Information, Codes and Ciphers

Summary Notes *

Hao Ren

November 10, 2020

*Based on MATH3411, UNSW 20T3.

Contents

0 All TODOs

Updated before start the part *3.4.1 Definition of Arithmetic Coding*.

0.1 Unfinished Contents

Finish this part at first and then add more items into *0.2 Example and Diagrams* feature if needed.

- 3.1.4 Decision Trees
- 3.2.4 Extensions of Huffman Coding
- 3.3.3 Huffman Coding for Stationary Markov Sources

0.2 Examples and Diagrams

- 3.1.3 Comma Codes
- 3.2.1 Huffman Coding
- 3.2.3 Radix r Huffman Codes
- 3.3.1 Definition of Markov Sources

0.3 Proofs

- Theorem 3.1 The Kraft-McMillan Theorem
- Theorem 3.2 Minimal UD-codes
- Theorem 3.3 Huffman Code Theorem
- Proposition 3.4 Knuth

3 Compression Coding

3.1 Variable Length Encoding

3.1.1 Definition

a source S	with q source symbols	$s_1, s_2, \cdots, s_q$,
	with probabilities	$p_1, p_2, \cdots, p_q$,
encoded by a code C	with q codewords	$c_1, c_2, \cdots, c_q$,
	of lengths	$l_1, l_2, \cdots, l_q$.

- with a radix r codewords,
- variable length codes,
- not channel noise for source coding.

3.1.2 UD and I-code

A code C is

UD uniquely decodable codes if it can always be decoded unambiguously,

I-code instantaneous if no codeword is the prefix of others.

3.1.3 Comma Codes

The standard comma code of length n is

- a code which every codeword has length $\leq n$,
- a code which every codeword contains at most one 0,
- and if a codeword contains 0 then 0 must be the final symbol in the codeword.

3.1.4 Decision Trees

3.1.5 The Kraft-McMillan Theorem

Theorem 3.1 (The Kraft-McMillan Theorem)**.**
A UD-code of radix r with q codewords $c_1, c_2, \cdots, c_q$ of lengths $l_1 \leq l_2 \leq \cdots \leq l_q$ exists
if and only if *an I-code with the same parameters exists*
if and only if

$$K = \sum_{i=1}^{q} \frac{1}{r^{l_i}} \leq 1.$$

3.1.6 Length and Variance

The expected or **average length** of codewords is given by

$$L = \sum_{i=1}^{q} p_i l_i$$

and the **variance** is given by

$$V = \sum_{i=1}^{q} p_i l_i^2 - L^2.$$

Our aim is to minimise L for a given source S and, if more than one code C gives this value, to minimise V.

Theorem 3.2 (Minimal UD-codes)**.**
Let C be a UD-code with minimal expected length L for the given source S. Then, after permuting codewords of equally likely symbols if necessary,

- $l_1 \leq l_2 \leq \cdots \leq l_q$ *and*
- $l_{q-1} = l_1$.

Furthermore, if C is instantaneous, then

- c_{q-1} *and* c_q *differ only in their last place.*

If C is binary, then

- $K = \sum_{i=1}^{q} 2^{-l_i} = 1$.

3.2 Huffman's Algorithm

> Huffman's algorithm for computing minimum-redundancy prefix-free codes has almost legendary status in the computing disciplines. Its elegant blend of simplicity and applicability has made it a favourite example in algorithms courses, and as a result, it is perhaps one of the most commonly implemented algorithmic techniques. [1]

3.2.1 Huffman Coding

In 1952, David A. Huffman [2] published a new lossless data compression method as an Sc.D student at MIT. Here is a demonstration to compute Huffman prefix-free code which is provided by Princeton University in the course COS226 [3]:

- Count character frequencies p_s for each symbol s in file.
- Start with a forest of trees, each consisting of a single vertex corresponding to each symbol s with weight p_s.
- Repeat:
  - select two trees with min weight p_1 and p_2
  - merge into single tree with weight $p_1 + p_2$

Applications JPEG, MP3, MPEG, PKZIP.

Theorem 3.3 (Huffman Code Theorem).
For the given source S, the Huffman algorithm produces a minimum average length UD-code which is an instantaneous code.

Proposition 3.4 (Knuth).
For a Huffman code created by the given algorithm, the average code word length is sum of all the probabilities at child nodes.

3.2.2 Properties of Huffman Codes

1. The place high strategy always produces a minimum variance Huffman code .
2. If there are 2^n equally likely source symbols then the Huffman code is a block code of length n.
3. If for all j, $3p_j \geq 2\sum_{k=j+1}^{q} p_k$ then the Huffman code is a comma code.
4. Small changes in the pi can change the Huffman code substantially, but have little effect on the average length L. This effect is smaller with smaller variance.

3.2.3 Radix r Huffman Codes

For r radix Huffman codes, which $r \geq 3$, we have a better strategy by adding some dummy variables [4].

For a r radix encoding, the procedure is similar except r least probable symbols are merged at each step. Since the total number of symbols may not be enough to allow r variables to be merged at each step, we might need to add some dummy symbols with 0 probability before constructing the Huffman tree.

How many dummy symbols need to be added? Since the first iteration merges r symbols and then each iteration combines $r-1$ symbols with a merged symbols, if the procedure is to last for k (some integer number of) iterations, then the total number of source symbols needed is $1 + k(r-1)$. So before beginning the Huffman procedure, we add enough dummy symbols so that the total number of symbols look like $1 + k(r-1)$ for the smallest possible value of k.

Remark.

1. *If more than two symbols have the same probability at any iteration, then the Huffman coding may not be unique (depending on the order in which they are merged). However, all Huffman codings on that alphabet are optimal in the sense they will yield the same expected code length.*
2. *One might think of another alternate procedure to assign small code lengths by building a tree top-down instead, e.g. divide the symbols into two sets*

with almost equal probabilities and repeating. While intuitively appealing, this procedure is suboptimal and leads to a larger expected code length than the Huffman encoding. You should try this on the symbol distribution described above.

3.2.4 Extensions of Huffman Coding

3.3 Markov Sources

3.3.1 Definition

A finite-state Markov chain is a sequence $S_0, S_1, \cdots$ of discrete random symbols from a finite alphabet, S. There is a probability mass function (pmf) $q_0(s), s \in S$ on S_0, and there is a conditional pmf $Q(s|s')$ such that for all $m \geq 1$, all $s \in S$, and all $s' \in S$,

$$\mathrm{P}r(S_k = s|S_{k-1} = s') = \mathrm{P}r(S_k = s|S_{k-1} = s', \cdots, S_0 = s_0) = Q(s|s')$$

There is said to be a *transition* from s' to s, denoted $s' \to S$, if $Q(s|s') > 0$. [5]

3.3.2 Transition Matrix

The matrix $M = (p_{ij})$ is called the transition matrix of the Markov process, which could be displayed as (from s_j) $\to (p_{ij}) \to$ (to s_i).

Remark.

- $P(s_1|sj) + P(s_2|sj) + \cdots + P(s_q|sj) = 1$
- $p_{1j} + p_{2j} + \cdots + p_{qj} = 1, \text{ for } j = 1, \cdots, q$

$$\begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1j} \\ p_{21} & p_{22} & \cdots & p_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ p_{i1} & p_{i2} & \cdots & p_{ij} \end{bmatrix}$$

3.3.3 Huffman Coding for Stationary Markov Sources

3.4 Arithmetic Coding

3.4.1 Definition

Arithmetic coding is very efficient and approaches the entropy limit faster than Huffman coding without any contradiction because the arithmetic coding is not a UD-code. The idea is to assign a subinterval of $[0, 1) \subseteq \mathbb{R}$ to the message and successively narrow this subinterval down as each symbol is encoded. The message must end with a stop symbol $\bullet$, which could be displayed as

$$s_a s_b s_c s_d s_e \cdots s_n \bullet,$$

and after the subinterval corresponding to the message plus $\bullet$ is found, then any suitable *single number* in the subinterval is transmitted – this is the actual code number or codeword.

3.4.2 Encoding

3.4.3 Decoding

3.5 Dictionary Methods

3.5.1 Encoding

3.5.2 Decoding

3.6 Other Types of Compression

References

[1] A. Moffat, "Huffman coding," *ACM Comput. Surv.*, vol. 52, no. 4, August 2019.

[2] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the Institute of Radio Engineers*, vol. 40, no. 9, pp. 1098–1101, September 1952.

[3] R. Sedgewick and K. Wayne, "5.5 data compression," https://www.cs.princeton.edu/courses/archive/fall20/cos226/lectures/55DataCompression.pdf, 2020.

[4] A. Singh, "Lecture 8: Source coding theorem, huffman coding," http://www.cs.cmu.edu/~aarti/Class/10704/lec8-srccodingHuffman.pdf, 2012.

[5] L. Zheng and R. Gallager, "Chapter 2 coding for discrete sources," https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-450-principles-of-digital-communications-i-fall-2006/lecture-notes/book_2.pdf, 2006.