Study Guide to Data Compression

Lawrence Searcy

This study guide was written to fullfill the course requirements for CMPE 263 Data Compression. All quoted material is taken directly from *Data Compression and Image Coding* by Professor Glen Landon, UCSC.

Index

CHAPTER 1

Introduction to Data Compression

Objective: Learn the basic building blocks of data compression and early coding schemes including: probability theory, codes in general, and bit serial coding.

Symbols

- ***A*** - An alphabet consisting of ordered symbols (e.g Alphabet *A*:{x,y,z} has 3 symbols a_1 (x), a_2 (y), and a_3 (z).
- **a_j** - One symbol in an alphabet. Not to be confused with Symbol s_k which may be repeated in the input string. Symbol a_j can only occur once in the alphabet.
- **Cd(S)** - Compressed output of input string *S*. Also called code string of string *S*.
- ***cl*(s_k)** - Code word length (in bits) of symbol s_k (e.g. symbol a's code word is 101 and has a code word length of 3). Also called codelength of s_k.
- **CL(S)** - Code word length of the entire string *S* after compression (i.e. number of bits in compressed string). Used to calculate the compressed length of the string *S* without actually encoding it. Also called codelength of string *S*.
- **Ct(s_k)** - The number of times symbol s_k occurs in string *S*.
- **CW(s_k)** - Code word corresponding to symbol s_k (e.g. symbol a's code word is 101).
- ***D*** - probability distribution.
- ***il*(p)** - Ideal (or optimal) code word length of symbol s_k in bits where p = p(s_k). Also called self-information.
- **p(s_k)** - Probability of symbol s_k.
- **N** - The total number of symbols s_k in input string S.
- **S** - Input string or file fed to compression algorithm.
- **s_k** - One symbol in an input string *S*.

Definitions

Bit Rate - The average code word length or average number of bits/symbol. See **bps.**

bps - Bits per symbol = bit rate.

Code String - "A concatenation of code words." e.g. encoding A's code word [101] and B's code word [110] gives code string 101110.

Code Table - An alphabet to code or code to alphabet translation table that assigns a code word to each symbol.

Code word - A bit string of 0s and 1s.

Compression Performance - The ratio of the size of the uncompressed file to the compressed file or the number of bits per symbol (Author specified!).

Discrete Probability Distribution - A probability distribution which has the following 2 properties:
1) The probability of each event or symbol lies between 0.0 and 1.0 including the endpoints (inclusive).
2) The sum of all the probabilities is equal to 1.

Empirical Probability - Actual probability determined by counting how many times event A occurred in N repetitions. Also called relative frequencies.

FVL - Fixed to variable length encoding (e.g. Morse code, Huffman codes).

Ideal Code Length - See **Self-Information**. Also called **Optimal Code Length**.

M-ary alphabet - An alphabet *A* with M distinct symbols. i.e. alphabet $A = (a_1, a_2, ..., a_j, ..., a_m)$.

Manchester Code (MC) - A bit coding scheme that encodes both the data signal and clocking in the same signal.

Memoryless - The probability of the current symbol s_k does not depend on the previous symbol. Each symbol is statistically independent from one another. It can also be stated as "the probability of the next symbol cannot be affected by the current symbol" or the probability for the symbol does not change.

Optimal Code word Length - See **Ideal Code Length**.

Optimal Probability - The ideal probability for symbol s_k given the optimal code word length.

Phased-Locked Loop (PLL) - A hardware circuit that uses the incoming signal to synchronizes to the original clock signal so that the incoming data can be read. Most PLLs generally only use the beginning of the incoming signal.

Prefix Code - "A set of code words where the code words are in a 1:1 correspondence to the leaves of a full binary tree." and all code words have the prefix property.

Prefix Property - A longer code word's first K bits will not be equal to any shorter code word. This can also be stated as "No code word is the prefix of another code word".

Pulse Code modulation - A method by which a continuous signal is represented as digital data.

Relative Frequency - See **Empirical Probability**.

Return-To-Zero (RZ) - A bit coding scheme that recovers the original clocking only at each 1 bit.

Sample Space - *By definition* is the set of all possible outcomes of a probability trial.

Self-Information - The theoretical length of a code word determined by its symbol's probability. If the symbols's probability is 1/2 then by definition one bit of self information has occurred. Also called **Ideal Code Length**.

Statistically Independent - Occurs between two events when each event does not affect the outcome of the other event.

VFL - Variable to Fixed length encoding.

Formulas

Probability of event A given that event B has already occurred $P\langle A|B\rangle = \frac{P(A \cap B)}{P(B)}$

Ideal Code Length $il(p) = \log_2 \frac{1}{(p)} = -\log_2 (p)$ where $p = p(s_k)$.

Conversion from $\log_2$ to $\log_{10}$ $\log_2(X) = \frac{\log_{10}(X)}{\log_{10}(2)} = \log_{10}(X) \cdot 3.32192$

Codelength of string S $CL(S) = \sum_k N \times p(s_k) \times cl(s_k)$ where N x $p(s_k)$ is equal to the number of times symbol s_k is in string S.

Empirical probability $= \frac{Ct(s_k)}{N}$

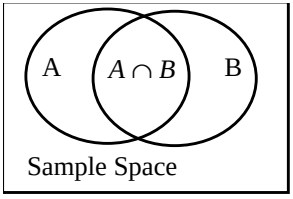
Bits/symbol for CL(S) $bps = \frac{CL(S)}{N} = \sum_k p(s_k) \times cl(s_k)$

Optimal probability $= 2^{-cl(s_k)}$

Comments

General Ideas on Probability Theory

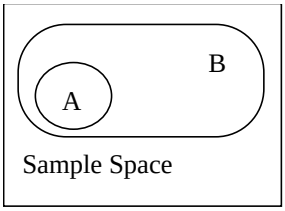
When events A and B are statistically dependent



P(A|B) is the probability of event A given that event B has already occurred. Knowledge that event B has occurred implies that the outcome of the experiment lies in the set B.

$P\langle A|B\rangle = \frac{P(A \cap B)}{P(B)}$ $P\langle B|A\rangle = \frac{P(B \cap A)}{P(A)}$

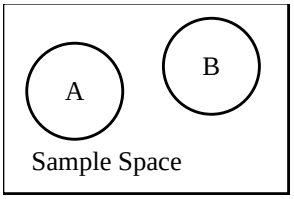
$P\langle A|B\rangle \cdot P(B) = P(A \cap B) = P(B \cap A) = P\langle B|A\rangle \cdot P(A)$



When one event is completely within the probability sample space of another event $P(A \cap B) = P(A)$ and $P\langle A|B\rangle = \frac{P(A \cap B)}{P(B)}$ reduces to $P\langle A|B\rangle = \frac{P(A)}{P(B)}$.

When events A and B are statistically independent

When the two events are statistically independent then *by definition* $P(A \cap B) = P(A) \cdot P(B)$ and $P\langle A|B\rangle = P(A)$



Independent Events

*H*ow can you tell when two events A and B are independent? First, you must figure out the probabilities of event A and event B. Next, figure out the probability of event A and event B occurring at the same time. Is it equal to $P(A) \cdot P(B)$? If so, then the two events are independent. Is it equal to 0? Then the two events are mutually exclusive. Is it equal to some other value? Then the two events are dependent.

General Ideas on Data Compression

Best Compression

*T*he more frequent the symbol, the shorter the code word. Another way of saying this is that "shorter code words are assigned to the most popular symbols." This is because the "Best compression occurs when the longest code words are assigned to the least probable symbols." These statements explain why data compression is based on probabilities and relative frequencies. Data compression either guesses which symbol will be more frequent (probabilities) or actually counts the number of symbols (relative frequencies).

Skewed Distributions

*C*ompression comes only from skewed distributions. If the symbols are equally likely then there is no compression.

Code Tree

Given a code tree, the code words for any symbol at a leaf corresponds to the branch labels from the root to the leaf.

Figure 2, Chapter 1

*L*ook at figure 2 on page 13. This figure gives a functional views of the complete coding and decoding algorithms for a fixed to variable length (FVL) compression

system. If you draw a horizontal line just under “media” and “Cd(s)” you will separate the encoding stages from the decoding stages. It follows the general substitution methods described below. Let’s take a look at how this works.

Encoding

Given input string (or file) *S*, remove the first symbol s_k. Look up the symbol’s code word in the lookup table. From the table you get the symbol’s code word $CW(s_k)$ and the length of the code word, $cl(s_k)$. Next we shift the buffer by $cl(s_k)$ bits and tack on $CW(s_k)$ to Cd(S). We repeat this process until there are no more symbols in the input string to process.

*T*he diagram then shows the code string being transmitted over a media, say to another computer, where it is stored in a buffer. If we incrementally accept the code string into a new buffer we can decode it.

Decoding

*W*e accept the first k bits into the new buffer where k is equal to the number of bits of the longest code word in the encoding table. We then use the code word in the new buffer to index into the decoding table. This returns the symbol corresponding to its codeword and the length of the codeword. We need the length of the codeword since this can vary between code words [e.g codeword for a is 01 (2 bits) and code word for b is 101 (3 bits)]. Next we shift out the number of bits of the previously decoded codeword and add in the exact same number of bits from the code string to be decoded. Notice that we again have the same number of bits as the longest code word in the encoding table. We repeat this process until the code string is completely decoded.

Algorithms

General Substitution Method

General substitution method encoding: read the symbol s_k, then look its code word up in the translation table. Add symbol s_k’s code word to the output code string Cd(S). Repeat.

*G*eneral substitution method decoding: read in bits until the number of bits in your buffer equals the number of bits in the longest code word. Then check the lookup table for the decode symbol and the number of bits to remove from the front of the input string just read in. Repeat.

Examples

Got to get around to this! There should be an example for every formula!

CHAPTER 2

The Huffman Coding Algorithm

Objective: Learn the huffman encoding technique.

Formulas

Code Length of String S $CL(S) = \sum_{k=0}^{N} cl(s_k)$

Average Code Length for a Huffman Tree $= \frac{\sum internalNodes}{\sum leaves}$

Comments

General Comments on Huffman encoding

Algorithms

General Substitution Method

General substitution method encoding: read the symbol s_k, then look its codeword up in the translation table. Add symbol s_k's codeword to the output. Repeat.