

information theory; source coding

Linda Doyle

codes again

- so we looked at two very different coding mechanisms
- Huffman is based on statistical properties of the file to be coded and leads to variable length codes.
- LZW is a dictionary technique. In dictionary-based data compression techniques, a symbol or a string of symbols generated from a source alphabet is represented by an index to a dictionary constructed from the source alphabet.

- let's do a comparison of the two coding schemes we looked at.

an example

- aaaaaaaaaaaaaaaaaaabbcccccccccccccccc
- I am using this string as a way of looking at the two different techniques
- Which technique would best suit this string of characters?

using Huffman

- To do a Huffman on this I count all the letters and find that a occurs 50% of the time, b 25% of the time and c and d occur 12.5% of the time.
- I can create a Huffman code using the process we learned already and get codes a=1, b=01, c=001 and d=000

using Huffman

- The average length of the code for the string we sent is 2.25 bits
- The number of bits sent in total for our message is 90 bits in total
- The entropy is
$$-[0.5\log(0.5) + 0.25\log(0.25) + 0.125\log(0.125) + 0.125\log(0.125)]$$
$$=1.75$$

if we had been sending a longer message we can get much closer to entropy - this is actually part of the message in Shannon's lossless coding theorem

using LZW

- we can go through the process we developed in class and we can get
- 0,4,5,6,7,7,1,1,10,11,12,2,14,3,17,17
- if we have 5 bits per code transmitted we transmit 16×5 bits which is 80 bits for the message rather than 90 with Huffman

intuitive comments

- The strategy of the dictionary coding is to build a dictionary that contains frequently occurring symbols and string of symbols.
- In other words the dictionary mechanism inherently tends to take into account that symbols have some dependence on previous symbols.

the example redone

- abacadaabbacbaacdaaaddababcbaabdaaaabdcb
- the statistics are the same here so the Huffman code is as efficient as before
- however there are less numbers of repeated strings and the LZW performs less well as expected - we need to transmit 26 different codes and using 5 bits per code we get 130 bits for the message.
- you can check this for yourselves

- LZW does not use the statistics of the source like Huffman does but it does manage to capture some idea of memory
- Let us now go back and look more formally at the statistics of a source and take memory into account.

Markov Chains

Definition A discrete stochastic process X_1, X_2, \dots is said to be a *Markov chain* or a *Markov process* if for $n = 1, 2, \dots$

$$\begin{aligned} \Pr(X_{n+1} = x_{n+1} | X_n = x_n, X_{n-1} = x_{n-1}, \dots, X_1 = x_1) \\ = \Pr(X_{n+1} = x_{n+1} | X_n = x_n) \end{aligned} \quad (4.2)$$

for all $x_1, x_2, \dots, x_n, x_{n+1} \in \mathcal{X}$.

In this case, the joint probability mass function of the random variables can be written as

$$p(x_1, x_2, \dots, x_n) = p(x_1)p(x_2|x_1)p(x_3|x_2) \cdots p(x_n|x_{n-1}). \quad (4.3)$$

Definition The Markov chain is said to be *time invariant* if the conditional probability $p(x_{n+1}|x_n)$ does not depend on n ; that is, for $n = 1, 2, \dots$

$$\Pr\{X_{n+1} = b | X_n = a\} = \Pr\{X_2 = b | X_1 = a\} \quad \text{for all } a, b \in \mathcal{X}. \quad (4.4)$$

We will assume that the Markov chain is time invariant unless otherwise stated.

If $\{X_i\}$ is a Markov chain, X_n is called the **state** at time n . A time-invariant Markov chain is characterized by its initial state and a *probability transition matrix* $P = [P_{ij}]$, $i, j \in \{1, 2, \dots, m\}$, where $P_{ij} = \Pr\{X_{n+1} = j | X_n = i\}$.

If the probability mass function of the random variable at time n is $p(x_n)$, the probability mass function at time $n + 1$ is

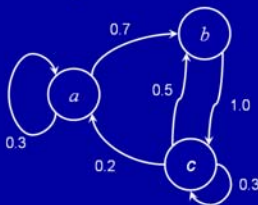
$$p(x_{n+1}) = \sum_{x_n} p(x_n) P_{x_n, x_{n+1}}. \quad (4.5)$$

acknowledgement

- next few slides taken from Jorgen Ahlberg
- we are interested in first order Markov Sources – symbol depends only on the previous symbol

The Markov Source

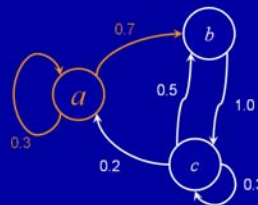
- A symbol depends only on the previous symbol, so the source can be modelled by a state diagram.



A ternary source with alphabet $\mathcal{A} = (a, b, c)$.

The Markov Source

- Assume we are in state a , i.e., $X_k = a$.
- The probabilities for the next symbol are:



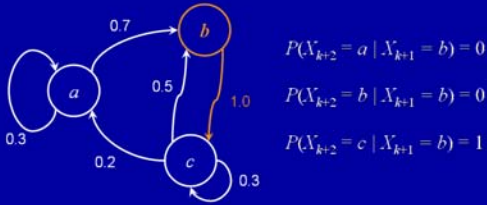
$$P\{X_{k+1} = a | X_k = a\} = 0.3$$

$$P\{X_{k+1} = b | X_k = a\} = 0.7$$

$$P\{X_{k+1} = c | X_k = a\} = 0$$

The Markov Source

- So, if $X_{k+1} = b$, we know that X_{k+2} will equal c .



Entropy for a Markov Source

The entropy for a state S_k can be expressed as

$$H(S_k) = \sum_{l=1}^r P_{kl} \log \frac{1}{P_{kl}}$$

Averaging over all states, we get the entropy for the Markov source as

$$H_M = \sum_{k=1}^r P(S_k) H(S_k)$$

transition probability from state k to state l

using Huffman with a Markov Source

B_2	$p(B_2)$	Codeword
aa	0.45	0
bb	0.45	10
ab	0.05	110
ba	0.05	111

$R = 0.825$ bits/character

An example:

Original Data: a a a a a a b b b b b b b b b b b a a a a
 Compressed Data: 0 0 0 110 10 10 10 10 10 10 10 0 0

Note that 20 bits are used to represent 24 characters --- an average of 0.83 bits/character.

so end of compression for the moment

- you learned how to measure information
- relate this to compression - how much something can be compression
- you looked at entropy encoding - encoding based on statistical information of the source - Huffman
- you mainly looked at Huffman for memoryless sources but had a quick look at sources with memory - Markov Chains
- you looked at universal encoding