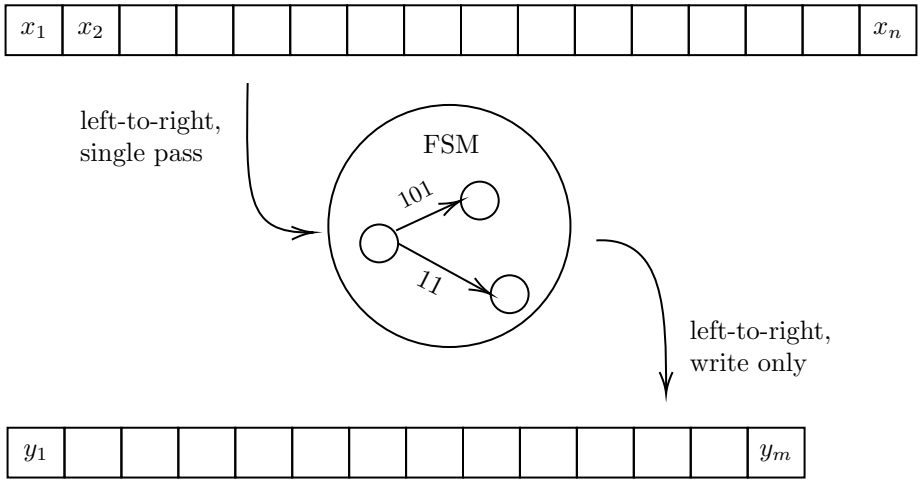
1. First, we parse x into consecutive phrases $s_0 \circ s_1 \circ s_2 \circ \cdots \circ s_t$, such that
   (a) s_0 is the empty string.
   (b) $s_i = s_{j_i} \circ b_i$, for some $b_i \in \Omega$ and $j_i < i$.
   (c) $s_i \neq s_j$ for $i \neq j$.
2. Encode each j_i using $\log t + o(\log t)$ bits, and encode each b_i using $\log |\Omega|$ bits.
3. Transmit the encoding of each j_i and b_i.

The resulting compression length is $(t \log t)(1 + o(1))$ in total.

4.1 Analyzing Lempel-Ziv

Before analyzing the Lempel-Ziv algorithm, we need to introduce one more important construct: the finite state compressor.

Definition 7. A *finite state compressor* (FSC) is a finite state machine (FSM) that writes 0 or more symbols to an output tape during each state transition (see diagram).



Finite state compressors have a very important property related to Markovian sources, which will be crucial in our analysis of Lempel-Ziv:

Theorem 8. *For any hidden Markov model M and $\varepsilon > 0$, there exists a finite state compressor that achieves a compression rate of $(1+\varepsilon)H(M)$.*

It turns out that the proof of the theorem involves constructing a FSC that reads its input sequence in blocks of size ℓ. As a result, the number of states S in the machine exceeds Ω^ℓ, which is exorbitantly large compared to the k-state machine M used to produce the input!

Exercise 9. *Prove Theorem 8 by showing that* Univ-Huffman$^\ell$ *can be converted to a finite state compressor (with preprocessing of M).*

Recall that we want to prove that Lempel-Ziv achieves a nearly optimal compression rate. By the previous theorem, it suffices to show that the performance of Lempel-Ziv is "not much worse" than that of any finite state compressor. To make this comparison precise, we define $C(x)$, for a string $x \in \Omega^*$, as the maximum integer t such x can be written as the concatenation of t distinct strings $Y_1, \ldots, Y_t$ in Ω^*. One might interpret $C(x)$ as a measure of the inherent complexity of compressing the string x. Notice that, in the Lempel-Ziv parsing $x = s_1 \circ \cdots \circ s_t$, we have $t \leq C(x)$. Using $C(x)$, we can state the following important bounds:

1. The length of the Lempel-Ziv encoding of x satisfies
$$|LZ(x_1, \ldots, x_n)| \leq (t \log t)(1 + o(1)) \leq C(x) \log C(x)(1 + o(1)).$$
2. For a string x,
$$|x| \geq C(x) \log \frac{C(x)}{4|\Omega|^2}$$
3. Finally, for any S-state compressor F, the compression length satisfies
$$|\mathrm{Comp}_F(x)| \geq C(x) \log \frac{C(x)}{4|\Omega|^2 S^2}.$$

In the next lecture, we will further discuss the inequalities above, which provide the relationship between Lempel-Ziv and FSCs needed to complete our analysis.

Addendum inserted by Madhu

The proofs of the three inequalities claimed above is not hard nor is it crucial for future lectures, so instead of using half a lecture to go over them, let me just add them here and conclude why this implies that the Lempel-Ziv algorithm is a univeral compressor. We start with the second one above, and then turn to the first and finally the last.

Claim 10. *For every string $x \in \Sigma^*$,*

$$|x| \geq (1 - o(1))C(x) \log_{|\Sigma|} C(x).$$

Proof. Let $x = y_1 \circ y_2 \cdots y_t$ with y_i's being distinct elements of Σ^*. To minimize the length of x we might as well assume that the y_i's are lexicographically ordered strings with the shortest ones first. So there are $|\Sigma|$ strings of length 1, $|\Sigma|^2$ strings of length 2 etc., till we get to t distinct strings. The claim now follows from some calculations (omitted here). □

Claim 11. *The length of the Lempel-Ziv encoding of $x = (x_1, \ldots, x_n)$, denoted $|LZ(x)|$, satisfies*

$$|LZ(x)| \leq t \log t(1 + o(1)) \leq C(x) \log C(x)(1 + o(1)),$$

where the $o(1)$ term goes to zero as $n \to \infty$.

Proof. This follows from the fact that Lempel-Ziv produces a split of $x = y_1 \cdots y_t$ with distinct y_i's. It follows that $t \leq C(x)$ and since each y_i is encoded using at most $logt \cdot (1 + o_t(1))$ bits, it follows that the compression length

$$|LZ(x)| \leq t \log t(1 + o_t(1)) \leq C(x) \log C(x)(1 + o_t(1)).$$

It is left as an exercise to see that $o_t(1) = o_n(1)$ for Lempel-Ziv (i.e., that t grows as n grows. □

Claim 12. *For every S-state compressor F, the compression length satisfies*

$$|Comp_F(x)| \geq (1 - o(1))C(x) \log \frac{C(x)}{|S|^2} = (1 - o(1))C(x) \log C(x).$$

Proof. Fix a partition $x = y_1 \cdots y_t$ of x with y_i's being distinct. For states a, b of F, let S_{ab} be the set of indices $i \in [t]$ such that the compressor is in state a at the beginning of parsing y_i and ends in state b after parsing y_i. Let $y'_i \in \{0,1\}^*$ denote the string output by F while parsing y_i. The crux of this proof is the claim that $y'_i \neq y'_j$ for $i, j \in S_{a,b}$, i.e., if the compressor starts and ends in the same state then it must produce distinct outputs on distinct inputs (or else the "compression" is not invertible). Once we have this, we have that $\sum_{i \in S_{a,b}} |y'_i| \geq (1 - o(1))|S_{a,b}| \log_2 |S_{a,b}|$ (from the first claim above). And thus $|Comp_F(x)| = \sum_{a,b} \sum_{i \in S_{a,b}} |y'_i| \geq (1 - o(1)) \sum_{a,b} |S_{a,b}| \log_2 |S_{a,b}|$. Finally by the Cauchy-Schwartz inequality and using the fact that $\sum_{a,b} |S_{a,b}| = t$ we get that $|Comp_F(x)| \geq (1 - o(1)) \sum_{a,b} |S_{a,b}| \log_2 |S_{a,b}| \geq (1 - o(1))t \log(t/|S|^2)$. □

Putting the last two claims together we have that Lempel-Ziv compresses almost as well as an finite state compressor. Combining with Theorem 8 we have the Lempel-Ziv achieves a compression length of at most $(1 + \varepsilon)H(M)$ per input symbol.

5 Solutions to Exercises

Solution to Exercise 4. Suppose that

$$-\frac{1}{\ell}\log p(X_1,\ldots,X_\ell) \to H(M)$$

in probability as $\ell \to \infty$. Fix $\varepsilon > 0$, and define

$$S_\ell = \left\{(\alpha_1,\ldots,\alpha_\ell) \in \Omega^\ell : \left|H(M) - \frac{1}{\ell}\log p(\alpha_1,\ldots,\alpha_\ell)\right| < \varepsilon\right\}.$$

By the definition of convergence in probability, we have

$$1-\varepsilon \leq \Pr[(X_1,\ldots,X_\ell) \in S_\ell] \leq 1$$

for sufficiently large ℓ (Property 1). Rearring terms in the inequality defining S_ℓ yields

$$2^{-\ell(H(M)+\varepsilon)} \leq p(\alpha_1,\ldots,\alpha_\ell) \leq 2^{-\ell(H(M)-\varepsilon)}$$

for $(\alpha_1,\ldots,\alpha_\ell) \in S_\ell$. Properties 2 and 3 in the statement of the AEP now follow easily from these inequalities. □

Solution to Exercise 6. One way to approach this problem is by relating the complexity of compressing hidden Markov models to that of the "Learning Parity with Noise" (LPN) problem from learning theory. Specifically, one can show that if a universal compression algorithm for HMMs could compress n length sequences to an expected length of $(H(M)+\varepsilon)\cdot n$ in $\text{poly}(k, 1/\varepsilon)$ time, then one could solve LPN in subexponential time, which is widely believed to be impossible. □

Solution to Exercise 9. Since preprocessing of M is allowed, the FSC need not compute empirical frequencies. Thus, our version of the Huffman encoding algorithm must simply parse the first ℓ symbols of its input, output the encoding of this string, and repeat. To accomplish this, we can represent the internal states of the FSC as a complete $|\Omega|$-ary tree of depth ℓ, a kind of parsing dictionary for all the strings in Ω^ℓ. The root of the tree is the starting state. The FSC outputs nothing when moving from one internal state to another. When the FSC reaches a leaf, it outputs the encoding of the string along the corresponding root-to-leaf path, and then returns to the root state. Notice that this FSC has

$$\frac{|\Omega|^{\ell+1}-1}{|\Omega|-1} \geq |\Omega|^\ell$$

states. □