6.441 Transmission of Information Mar 9, 2005

Lecture 9

Lecturer: Madhu Sudan *Scribe: Jin Woo Shin*

Today we are going to continue talking about data compression; You can get more detail information of Lempel-Ziv algorithm at the lecture note of Gallager 2/7/1994 dated.

1 Today's topics

- Markov source
- Universal coding algorithm
- Lempel-Ziv algorithm

2 Markov source

Let's assume that there is a Markov process which has a finite state S and whose state transition matrix P is fixed. Also there is a function of output sequence X that only depends on current and one step before states. The sequence of S goes to produce the random source X and we can only observe the output sequence $X's$. This process is called as Markov source or Markovian process. The detailed description is as following:

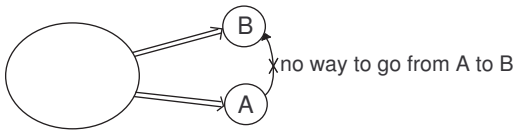
$$\begin{cases} S & = \{1, 2, \ldots, n\} \\ P & = \{P_{ij}\}_{i,j \in S} = \text{Prob. chain goes to state } j \text{ / state } i\text{(given) in one step} \\ f & = S \times S \to \{0, 1\} \quad \text{(output function)} \end{cases}$$

- $(y_0, \ldots, y_t) \in S^t$ s.t. $y_0 \in S$: initial state
- $Pr[y_t = j | y_{t-1} = i \quad y_{t-1} \cdots y_0] = P_{ij}$
- $(x_0, \ldots, x_t) \in \mathscr{X}$ s.t. $x_t = f(y_{t-1}, y_t)$

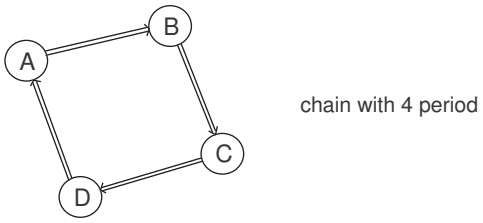
2.1 Notations for Markov source

There are several prime notations for Markov source.

- Source is **reducible** if $\exists i, j \in S$ with no path of prob from i to j, e.g.



- Source is **L periodic** if $\forall C$ paths from i back to i has length divisible by **L**



  We know where the state is after 4 steps.

- Source is **irreducible** if it is not reducible and aperiodic(:= if it is not L periodic for any $L \geq 2$

$$\boxed{\text{"}\textbf{irreducible + aperiodic}\text{"} \Leftrightarrow \text{ergodic}}$$

We don't get into the general concept of ergodic process. However, above relationship will help us set up what we want to do.

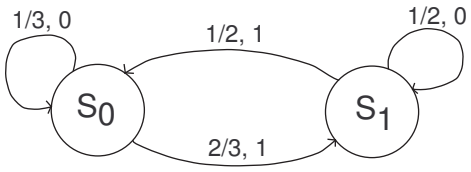
2.2 Entropy rate

We have already learned the definition of Entropy rate $H(\mathscr{X})$ as following:

$$H(\mathscr{X}) := \lim_{t\to\infty} H(X_t|X_{t-1}, \ldots, X_1)$$

This limit does exist in regular markov chain. However, in the case of markov source,it may not. $\{Y_i\}$ definitely follow markov chain and $Y_t|Y_{t-1}$ is fixed for $\forall Y_i$, but X_i does not construct markov chain because X_{t-1} does not contain all of past information of X_i.

Example



In the above Markov source, the observation 0 or 1 does not give us perfect information of states. We can also notice that more past sequence $(0, 1, \cdots, 1)$ gives us more information of states. Generally, it is very hard to calculate entropy rate in this type of problems because whenever we partially observe markov chain, we cannot find correct start state.

[Typical Set Theorem for ergodic Markov source $\mathscr{X}$]

$\forall t$, $\exists T_t \in \{0,1\}^t$

- $\lim_{t\to\infty} \frac{log|T_t|}{t} \to H(\mathscr{X})$
- For $\forall (x_1, \ldots, x_t) \in T_t$,
  $Pr_{(X_1,\ldots,X_t)}[(X_1, \ldots, X_t) = (x_1, \ldots, x_t)] \approx 2^{-H(\mathscr{X})(1\pm\epsilon)t}$
- $\lim_{t\to\infty} Pr_{(X_1,\ldots,X_t)}[(X_1, \ldots, X_t) \in T_t] \to 1$

With this theorem, we can achieve the fact that 1) in typical set, probability distribution of $\{X_i\}$ is almost uniform distribution and each prob. is $\approx 2^{-H(\mathscr{X})(1\pm\epsilon)t}$, 2) we don't need to think out of T_t.

Example Let's think of $\{X_i\}$ i.i.d and satisfies following property.

$$X_i = \begin{cases} 0 & \text{, with probability 0.9,} \\ 1 & \text{, with probability 0.1.} \end{cases}$$

Then, 1) the most probable sequence $(X_1, \ldots, X_t)$ is all 0 sequence, but this sequence is not contained in the typical set. 2) The typical set is roughly uniformly distributed large domain. 3) The size of typical set T_t is virtually lower bound of compression.

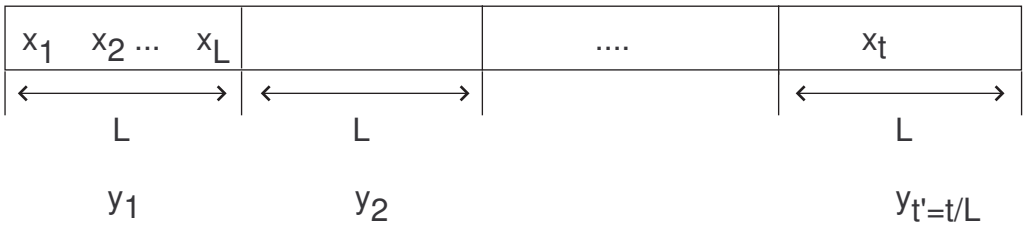
3 Universal Encoding

As we studied in the last lecture, preview the universal encoding. Huffman coding compresses and i.i.d source with a known distribution to its entropy entropy limit. But, what compression can be achieved if it is an unknown distribution? Universal Encoding starts from this idea, and our goal is finding such an uniquely decodable coding algorithm C that satisfies the following property.

$$\forall \mathscr{X},\ \lim_{t\to\infty} \frac{E_{X_i\sim\mathscr{X}}[|C(X_1, X_2, \ldots, X_t)|]}{t} = H(\mathscr{X})$$

I will introduce the 'shabby' encoding as such an example.

3.1 Shabby Algorithm



- At first, divide the data $(x_1, x_2, \ldots, x_t)$ into L sections, and each section is denoted by $y_i \in \{0,1\}^L$, $1 \le i \le t' = t/L$.
- As a first step of encoding, build a dictionary of frequent strings in $\{0,1\}^L$. The frequency constant k is what we will find later, and $I(w)$ indicates the index of w in the dictionary. This step can be formulated as follows,

  count $\leftarrow 0$
  For $w \in \{0,1\}^L$ do
  if $|j|y_j = w| \ge k$, then $Z_w \leftarrow 1$, count $\leftarrow$ count $+ 1, I(w) \leftarrow$ count
  else $Z_w \leftarrow 0$

- The second step of encoding is the real encoding step using the dictionary we built in the previous step. If the section data y_j is in the dictionary, encode it as the index of the dictionary. Otherwise, the encoded data is just the plain data. Also, we add one bit to the encoded data

to indicate its type. This step can be formulated as follows,

$$\begin{aligned}&\text{For } j = 1 \text{ to } t' \text{ do}\\ &\quad w \leftarrow y_j\\ &\quad \text{if } Z_w = 1, \text{ then } u_j \leftarrow (1, I(w))\\ &\quad \text{else } u_j \leftarrow (0, w)\end{aligned}$$

- Therefore, the encoded data is consisted of the dictionary$((Z_w)_{w\in\{0,1\}^L})$ and $u_1, u_2, \ldots, u_{t'}$.

Now, the remaining problem is to determine L, k to minimize the encoded data's length. The following theorem tells about that.

Theorem 1 *If* $k = \frac{t}{L}2^{-H(X)(1+\epsilon)L}$,

$$\lim_{\epsilon\to 0}\lim_{L\to\infty}\lim_{t\to\infty}\frac{E[|Shabby_{L,k}(X_1, X_2, \ldots, X_t|]}{t} \to H(\mathscr{X})$$

Proof Idea The size of dictionary$(\{Z_w\})$ is at most 2^L, because there are 2^L w's. And, the AEP says the size of the dictionary is bounded by $2^{H(\mathscr{X})(1+\epsilon)L}$. Therefore, if $Z_{u_j} = 1$, the length of u_j is $H(\mathscr{X})(1+\epsilon)L + 1$, and the total length of u_j, where $Z_{u_j} = 1$ is $\frac{t}{L}(H(\mathscr{X})(1+\epsilon)L + 1)$. Also, from AEP, we can know the total length of u_j, where $Z_{u_j} = 0$ is $\delta\frac{t}{L}(L+1)$. In sum, the length of the encoded data is at most $2^l + \frac{t}{L}(H(\mathscr{X})(1+\epsilon)L + 1) + \delta\frac{t}{L}(L+1)$. The dominant term of them is $\frac{t}{L}(H(\mathscr{X})(1+\epsilon)L + 1)$, and we can lead the result. ■

As we see the above theorem, this encoding scheme is not elegant and practical because it is not easy to find L, k from an unknown distribution.

3.2 LEMPEL-ZIV CODING

We now describe another more elegant and simple scheme for universal encoding. The algorithm defines simply as follows,

- (Parsing) Parse the source data $(x_1, x_2, \ldots, x_t)$ into t' sections $(y_1, y_2, \ldots, y_{t'})$ such that

$$\begin{aligned}&\forall j, \forall j' < j,\ y_j \neq y_{j'}\\ &\forall j, \exists j' < j,\ y_j = y_{j'}b\ (b \in \{0,1\})\end{aligned}$$

- (Encoding) Encode each section y_j $(1 \le j \le t')$ to (j', b).

We just touch the overview of analysis that this encoding scheme is a good compressor. There are two ideas to prove the statement.

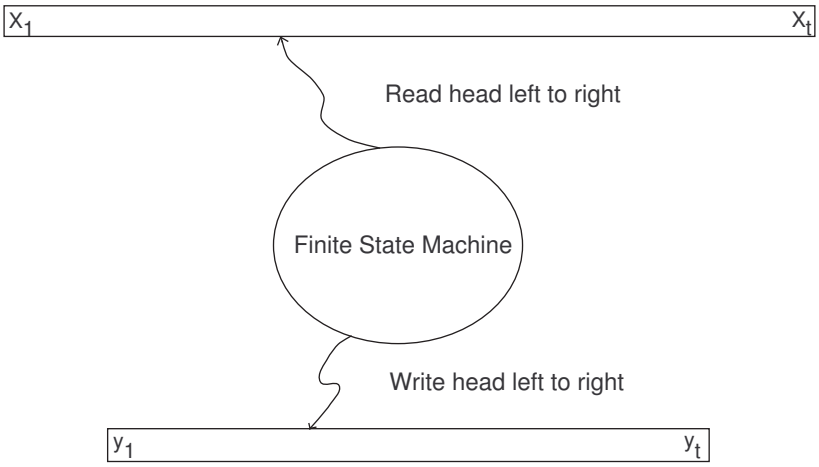
- Lempel Ziv compression is no worse than any finite-state compressor.

$$\forall S \quad \lim_{t\to\infty}\left\{\frac{LZ(\text{comp.})}{Cs(\text{comp.})}\right\} \le 1$$

- Finite state compressors are not too bad.

Sketch of Proof

1. What is finite state machine?



$$\begin{cases} S = \{1, \ldots, s\} \\ \delta(s, X_i) \to s' \\ W(s, X_i) \to y \in \{0,1\}^* \text{ (we can write nothing or many strings at a time.)} \end{cases}$$

Finite state machine consists of finite number states. In each state, machine reads the input sequences $\{X_i\}$, and based on the current state and input value, it determines next state and output sequences $\{Y_i\}$. All procedures in finite state machine are deterministic.

2. The finite state compression is not too bad. For example, 'Shabby' is $\approx 2^L$ state compressor, and we checked in the previous section that it is not too bad.

3. Lemple Ziv is no worse than any finite state compressor.

   Let $c(X_1, \ldots, X_t)$ be the maximum value of t' such that $\exists\ y_1, y_2, \ldots, y_{t'}, X_1X_2 \ldots X_t = y_1y_2 \ldots y_{t'}$ and y_i are all distinct. In other words, it can be interpreted as the largest number of distinct strings into which $X_1X_2 \ldots X_t$ can be parsed.

   Claim 2 *if* $c(X_1, \ldots, X_t) = t'$ *then* $t \geq t' \log\left(\frac{t'}{4}\right)$

   Claim 3 *if* $c(X_1, \ldots, X_t) = t'$ *then for every* S *state compressor* Cs*,*

   $$|Cs(X_1, \ldots, X_t)| \geq t' log\left(\frac{t'}{4S^2}\right) \textit{ (by the Jensen's inequality)}$$

   Claim 4 *if* $c(X_1, \ldots, X_t) = t'$ *then*

   $$|C_{LZ}(X_1, \ldots, X_t)| \leq t' log\left(t'(1 + o(1))\right)$$

■

Lempel-Ziv algorithm is practically very elegant, however analyzing it is very messy and complicate.