Data Compression and Quantization

by

Benjamin David du Toit

Submitted in partial fulfilment of the requirements of the degree

Magister Scientiae Mathematical Statistics

In the Department of Statistics

University of Pretoria

Pretoria

February 2014

Summary

Title: Data Compression and Quantization

Name: Benjamin du Toit

Supervisor: Dr. Frans Kanfer

Co-supervisor: Mr. Sollie Millard

Department: Statistics

Degree: MSc. Mathematical Statistics

Index

List of Figures pg.6
List of Tables pg.8
List of Codes pg.9
Abstract: Data Compression and Quantization pg.10

Chapter One Introduction to Data and Data Compression pg.13

1.1 Some Introductory Concepts

1.2 An Introduction to Data Compression

1.3 Theoretical Aspects of Data Compression

1.4 Study 1: Huffman Encoding applied to Bernoulli Data

1.5 Study 2: Huffman Encoding applied to Bernoulli data with Dependence Structure

1.6 Conclusions of Chapter One

Chapter Two Information Theory pg.55

2.1 What is Information?

2.2 Information in a String

2.3 An Intuitive Introduction to Uncertainty and Entropy

2.4 The Derivation of Entropy

2.5 The Kraft inequality for a Binary Codebook

2.6 Entropy of Random Vectors of Dimension m

2.7 Independent $Y_i's$

2.8 Conditionally Dependent $Y_i's$

2.9 The Chain Rule for Entropy

2.10 Mutual Information and Information Inequality

2.11 Blocking Random Variables

2.12 Alternative Entropy Derivation

2.13 Conclusions of Chapter Two

Chapter Three Scalar Quantization pg.78

3.1 Quantization in Data Compression

3.2 Scalar and Vector Quantization

3.3 Quantizer Design

3.4 Scalar Quantization

3.5 Distortion Measures of Scalar Quantization

3.6 Three Methods of Scalar Quantization

3.7 High Code Rate Estimate of Distortion

3.8 Conclusions of Chapter Three

Chapter Four Vector Quantization pg.107

4.1 Vector Quantization

4.2 Scalar Quantization to Vector Quantization

4.3 Vector Distortion

4.4 Rate and Distortion Minimization

4.5 Efficacy of the Vector Quantizer

4.6 Generalized Lloyd Vector Quantization: The Optimal Problem

4.7 Two Variations on the Generalized Lloyd Algorithm

4.8 Conclusions of Chapter Four

Chapter Five Scalar Transformations and Picture Study pg.131

5.1 Complexity

5.2 Bit Allocation

5.3 Techniques that Increase Scalar Quantization Gains

5.4 Linear Transformations

5.5 Picture Study

5.6 Conclusions of Chapter Five

Final Conclusion and Summary pg.158

Appendix pg.160

Bibliography pg.164

List of Figures

Figure 1.1 Analog-to-Digital Converter.

Figure 1.2 3-by-10 Pixel Illustration (unquantized).

Figure 2.3 3-by-10 Pixel Illustration (quantized).

Figure 1.4 Binary Tree.

Figure 1.5 The Entropy Function $H(\boldsymbol{Y}; p, m)$ Calculated for Varying Block Sizes.

Figure 1.6 The Scaled Entropy Function.

Figure 1.7 Entropy vs. Code Rate for Different Block Sizes.

Figure 1.8 Decrease in Code Rate for Increasing Heterogeneity.

Figure 1.9 Comparison of the Entropy of Independent and Dependent Bernoulli Blocks.

Figure 3.1 Scalar Quantization Applied to a Continuously Sampled Signal.

Figure 3.2 Regular Scalar Quantizer.

Figure 3.3 Midrise and Midtread Scalar Quantizer.

Figure 3.4 Compressor, Uniform Quantizer, Expander.

Figure 3.5 The Logarithmic Compressor Curve based on the μ-law.

Figure 3.6 The Logarithmic Expander Curve based on the μ-law.

Figure 3.7 The Logarithmic Compressor Curve based on the A-law.

Figure 3.8 The Logarithmic Expander Curve based on the A-law.

Figure 4.1 A Voronoi Tessellation.

Figure 4.2 Regular Vector Quantization.

Figure 4.3 Rate – Distortion Function.

Figure 5.1 Reflection of Image over Each of its Edges.

Figure 5.2 A 24 Bit RGB Format Photo

Figure 5.3 Zooming into the Photo

Figure 5.4 Process of Image Compression.

Figure 5.5 Histogram of the AC and DC Coefficients.

Figure 5.6 Comparison of the Compressed Image with its Uncompressed Counterpart.

Figure 5.7 The Error Image.

List of Tables

Table 1.1 Example of Huffman Encoding.

Table 1.2 Expected Length based on Huffman Encoding.

Table 1.3.1 Block of Size $m = 2$ and $P(Y_i = 1) = 0.6$.

Table 1.3.2 Block of Size $m = 2$ and $P(Y_i = 1) = 0.7$.

Table 1.3.3 Block of Size $m = 2$ and $P(Y_i = 1) = 0.8$.

Table 1.3.4 Block of Size $m = 2$ and $P(Y_i = 1) = 0.9$.

Table 1.4.1 Block of Size $m = 3$ and $P(Y_i = 1) = 0.6$.

Table 1.4.2 Block of Size $m = 3$ and $P(Y_i = 1) = 0.7$.

Table 1.4.3 Block of Size $m = 3$ and $P(Y_i = 1) = 0.8$.

Table 1.4.4 Block of Size $m = 3$ and $P(Y_i = 1) = 0.9$.

Table 1.5.1 Block of Size $m = 4$ and $P(Y_i = 1) = 0.6$ and Huffman Encoding.

Table 1.5.2 Block of Size $m = 4$ and $P(Y_i = 1) = 0.7$ and Huffman Encoding.

Table 1.5.3 Block of Size $m = 4$ and $P(Y_i = 1) = 0.8$ and Huffman Encoding.

Table 1.5.4 Block of Size $m = 4$ and $P(Y_i = 1) = 0.9$ and Huffman Encoding.

Table 1.6 Huffman Encoding: A Comparison of Code Rates.

Table 1.7 A Comparison of Code Rates with Dependency Structure for $m = 2$.

Table 1.8 A Comparison of Code Rates with Dependency Structure for $m = 3$.

Table 5.1 Eight by Eight Discrete Cosine Transform Matrix.

Table 5.2 Luminance Quantization Matrix.

Table 5.3 Chrominance Quantization Matrix.

Table 5.4 (3×3) Block Sub-image Pixel Matrix.

Table 5.5 Sample Correlation Matrix.

Table 5.6 DCT and KLT Transformed Coefficients.

Table 5.7 Comparison of Compression Ratios.

List of Codes

Code 1.1 Pseudo Code for the Binary Tree Encoding Method.

Code 1.2 SAS Code for Huffman Encoding.

Code 1.3 SAS Code for the Comparison of Indep. and Depen. Bernoulli Blocks.

Code 5.1 SAS 9.2 Code: RGB Transform.

Code 5.2 SAS 9.2 Code: Discrete Cosine Transform.

Code 5.3 SAS 9.2 Code: Quantization Matrix.

Code 5.4 SAS 9.2 Code: Image Reconstruction (Quantization Matrix).

Code 5.5 SAS 9.2 Code: RGB Transform.

Code 5.6 SAS 9.2 Code: Discrete Cosine Transform.

Code 5.7 SAS 9.2 Code: Quantization Matrix with Scaling Factor $\rho = 5$.

Code 5.8 SAS 9.2 Code: Creating the Error Image.

Abstract: Data Compression and Quantization

Data Compression

Due to limitations in data storage and bandwidth, data of all types has often required compression. This need has spawned many different methods of compressing data. In certain situations the fidelity of the data can be compromised and unnecessary information can be discarded, while in other situations, the fidelity of the data is necessary for the data to be useful thereby requiring methods of reducing the data storage requirements without discarding any information.

The theory of data compression has received much attention over the past half century, with some of the most important work done by Claude E. Shannon in the 1940's and 1950's and at present topics such as Information and Coding Theory, which encompass a wide variety of sciences, continue to make headway into the interesting and highly applicable topic of data compression.

Quantization

Quantization is a broad notion used in several fields especially in the sciences, including signal processing, quantum physics, computer science, geometry, music and others. The concept of quantization is related to the idea of grouping, dividing or approximating some physical quantity by a set of small discrete measurements.

Data Quantization involves the discretization of data, or the approximation of large data sets by smaller data sets.

This mini dissertation is a research dissertation that considers how data, which is of a statistical nature, can be quantized and compressed.

Chapter One:
Introduction to Data and Data Compression

1.1 Some Introductory Concepts

In order to understand the concepts of quantization and compression a few introductory notions need to be discussed.

1.1.1 Analog and Digital Data

Digital data uses discrete values to represent information. Analog data, on the other hand, uses a continuous range values to represent information. Digital data may represent discrete information, such as numbers, letters or symbols, however it can also be used to represent approximations of continuous information, such as sounds and images [1].

Only digital data can be compressed, while continuous data requires discretization, this can be achieved with the help of an Analog-to-Digital Converter.

Analog-to-Digital Converter and Digital-to-Analog Converter

"An analog-to-digital converter (ADC) is a device that converts a continuous quantity to a discrete time digital representation of the continuous quantity." A"digital-to-analog converter (DAC) performs the inverse operation to the analog-to-digital converter (ADC)" [2].

In general, ADC uses sampling that is repeated over fixed time periods (or spaces) satisfying certain conditions in order that the signal may be reconstructed [3]. The sampled data is then approximated (quantized) by a set of representative values, this approximation is often in the form of high precision rounding.

In Figure 1.1 a simplified Analog to Digital process is illustrated.

The signal is sampled at fixed time intervals T. Quantization of the signal amplitude is then achieved by rounding the signal amplitude to its nearest (integer) value. The quantized amplitude values are then encoded into binary values. The digital signal is then sent across a channel or stored and then requires decoding which reconstructs an approximation of the original signal.

[1] Mitra, S. K. Digital Signal Processing a Computer-Based Approach. Mcgraw Hill Higher Education. 2002. Page 1.
[2] en.wikipedia.org/wiki/Analog-to-digital_converter confirmed by Mitra, S. K. Digital Signal Processing a Computer-Based Approach. Mcgraw Hill Higher Education. 2002. Page 38.
[3] For a more detailed discussion on statistical signal sampling theory, see Gray, R.M and Davisson, L.D. An Introduction to Statistical Signal Processing. Cambridge University Press. 2010. And in the context of image data, see Rosenfeld, A. and Kak, A. C. Digital Picture Processing. Academic Press. 1976.

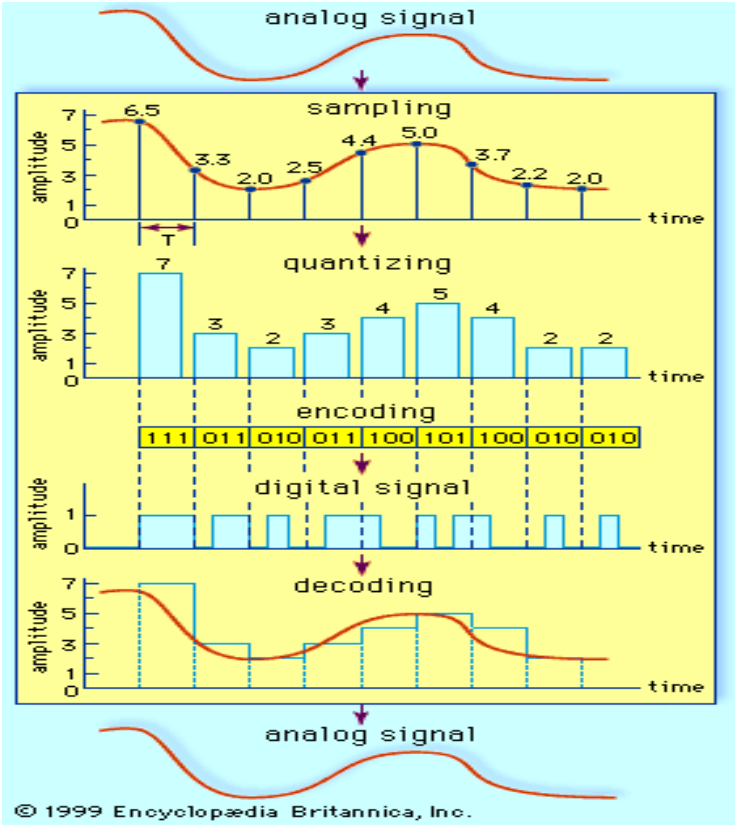


Figure 1.1[4] Analog-to-Digital Converter The steps of analog to digital conversion: sampling, quantizing and encoding &decoding.

1.1.2 The Stochastic Representation of a Digital Signal

Before introducing a digital signal, it is useful to consider its analog counterpart.

The Analog Signal

An analog signal can be considered as a continuous time, continuous space stochastic process.

For example audio or visual data signals, which according to [5] "travel in a waveform that can vary continuously and infinitely along two parameters, amplitude and frequency. Amplitude refers to signal intensity or signal strength, which manifests as volume in audio signals and brightness in visual signals. Frequency refers to the number of waveforms per second, or cycles per second … Frequency manifests as pitch, or tone, in audio signals, and as colour in image and video signals."

[4] Image from www.Encyclopedia_Britannica.com/Basic-steps-in-analog-to-digital-conversion-An-analog-signal obtained from a Google search.
[5] Webster's New World Telecom Dictionary. Wiley Publishing. 2008.

The Digital Signal

According to statistical signal processing texts, a digital signal is a stochastic process with a countable set of indexing parameters $t \in T$ and a finite set of output values or states $s \in S$, making it a discrete time, discrete space stochastic process (or random field).

A digital signal will often contain significant statistical dependence structures, these include both linear and non-linear dependencies, both of which are very useful in data compression, however isolating and exploiting non-linear dependencies in digital signal data is often far more challenging to achieve than linear sources of dependency.

1.1.3 Signal Sampling and Quantization

According to the paper [6], quantization can be studied as a particular type of sampling, in fact quantization and sampling are related processes, where sampling here refers to signal sampling. It is shown that sampling theory can be used to analyse the statistical properties of quantized variables.

Although statistical sampling theory will not be discussed in this mini dissertation, a brief comparison of the processes is useful.

- Signal sampling discretizes time and space, while quantization discretizes amplitude or energy.
- Signal sampling is a linear process, while quantization is non-linear.
- Signal sampling and quantization are orthogonal processes and are independent, therefore the order of the processes of sampling and quantization are reversible. [7]

Data in the Mini-Dissertation

The examples, simulations and picture study are intended to describe the basic theory of the compression process.

The data used in the mini-dissertation will be sourced from random variables (or vectors) with a discrete, finite indexing parameter t and a discrete finite sample space although this indexing parameter space and sample space may be very large.

The long term behaviour of a signal, stationarity and ergodicity of a particular signal will not be discussed in this mini-dissertation.

---

[6] Widrow, B. Statistical Theory of Quantization. IEEE Transactions on Instrumentation and Measurement, Vol. 45, No 2, April 1996.

[7] Quantization is often more computationally expensive than sampling, especially in the case of vector quantization. Often the data signal is sampled prior to quantization, to lessen the burden of quantizing a larger data set.

1.2 An Introduction to Data Compression

Data compression is the reduction of bits required for the storage and/or transmission of data.

Different types of data contained in text, programs, games, images, audio, video, etc. can be compressed to a reduced size, and then decompressed (reconstructed).

There are many types of data compression, all of which fall into the two categories, lossless and lossy data compression.

1.2.1 Lossless and Lossy Data Compression

Lossless data compression is the compression of digital data, where there is no change (error) between the original data and the data after it has been reconstructed. The goal of lossless compression is the maximization of compression with zero loss of fidelity.

Lossy data compression is the compression of analog or digital data that introduces an error (change) between the original data and the reconstructed data. The goal of lossy compression is the reduction of the data size to a required level so that the data may be reconstructed as accurately as possible.

1.2.2 Lossy vs. Lossless Data Compression

Lossy and lossless data compression are similar in many ways, the differences between these two data compression methods result from the quantization of the data before compression. A few of the differences are listed below.

- Bounds of Compression

  For discrete data, both lossy and lossless compression may be applicable, lossless compression, however has a lower bound on the measure of compression achievable this will be shown in the next chapter. Lossy compression, on the other hand, can compress a data signal to arbitrarily low levels, this, however at the cost of increased error this will be shown in chapters three, four and five.

- Generation Loss

  The repetition of lossy data compression to the same data will result in *generation loss*, which is the progressive loss in quality due to repeatedly compressing and decompressing the data file. In contrast, repeatedly compressing and decompressing the data file using lossless data compression will not result in any loss of quality.

- Approximation Error

  Lossless compression can be used where lossy compression cannot, since no error is introduced, i.e. the reconstructed data is the same as the original data. In situations

where even the slightest error could render the data redundant, or where data errors could cause some program to malfunction, lossless compression will be used as opposed to lossy compression.

- Analog-to-Digital Conversion

  Any analog-to-digital conversion requires a form of lossy compression, since analog data is continuous, while digitizing requires that the continuous data be discretized in a finite fashion, thereby introducing errors in the data. The direct application of lossless compression is not applicable for the ADC process.

- Image and Sound Data Compression

  In image and sound data compression, often a large amount of data can be discarded before degradation in the output is perceivable by the human senses. This redundancy is often referred to as *psycho-visual* or *audio redundancy* in image and sound data compression respectively.

  Lossy compression is very effective at reducing the data size when significant redundancy is present within the data. Lossless compression, on the other hand is less effective at compressing image and sound data, since the complexity of the data cannot easily be restructured in order for any significant data compression to take place.

- Quantization

  When lossless compression is used in conjunction with quantization, then it is considered as lossy compression.

1.2.2 Data Information Redundancy

Compression systems reduce information redundancy in data. Information redundancy is measured in bits which require additional bandwidth for transmission or extra hard drive space for storage in addition to the bandwidth or space required to transmit or store the information without the redundancy.

Much of this redundancy is statistical redundancy i.e. patterns and dependency amongst data elements. This redundancy can be measured in bits by considering the difference between the information content of the data and the space required to store or transmit the data.

The information content of data is a measureable quantity and will be discussed in more detail in the next chapter.

Example 1.1

Rounding Quantization, Encoding and Data Redundancy in Image Data

Consider the following context in image data compression. Redundancy in image data is a combination of coding redundancy (see chapter two), inter-pixel redundancy (see chapter five) and psycho-visual redundancy.

Lossless compression is able to compress data with substantial redundancy, however in image compression the complexity of the data is often far too high to exploit the redundancy and thereby attain a high level of compression. When quantization and lossless data compression are used together, image data can be compressed more effectively.

Quantization can introduce *small* enough errors that do not change the perceived quality of the image. These *small* errors however change the data structure in order that lossless encoding is better able to take advantage of the increased data redundancy and compress the data more effectively than without the use of quantization.

Consider an RGB (Red-Green-Blue) image in colour image compression. The size of the image is $(N \times M)$ pixels and the RGB data matrix is a $(3N \times M)$ matrix representing the bitmap image [8].

$$\begin{pmatrix} \boldsymbol{x}_{1,1} & \cdots & \boldsymbol{x}_{1,M} \\ \boldsymbol{x}_{2,1} & \cdots & \boldsymbol{x}_{2,M} \\ \vdots & \ddots & \vdots \\ \boldsymbol{x}_{N,1} & \cdots & \boldsymbol{x}_{N,M} \end{pmatrix} \text{ with } \boldsymbol{x}_{i,j} = \begin{pmatrix} r_{i,j} \\ g_{i,j} \\ b_{i,j} \end{pmatrix}.$$

If, for example, the red hue of an RGB data vector is to be compressed independently of the blue and green hues, then let $\boldsymbol{R}$ be the red vector of the top left-hand corner of the RGB matrix of an image [9].

$$\boldsymbol{R} = \begin{pmatrix} 67 & 62 & 66 & 65 & 71 & 63 & 63 & 64 & 61 & 64 \\ 65 & 65 & 63 & 66 & 68 & 62 & 61 & 62 & 63 & 66 \\ 64 & 66 & 64 & 67 & 66 & 65 & 63 & 62 & 64 & 67 \end{pmatrix}$$

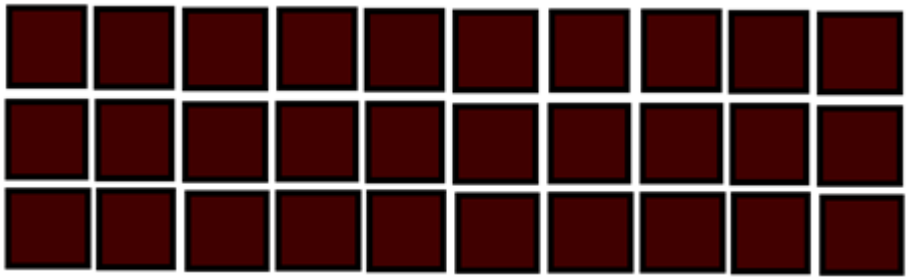


Figure 1.2 3-by-10Pixel Illustration (Unquantized) Visualization of the colours represented by the **R** matrix.

[8] The digital image will be considered in more detail in section 5.3.
[9] The values of the RGB matrix were not simulated but rather picked artificially for the sake of illustration.

Notice that the colours in the 30 blocks representing $\boldsymbol{R}$ appear to be the same to the observer and no noticeable change in colour is apparent to the naked eye.The values of the elements of the matrix $\boldsymbol{R}$ seem to be scattered around 65.

Applying lossless compression to the above data set will probably only yield slight compression, as almost no values are repeated in the data vector.

If however, the values in $\boldsymbol{R}$ were quantized by assigning all numbers between 60 and 66 to the value 63 and all numbers between 67 and 73 to 70, then the quantized matrix $Q(\boldsymbol{R})$ will be as follows

$$Q(\boldsymbol{R}) = \begin{pmatrix} 70 & 63 & 63 & 63 & 70 & 63 & 63 & 63 & 63 & 63 \\ 63 & 63 & 63 & 63 & 70 & 63 & 63 & 63 & 63 & 63 \\ 63 & 63 & 63 & 70 & 63 & 63 & 63 & 63 & 63 & 70 \end{pmatrix}$$

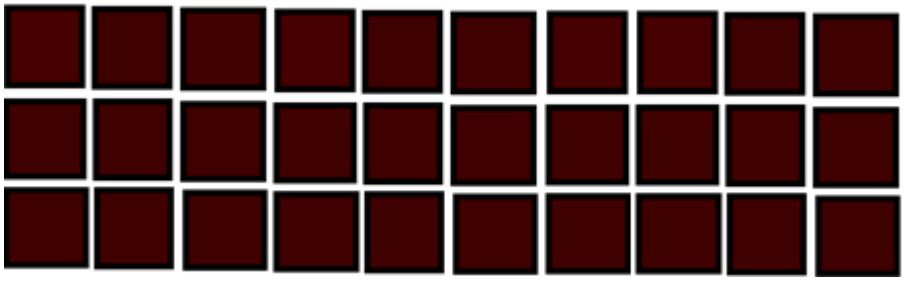


Figure 1.3 3-by-10Pixel Illustration (Quantized) Visualization of the colours represented by the $Q(\mathbf{R})$matrix.

Notice that the variation within the block colours does not seem to have changed, since the alterations (quantization) were inadequate in alerting the observer to visible irregularities, this demonstrates psycho-visual redundancy.

There is however an increase in redundant information, the value 63 is repeated 25 times, reading the values left-to-right yields runs of 63's up to 9 values long and reading in a zigzag pattern up to 14 values long. The value 70 is only repeated 5 times, and may even be able to be left out, thereby reducing the original random matrix to a set of 30 values of 63 [10], which in fact will characterize this matrix completely. We could of course also use 30 values of 70, however, since 63 occurs substantially more, it would be more correct to replace the values by 63, this will also reduce the overall distortion error which will be discussed in chapter three.

The compression of the example was substantial however, it was also isolated, pixilation may occur in particular areas in the reconstructed image where the colour variation is very low. Pixilation and other are visible inconsistencies are known as artefacts and are undesirable.

This was an example of uniform scalar quantization applied to the red hue vector of an RGB matrix. Uniform and non-uniform scalar quantization will be discussed in detail in chapter three. Uniform scalar quantization, however is seldom used alone in the compression of image data as done in Example 1.1.

[10] Explicitly stating a value and its number of sequential occurrences is known as "run length encoding" and will be briefly illustrated in the picture study in Chapter Five.

1.3 Theoretical Aspects of Data Compression

1.3.1 Information Source

The information source is a process, where data is created, transferred or stored.

After quantization, the information source can be viewed as a discrete process which is represented by discrete random variable. The output of an information source with a finite outcome set $S = \{\boldsymbol{x}_i\}_{i=1}^{k}$ will be represented by the ordered pair $\left(\boldsymbol{x}_i, f_{\boldsymbol{X}}(\boldsymbol{x}_i)\right)$ where $f_{\boldsymbol{X}}(\boldsymbol{x})$ is the discrete joint probability mass function of $\boldsymbol{X}$

If the source is viewed as the output of a data quantizer Q then the ordered pair will be made up of a discrete value $Q(\boldsymbol{x}) = \boldsymbol{y}$ and its related probability $f_{\boldsymbol{Y}}(\boldsymbol{y})$ where $\boldsymbol{Y}: p \times 1$, that is if $p = 1$, then y is a scalar and if $p > 1$ then $\boldsymbol{y}$ is a vector.

1.3.2 Encoding

Data is encoded when a value $\boldsymbol{y}: p \times 1$ is assigned to a binary representative $\mathfrak{E}(\boldsymbol{y})$. Scalar encoding assigns a scalar value y to a binary representation $\mathfrak{E}(y)$, while vector encoding assigns a vector $\boldsymbol{y}$ to a binary representation $\mathfrak{E}(\boldsymbol{y})$. The process of encoding is one-to-one [11] so that the scalar y or the vector $\boldsymbol{y}$ can be retrieved uniquely when the representative is decoded $\mathfrak{E}^{-1}\big(\mathfrak{E}(\boldsymbol{y})\big) = \boldsymbol{y}$.

1.3.3 Binary Code Alphabet

The set of symbols $\big\{\{0\},\{1\}\big\}$ is known as a binary code alphabet. It has a radix 2, where the radix is the number of elements in the set [12].

1.3.4 Binary Codeword

A binary codeword is a sequence from the binary code alphabet which represents the value $Q(\boldsymbol{x}) = \boldsymbol{y}$.

The binary code alphabet will be indicated by BA where $BA = \big\{\{0\},\{1\}\big\}$. $BA_{[p]}^{*}$ is the set of all permutations up to and including the permutations of length p of the elements of BA. So for $p = 1, 2, 3$ we have

$$BA_{[1]}^{*} = \big\{\{0\},\{1\}\big\}.$$
$$BA_{[2]}^{*} = \big\{\{0\},\{1\},\{00\},\{01\},\{10\},\{11\}\big\}.$$

For ease of notation we will adopt the relaxed notation

[11] This criterion will be assumed throughout this mini-dissertation. The one-to-one requirement for encoding results in a non-singular code, and will be discussed later this chapter

[12] All the theory in the section will be considered for binary data, due to the nature of the problem of electronic data storage and compression, however the theory of binary data codes is easily extended to "r-ary" data codes.

$$BA^*_{[3]} = \{0, 1, 00, 01, 10, 11, 000, 001, 010, 100, 011, 101, 110, 111\}.$$

The binary codewords of up to length p are sourced from the set C which is a non-empty subset of $BA^*_{[p]}$.The set of binary codewords C is known as a codebook. The number of codewords in the codebook is referred to as the size of the codebook.

1.3.5 Binary Encoder

A binary encoder $\mathfrak{E}$: $Q(S) \rightarrow C$ is one-to-one mapping from the quantized space $Q(S)$ onto the codebook C [13].

The binary encoding of the value $Q(\boldsymbol{x}) = \boldsymbol{y}$ is the codeword $\mathfrak{E}(\boldsymbol{y}) = \boldsymbol{e}$, where $\boldsymbol{e}$ is a $(n \times 1)$ binary codeword. The codeword $\boldsymbol{e_i}^T = (e_{i,1}\ e_{i,2}\ \ldots\ e_{i,n})$ where $e_{i,j}$ is either 0 or 1.

The inverse of $\mathfrak{E}$ is known as the decoder $\mathfrak{E}^{-1}$. $\mathfrak{E}^{-1}$maps a binary codeword onto its allocated output value $\mathfrak{E}\left(\mathfrak{E}^{-1}(\boldsymbol{y})\right) = \boldsymbol{y}$.

1.3.6 Binary String

A binary string is a sequence of concatenated binary codewords each representing a specific value $Q(\boldsymbol{x}) = \boldsymbol{y}$.

The string is not delineated in any way, in other words, no commas, hyphens, spaces, etc. are used between the binary codewords. For the purpose of data compression, there are more efficient means to identify a binary codeword within a binary string than to use delineation [14].

For example, the vector $\boldsymbol{y}^T = (y_1\ y_2\ y_3)$ may be represented by the binary string or vector by means of scalar encoding $\mathfrak{E}(\boldsymbol{y}^T) = \left(\mathfrak{E}(y_1)\ \mathfrak{E}\ (y_2)\ \mathfrak{E}(y_3)\right)^T = \boldsymbol{e}^T = (\boldsymbol{e}_1^T\ \boldsymbol{e}_2^T\ \boldsymbol{e}_3^T)$ or the vector encoding $\mathfrak{E}(\boldsymbol{y}^T) = \boldsymbol{e}^T$ [15].

1.3.7 Binary String Length

The binary string length is the number of binary code alphabet elements contained within the string. The length l of the binary string $\boldsymbol{a}$ is denoted as $l(\boldsymbol{a})$ and is measured in bits.

For the binary string $\boldsymbol{e}^T = (\boldsymbol{e}_1^T \boldsymbol{e}_2^T \boldsymbol{e}_3^T)$, the length $l(\boldsymbol{e}^T) = l\left((\boldsymbol{e}_1^T \boldsymbol{e}_2^T \boldsymbol{e}_3^T)\right)$ which can be calculated as $l(\boldsymbol{e}_1^T) + l(\boldsymbol{e}_2^T) + l(\boldsymbol{e}_3^T)$. This will be illustrated in the next example.

---

[13] Roman, S. Coding and Information Theory. Springer-Verlag. 1997. Page 39.
[14] This will be discussed later in this chapter and in chapter two.
[15] The difference between scalar and vector or block encoding will be illustrated in example 1.2 and example 1.3.

Example 1.2 Scalar Encoding

An example of an arbitrary binary string $\boldsymbol{e}^T$ may be

$$\boldsymbol{e}^T = (0100100111010101010).$$

This string may be made up of, say, three arbitrary binary codewords each representing some output sequence $(y_1\ y_2\ y_3)$ sampled from the discrete random variables Y for example

$$\mathrm{Y} \rightarrow^{\text{sampling}} (\mathrm{y}_1\ \mathrm{y}_2\ \mathrm{y}_3) \rightarrow^{\text{encoding}} (0100)^{\mathrm{T}}(10011101010)^{\mathrm{T}}(1010)^{\mathrm{T}}.$$

The length of the binary string $\boldsymbol{e}$, $l(\boldsymbol{e}^T) = l\big(\mathfrak{E}(y_1)\big) + l\big(\mathfrak{E}(y_2)\big) + l\big(\mathfrak{E}(y_3)\big)$.

$$l(\boldsymbol{e}^T) = 4 + 11 + 4 = 19.$$

Example 1.3 Vector Encoding

In contrast with example 1.2, the arbitrary binary string may be

$$\boldsymbol{e}^T = (0100100111010101010).$$

This string may be the binary codeword of the vector or block $\boldsymbol{Y}$ of dimension 3 of a random vector $\boldsymbol{Y}^T = (\mathrm{Y}_1\ \mathrm{Y}_2\ \mathrm{Y}_3)$ with output vector $\boldsymbol{y}^T = (y_1\ \ y_2\ \ y_3)$ so for example

$$\boldsymbol{Y}^T = (\mathrm{Y}_1\ \mathrm{Y}_2\ \mathrm{Y}_3)\ \rightarrow^{sampling}\ \boldsymbol{y}^T = (y_1\ \ y_2\ \ y_3) \rightarrow^{encoding}\ (0100100111010101010)^T$$

The length of the binary string $\boldsymbol{e}$, $l(\boldsymbol{e}^T) = l\left(\mathfrak{E}\big((y_1\ y_2\ y_3)\big)\right) = 19.$

Scalar encoding individually encodes random variables, while vector or block encoding encodes random vectors.

1.3.8 Code rate

The code rate r is the average number of bits required to encode the random vector $\boldsymbol{Y}$.

The code rate r is the measure of the *efficiency* of the encoding scheme. An increased code rate r implies that a greater capacity is needed to store or transmit the same data, which is less efficient.

The string length l and the code rate r are determined by the encoding method. Two encoding methods will be discussed namely fixed rate encoding and variable rate encoding.

1.3.9 Fixed and Variable Code Rates

Fixed rate encoding uses a fixed number of bits for each codeword, while variable rate encoding does not use a fixed number of bits for each codeword [16].

For an output set of $\{y_i\}_{i=1}^{k}$ and a codebook $C = \{\boldsymbol{e}_i\}_{i=1}^{k}$ with $l(\boldsymbol{e}_i) = l_i$.

Fixed rate encoding uses a fixed length binary string for each codeword $l_i = l$ for all $i = 1,2 \dots, k$ the codebook C of size k will only contain codewords of length l where k is defined by $k = 2^l$.

$$r = \mathbb{E}(l) = \sum_{i=1}^{k} l \cdot f_Y(y_i) = l.$$

Therefore the code rate r is equal to the length l where $l = log_2 k$ and $l \in \mathbb{N}$.

Variable rate encoding of an output set of $\{y_i\}_{i=1}^{k}$ with $l_i = l(\boldsymbol{e}_i)$, the lengths may differ for each value y_i. So for the output set $\{y_i\}_{i=1}^{k}$ with probability mass function $f_Y(y_i)$, the code rate is

$$r = \mathbb{E}(l) = \sum_{i=1}^{k} l_i \cdot f_Y(y_i).$$

An interesting situation arises when $k \neq 2^l$, then fixed rate encoding will be inefficient, since if $k \neq 2^l$ and $2^l < k < 2^{l+1}$ for some $l \in \mathbb{N}$ then fixed rate encoding will require a fixed rate of $l + 1$. This inefficiency may be reduced or eliminated by grouping random variables into blocks which will be illustrated next.

The book [17] gives an example of grouping random variables into a block in order to reduce this inefficiency. If the random variables Y_1 has 3 output variables and Y_2 has 10 output variables, if both were encoded using fixed rate encoding, then $l(\mathfrak{E}(y_1)) = 2$, since $2^1 < 3 < 2^2$ and $l(\mathfrak{E}(y_2)) = 4$, since $2^3 < 10 < 2^4$, implying that individually encoding output values from Y_1 and Y_2 will require 6 bits on average to encode, while if Y_1 and Y_2 were grouped into the vector $\boldsymbol{Y}^T = (Y_1\, Y_2)$ then the output set would be of size $3 \times 10 = 30$ and $2^4 < 30 < 2^5$ which will require only 5 bits on average to encode, notice however that this has not entirely eliminated the inefficiency resulting from unused bits.

[16] Fixed code rate refers to the so called fixed to fixed mapping, which implies that a fixed number of output variables is assigned to a fixed code rate, while variable code rate refers to fixed to variable mapping, which implies that a fixed number of output variables is assigned to a variable code rate. Throughout the mini-dissertation, fixed code rate will refers to a fixed to fixed mapping, while variable code rate will refer to fixed to variable mapping, unless stated otherwise. This is true for both scalar and vector encoding.

[17] Gersho, A. and Gray, R. M. Vector Quantization and Signal Compression. Springer. 1992. Page 226.

1.3.10 Code Rate for Vector Encoding

A vector $\boldsymbol{Y}^T = (\mathrm{Y}_1\ \mathrm{Y}_2 \ldots \mathrm{Y}_\mathrm{m})$ of dimension m is a set of m random variables, that are not necessarily independent or identically distributed, that are grouped together and treated as a m-dimensional random vector, with joint density function

$$f_{\boldsymbol{Y}}(\boldsymbol{y}) = f_{\boldsymbol{Y}}\left(\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}\right)$$

Each possible output vector $\boldsymbol{y}^T = (\mathrm{y}_1\ \mathrm{y}_2\ \ldots\ \mathrm{y}_\mathrm{m})$ with the set $\boldsymbol{y} = \{\boldsymbol{y}_j\}_{j=1}^{k^{[m]}}$ is then encoded as a binary codeword. Where $k^{[m]} = \prod_{i=1}^{m} k_i$ is the number of possible outcome vectors of dimension m each with k_i possible outcomes for each element of the vector [18].
So for the output vector $\boldsymbol{y} = \{\boldsymbol{y}_j\}_{j=1}^{k^{[m]}}$ with probability mass function $f_{\boldsymbol{Y}}(\boldsymbol{y})$, the code rate is

$$r = \mathbb{E}(l) = \frac{1}{m} \cdot \sum_{j=1}^{k^{[m]}} l\left(\mathfrak{E}(\boldsymbol{y}_j)\right) \cdot f_{\boldsymbol{Y}}(\boldsymbol{y}_j).$$

The scaling factor $\frac{1}{m}$ before the summation is to determine the code rate as bits per output value or element-wise code rate, while without it, the summation determines the expected number of bits per block.

1.3.11 Non-Singular, Uniquely Decipherable and Instantaneous Codes

According to the book [19], non-singular codes are codes that are uniquely encoded /decoded so for any $y_i, y_j \in Y$ where $y_i \neq y_j$ implies that $\mathfrak{E}(y_i) \neq \mathfrak{E}(y_j)$.

Unique decipherability implies that all encoded values should be able to be recovered perfectly upon the decoding of the respective binary codewords. This requires that the sequence $\boldsymbol{y}^T = (y_1\ y_2\ y_3 \ldots y_N)$ is in a one-to-one relationship with its encoded counter part $\boldsymbol{e}^T = \left(\mathfrak{E}(y_1)\ \mathfrak{E}(y_2)\ \mathfrak{E}(y_3) \ldots\ \mathfrak{E}(y_N)\right)$ [20].

It is intuitively clear that uniquely decipherable codes are necessarily non-singular, however the opposite is not necessarily true. It is possible for a code to be non-singular but not be uniquely decipherable, this happens in the context of grouping, i.e. instantaneously decoded upon inspection.

---

[18] If the number of output levels is the same for each random variable representing the block, then $k^{[m]} = k^m$.
[19] Roman, S. Coding and Information Theory. Springer-Verlag. 1997. Pages 41-42.
[20] The lossless compression of a binary string must retain the data fidelity, in other words, it must be non-singular and uniquely decipherable.

For example if $\mathfrak{E}(y_1) = (010)$, $\mathfrak{E}(y_2) = (01010)$ and $\mathfrak{E}(y_3) = (01)$, then y_1, y_2 and y_3 are non-singular, however when the binary string (01010) is decoded it maybe decoded as $(y_3\ y_1)$ or as $y_2$this implies that the encoding is not uniquely decipherable.

An instantaneous code is a code that has the requirement that codewords can be decoded without considering the entire binary string.

An instantaneous code differs to a uniquely decipherable code, in that although a finite length, uniquely decipherable binary string has a one-to-one relationship with its decoded counterpart. This relationship may only be recognisable if the string is examined as a whole, while if the code is instantaneous, the binary codewords can be identified instantaneously, without considering the complete binary string. An instantaneous code is uniquely decipherable, but the converse is not necessarily true.

Fixed rate encoding is by definition instantaneous as will be shown in example 1.4, while variable rate encoding requires an additional constraint [[21]].

Example 1.4 Fixed Rate Encoding

Let $Y_i \sim Unif\left(\frac{1}{8}\right)$ with

$$f_{Y_i}(y) = \begin{cases} \frac{1}{8} & if\ y \in \{y_j\}_{j=1}^{8} \\ 0 & otherwise \end{cases}$$

For $i = 1, 2, 3, \dots, N$.

Consider the random vector $\boldsymbol{Y}^T = (Y_1\ Y_2 \dots Y_N)$,

and the vector of observed values

$$\boldsymbol{y}^T = (y_1\ y_2\ \dots y_N).$$

These could be encoded uniquely with a fixed rate encoder with $\boldsymbol{e}_i = \mathfrak{E}(y_i)$ where $\boldsymbol{e}_i$ can be presented as $\boldsymbol{e}_i^T = \left(e_{i,1}\ e_{i,2}\ e_{i,3}\right)$ which has a length of 3, therefore

$$\boldsymbol{e}^T = (\boldsymbol{e}_1^T \boldsymbol{e}_2^T \dots \boldsymbol{e}_N^T) = \left(e_{1,1}\ e_{1,2}\ e_{1,3}\ e_{2,1} \dots\ e_{N,1}\ e_{N,2}\ e_{N,3}\right).$$

The binary string $\boldsymbol{e}^T$has a length of $3 \cdot N$ which is due to the N Observations, each of length 3.

Since the output set contains $k = 8$ values, the length $l\left(\mathfrak{E}(y_i)\right) = l_i = 3$bits.
The expected length or code rate

[21] See section 1.3.12 entitled Variable Rate Encoding, the Prefix Condition and Binary Trees.

$$\mathbb{E}\left(l\left(\mathfrak{E}(\boldsymbol{Y})\right)\right) = r = \sum_{i=1}^{8} 3 \cdot \left(\frac{1}{8}\right).$$

The unique decipherability (in fact instantaneous encoding) of a fixed length code can be illustrated using the following scalar encoding

$$\mathfrak{E}(y_1) = (000)\ \ \mathfrak{E}(y_2) = (100)\ \ \mathfrak{E}(y_3) = (010)\ \ \mathfrak{E}(y_4) = (001)$$
$$\mathfrak{E}(y_5) = (110)\ \ \mathfrak{E}(y_6) = (101)\ \ \mathfrak{E}(y_7) = (011)\ \ \ \mathfrak{E}(y_8) = (111).$$

Consider the string

$$\boldsymbol{e}^T = (10100001001000 10 10011).$$

It follows that $N = 7$ and $\boldsymbol{e}_1^T = (101)$, $\boldsymbol{e}_2^T = (000)$, $\boldsymbol{e}_3^T = (010)$, … , $\boldsymbol{e}_7^T = (011)$ with
$$e_{1,1} = 1, e_{1,2} = 0, e_{1,3} = 1, \dots, e_{7,2} = 1, e_{7,3} = 1$$

Therefore the decoded string is $\mathfrak{E}^{-1}(\boldsymbol{e}^T) = (y_6\ y_1\ y_3\ y_3\ y_4\ y_3\ y_7)$ since
$\mathfrak{E}^{-1}(\boldsymbol{e}_1^T) = y_6$,$\mathfrak{E}^{-1}(\boldsymbol{e}_2^T) = y_1$,…, $\mathfrak{E}^{-1}(\boldsymbol{e}_7^T) = y_7$.

In this example, fixed rate encoding was used, it can be seen that the codeword $\boldsymbol{e}_i$ alone is required in order for the unique decoding $\mathfrak{E}^{-1}(\boldsymbol{e}_i) = y_i$, independent of the rest of the string. This illustrates the difference between instantaneous encoding and uniquely decipherable encoding.

1.3.12 Variable Rate Encoding, the Prefix Condition and Binary Trees

Instantaneous encoding is very advantageous since it allows for immediate decoding of a codeword within a string. If fixed rate codes are instantaneous, as shown in the example above, then why use variable rate encoding?

Variable Rate Encoding

If there is heterogeneity in the probabilities of the probability mass function $f_{\boldsymbol{Y}}(\boldsymbol{y})$, then variable rate encoding will encode data strings at a lower code rate than fixed rate encoding on average [22]. Variable rate encoding is superior in terms of compression to fixed rate encoding [23] and can be made instantaneous, this is achieved with the help of the prefix condition.

[22] Under certain conditions variable rate encoding can always result in lower average rates than fixed rate encoding.
[23] As will be shown in the next chapter, variable rate encoding is able to achieve the lower bound code rate under certain circumstances.

The Prefix Condition for Binary Codewords

The prefix of a codeword will be illustrated using the encoding in section 1.3.11.

The codewords $\mathfrak{E}(y_1) = (010)$, $\mathfrak{E}(y_2) = (01010)$ and $\mathfrak{E}(y_3) = (01)$ were used to encode the output set $\{y_j\}_{j=1}^{3}$. The codeword (01) is a prefix of both the codewords $\mathfrak{E}(y_1) = (010)$ and $\mathfrak{E}(y_2) = (01010)$ and the codeword (010) is a prefix of the codeword $\mathfrak{E}(y_2) = (01010)$. This does not satisfy the prefix condition, which is described below.

Let Y be a random variable with a discrete probability mass function $f_Y(y)$ and a finite output set $\{y_j\}_{j=1}^{k}$ of size k. Then for the case where the block size $m = 1$, each y_j can be represented by a binary codeword $\boldsymbol{e}_j^T = \left(e_{\mathrm{j},1} e_{\mathrm{j},2} e_{\mathrm{j},3} \ldots e_{\mathrm{j},\mathrm{l_j}}\right)$ of length l_j which may vary.

Then for a vector of observed values $\boldsymbol{y}^T = (y_1\ y_2\ \ldots y_N)$, the representative binary codewords are concatenated into a single binary string $\boldsymbol{e}^T = (\boldsymbol{e}_1^T \boldsymbol{e}_2^T \ldots \boldsymbol{e}_l^T)$ of length $l = \sum_{i=1}^{N} l_i$. Notice that this string contains no delineation between codewords.

The prefix condition requires that no codeword of y_i can be a prefix of any other codeword of y_j for all $i \neq j$.The prefix condition ensures that a binary encoding $\boldsymbol{e}^T = (\boldsymbol{e}_1^T \boldsymbol{e}_2^T \ldots \boldsymbol{e}_l^T)$ of each possible output vector $\boldsymbol{y}^T = (\mathrm{y_1\ y_2\ \ldots\ y_N})$ is instantaneous and therefore uniquely decipherable.

According to the book [24] instantaneous or prefix codes are a subset of the set of uniquely decipherable codes, however, the optimal prefix code, will perform as well as an optimal uniquely decipherable code. In other words, there exists a prefix encoding of a particular output string that performs as well as the optimal uniquely decipherable encoding of the same string. The prefix condition is sufficient for output uniqueness and instantaneous decoding.

[24] Gersho, A. and Gray, R. M. Vector Quantization and Signal Compression. Springer.1992. Pages 269-270.

Application of the Prefix Condition: The Binary Tree Method and Huffman Encoding

One way a codebook can be constructed satisfying the prefix condition is by using a binary tree to determine the codeword for each value y.

In Figure 1.4 a branch of a binary tree is shown. Since there is a unique path from each leaf back to the root of the tree, a code built on the notion of the binary tree can be made uniquely decipherable and by ending at unique terminal nodes the process will satisfy the prefix condition.

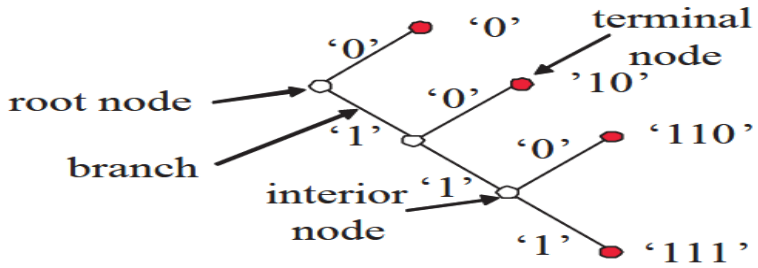


Figure 1.4 Binary Tree An example of a binary tree where the red nodes represent prefix tree leaf nodes [25].

The binary tree method can be programmed as follows by means of the pseudo code given below.

Code 1.1 Pseudo Code: The Binary Tree Encoding Method[26]

To encode an output set of k events $\{y_j\}_{j=1}^{k}$ with the codebook $C = \{\boldsymbol{e}_j\}_{j=1}^{k}$

Let $n_i = 0$ or 1 /*n_i is binary valued*/
Starting with $\mathfrak{E}(y_1) = n_0$ /* n_0 can be chosen arbitrarily as either 0 or 1*/
Replace $\mathfrak{E}(y_1) = n_0$ with $\mathfrak{E}(y_1) = \mathbf{e}_1$ /*$\mathbf{e}_1 = n_0$*/
For $i = 1, 2, \dots, k-2$
  If $\mathfrak{E}(y_i) = n_0 n_1 \dots n_{i-1}$
  Then $\mathfrak{E}(y_{i+1}) = n_0 n_1 \dots n_{i-2}\overline{n}_{i-1}n_i$ /*Where $\overline{n}_j = complement(n_j)$*/
  Replace $\mathfrak{E}(y_{i+1}) = n_0 n_1 \dots n_{i-2}\overline{n}_{i-1}n_i$ with $\mathfrak{E}(y_{i+1}) = \mathbf{e}_\mathrm{i}$
  /*$\mathbf{e}_\mathrm{i} = n_0 n_1 \dots n_{i-2}\overline{n}_{i-1}n_i$*/
  Increment i
For $i = k-1$
  If $\mathfrak{E}(y_{k-1}) = n_0 n_1 \dots n_{k-2}$
  Then $\mathfrak{E}(y_k) = n_0 n_1 \dots n_{N-1}\overline{n}_{k-1}$ /*$\mathbf{e}_\mathrm{N} = n_0 n_1 \dots n_{k-1}\overline{n}_{k-1}$*/
End

[25] Image used from Wiegand, T. and Schwarz, H. Source Coding: Part I of Fundamentals of Source and Video Coding. Page 25.
[26] This process is one of many ways to encode a string of data uniquely and instantaneously.

Example 1.5 Application of the Binary Tree Method

Let Y be a random variable with probability mass function $f_Y(y)$ with a finite output set of k events $\{y_j\}_{j=1}^{k}$.

Consider an output set of $k = 8$ events $\{y_j\}_{j=1}^{8}$ with related ordered probabilities $f_Y(y_{(j-1)}) \leq f_Y(y_{(j)})$ that are not all equal.

It would be more efficient encoding to assign shorter length codewords to more probable outcomes and longer codewords to less probable outcomes, decreasing the expected length $E(l)$.

Create a codebook using the binary tree encoding method described in section 1.3.11 by letting

$$\mathfrak{E}(y_{(1)}) = (0), \mathfrak{E}(y_{(2)}) = (10), \mathfrak{E}(y_{(3)}) = (110), \mathfrak{E}(y_{(4)}) = (1110), \mathfrak{E}(y_{(5)}) = (11110),$$
$$\mathfrak{E}(y_{(6)}) = (111110), \mathfrak{E}(y_{(7)}) = (1111110), \mathfrak{E}(y_{(8)}) = (1111111)$$

If the vector $\boldsymbol{y}^T = (y_{(6)}\ y_{(1)}\ y_{(3)}\ y_{(3)}\ y_{(4)}\ y_{(3)}\ y_{(7)})$ of length $N = 7$ is observed, then it can be uniquely encoded using the variable rate encoder as

$$\boldsymbol{e}^T = (11111001101101111011101111110).$$

Notice that the above encoding can be decoded instantaneously by reading the string from left to right.

1.3.13 Huffman Encoding

Huffman encoding is a simple and optimal prefix coding technique, that is, it has the shortest expected length for a given discrete probability mass function [27] [28].

Huffman encoding is based on the idea of the "r"-ary tree and simply assigns the "r" smallest initial probabilities, known as terminal nodes, to a parent node, by adding the associated probabilities together. The newly created parent node is treated as a terminal node and the process is repeated. The process terminates when only the root node, and its representative probability namely 1 remain.

In the case of binary encoding, the encoding begins by identifying the two terminal nodes with the smallest probabilities. The terminal node with the smallest probability is assigned a 0 and the terminal node with the second smallest probability is assigned a 1. These terminal node probabilities are added together and this probability is assigned to a parent node which connects the two processed terminal nodes. The parent node is now treated as

[27] Cover, T. M. and Thomas, J. A. Elements of Information Theory. Wiley Series in Telecommunications and Signal Processing. Second Edition. Page 123.
[28] See Roman, S. Coding and Information Theory. Springer-Verlag. 1997. Pages 59 – 61. For proofs of the existence of an optimal coding scheme based on the Huffman coding technique, and the proof of the optimality of the Huffman coding scheme.

a terminal node and the process is then repeated until all the nodes are connected to a single root node.

The binary numbers are then read off from the root node down each branch until the initial terminal nodes are reached. The binary numbers are concatenated into a codeword with the same ordering that they are read off. The codeword is then assigned to the outcome of the random variable associated with the initial probabilities.

The Huffman encoding algorithm assigns the longest codewords to the two output variables with the smallest probabilities and proceeds by assigning codewords with lengths that are inversely proportional to the probabilities of the respective output variables.

1.3.13.1 Huffman's Coding Algorithm (for binary encoding)

The PDF [29] uses the following encoding algorithm.

1. Select the two terminal nodes with the lowest associated probability, and then create a parent node for both nodes, that is a representative for the two outcomes in the binary code tree.
2. Replace the two outcomes and the associated probabilities with the parent node's representation and assign to it the probability equal to the sum of the combined terminal nodes' probabilities.
3. Repeat until the root node is reached and the sum of the probabilities is 1.

Example 1.6 Table 1.1 An Example of Huffman Encoding

Output Vector	**Probabilities due to Huffman Process**								
	Step1	Step2	Step3	Step4	Step5	Step6	Step7	Step8	
y_1	0.21	0.21	0.21	0.21	0.21	0.35	0.41	0.59	1
y_2	0.2	0.2	0.2	0.2	0.2	0.24	0.35	0.41	
y_3	0.19	0.19	0.19	0.19	0.24	0.21	0.24		
y_4	0.12	0.12	0.12	0.16	0.19	0.2			
y_5	0.08	0.08	0.08	0.12	0.16				
y_6	0.08	0.08	0.08	0.12					
y_7	0.07	0.07	0.12						
y_8	0.04	0.05							
y_9	0.01								
Huffman Encoding									
	Step1	Step2	Step3	Step4	Step5	Step6	Step7	Step8	Encoding
$\mathfrak{E}(y_1)$						1		0	01
$\mathfrak{E}(y_2)$						0		0	00
$\mathfrak{E}(y_3)$					1		1	1	111
$\mathfrak{E}(y_4)$				1			0	1	101
$\mathfrak{E}(y_5)$			1		0		1	1	1101
$\mathfrak{E}(y_6)$			0		0		1	1	1100
$\mathfrak{E}(y_7)$		1		0			0	1	1001
$\mathfrak{E}(y_8)$	1	0		0			0	1	10001
$\mathfrak{E}(y_9)$	0	0		0			0	1	10000

[29] Wiegand, T. and Schwarz, H. Source Coding: Part I of Fundamentals of Source and Video Coding. Page 32.

Table 1.2 Expected Length based on Huffman Encoding

Probability $f_Y(y)$	**Codeword** e_i^T	**Length** l_i	**Probability** $\times$ **Length**
0.21	01	2	0.42
0.2	00	2	0.4
0.19	111	3	0.57
0.12	101	3	0.36
0.08	1101	4	0.32
0.08	1100	4	0.32
0.07	1001	4	0.28
0.04	10001	5	0.2
0.01	10000	5	0.05
			$\mathbb{E}\big(l(\boldsymbol{Y})\big)$ =2.92

1.3.13.2 Huffman Encoding

Several properties of Huffman encoding will be discussed in the next section, these can be found in the book [30], [31].

Sibling Property

1. A Huffman code has the sibling property, this property is unique to Huffman coding, when a binary alphabet is used.
2. Every node in the binary code tree, except for the root node, has a sibling.
3. The nodes can be listed in the order of decreasing probability, where each node will be next to its sibling.

Optimality Property

1. Huffman coding is a greedy algorithm that converges to a local optimum, however, shows that this local optimum is in fact a global optimum due to the sibling property. This optimum is known as the entropy of the random variable and will be discussed in more detail later in this chapter and in the next chapter.

2. Huffman codes are optimal in the sense that they give a minimum expected length for a particular probability mass function. They are not always optimal for a specific string of data sourced from the probability mass function.

3. The optimality of a Huffman encoding is also not necessarily unique, there are various other encoding algorithms that may achieve the same minimum expected length for a particular probability mass function.

[30] Gersho , A. and Gray, R. M. Vector Quantization and Signal Compression. Springer. 1992. Pages 274-275
[31] Cover, T. M. and Thomas, J. A. Elements of Information Theory. Wiley Series in Telecommunications and Signal Processing. Second Edition. Pages 123-127.

Problems with the Huffman Encoding Algorithm in Practice

1. Data processes in video, image and sound compression are seldom stationary, the probability distributions are generally not known and adaptive Huffman encoding for non-stationary sources is often far too complex for real time situations. Better alternatives to Huffman encoding are available for this type of situation.

2. When Huffman encoding is applied to blocks of random variables, as these blocks increase with size, the decoder grows exponentially larger, which eventually becomes impractical and hence Huffman encoding is often only applied to small sized blocks of random variables.

3. The Huffman encoding requires a codebook to decode the encoded data, therefore the codebook must also be saved with the encoded data. If the outcome set is very large, then the codebook may also be very large and may erase the gains of the compression.

Different Huffman Encoding Algorithms

According to [32] there are several variations to the basic Huffman encoding algorithm, these are

- Adaptive coding for non-stationary sources
- Conditional Huffman coding for Markov processes
- Variable length vector Huffman coding, where blocks of varying sizes can be used as opposed to fixed size blocks. [33]

1.4 Study1: Huffman Encoding applied to Bernoulli Data

The rest of the chapter will explore the results of Huffman encoding applied to binary strings of differing probability structure. Several questions will be answered by means of examples and the principles illustrated will be proven in the next chapter. The questions are

How does the heterogeneity of the probability function affect the efficiency of Huffman encoding as measured by the code rate in comparison to the entropy lower bound?

How does blocking of random variables change the probability function, and how does that impact the efficiency of Huffman encoding?

Does the dependence structure between random variables impact the encoding efficiency of Huffman encoding and the entropy lower bound?

These questions are explored for binary data.

[32] Wiegand, T. and Schwarz, H. Source Coding: Part I of Fundamentals of Source and Video Coding. Pages 32-42.
[33] None of these variations will be used in this mini-dissertation.

1.4.1 Entropy

Entropy will be discussed in detail in next chapter, however for the purpose of these examples, entropy should be understood as the infimum of the code rate for a given probability function, independent of the encoding scheme used.

The joint entropy H can be calculated as a function of the discrete joint probability mass function $f_{\boldsymbol{Y}}(\boldsymbol{y})$ of $\boldsymbol{Y} = \begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_m \end{pmatrix}$ with finite output set that takes on the elements $\{y_{i1}, y_{i2}, y_{i3}, \dots, y_{ik_i}\}$ for $i = 1, 2, 3, \dots, m$.

$$H(\boldsymbol{Y}; \boldsymbol{\theta}, m) = -\sum_{y_m} \dots \sum_{y_3} \sum_{y_2} \sum_{y_1} \left(f_{\boldsymbol{Y}} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} \right) \cdot \log_2 \left(f_{\boldsymbol{Y}} \left(\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} \right) \right)$$

where the summation $\sum_{y_i}\ i = 1, 2, \dots, m$ is over the output set of the i^{th} element of $\boldsymbol{Y}$ and $\boldsymbol{\theta}$ the parameters for $f_{\boldsymbol{Y}}(\boldsymbol{y})$.

For $m = 1$ the entropy of the random variable Y is given by,

$$H(Y; \boldsymbol{\theta}, 1) = -\sum_{y} f_Y(y) \cdot \log_2\big(f_Y(y)\big).$$

1.4.2 Blocking random variables when used in conjunction with Huffman encoding

Let $\boldsymbol{Y}^T = (Y_1 Y_2 Y_3 \dots Y_m)$ with $Y_i \sim bernoulli(p)$ independently distributed

$$P(Y_i = y) = \begin{cases} p & if\ y = 1 \\ 1 - p & if\ y = 0 \end{cases}$$

for $i = 1, 2, 3, \dots, m$.

The entropy $H(\boldsymbol{Y}; \boldsymbol{\theta}, m)$ can be calculated by

$$H(\boldsymbol{Y}; \boldsymbol{\theta}, m) = -\sum_{y_m=0}^{1} \dots \sum_{y_3=0}^{1} \sum_{y_2=0}^{1} \sum_{y_1=0}^{1} f_{\boldsymbol{Y}} \left(\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} \right) \cdot \log_2 \left(f_{\boldsymbol{Y}} \left(\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} \right) \right).$$

A vector of length m with each element either a 0 or a 1 results in 2^m distinct vectors, but since the random variables are independently identically distributed there will be $\binom{m}{j}$ binary strings containing j "$1's$" and m - j"$0's$" where $j = 0, 1, 2, \dots, m$ and the probability associated with each one of these strings will be $p^j \cdot (1-p)^{m-j}$ therefore $H(\boldsymbol{Y}; \boldsymbol{\theta}, m)$ is reduced to

$$H(\boldsymbol{Y};p,m) = -\sum_{j=0}^{m} \binom{m}{j} \cdot p^j \cdot (1-p)^{m-j} \cdot \log_2\left(p^j \cdot (1-p)^{m-j}\right)$$

The entropy $H(\boldsymbol{Y};p,m)$ per block size m can be seen in Figure 1.5

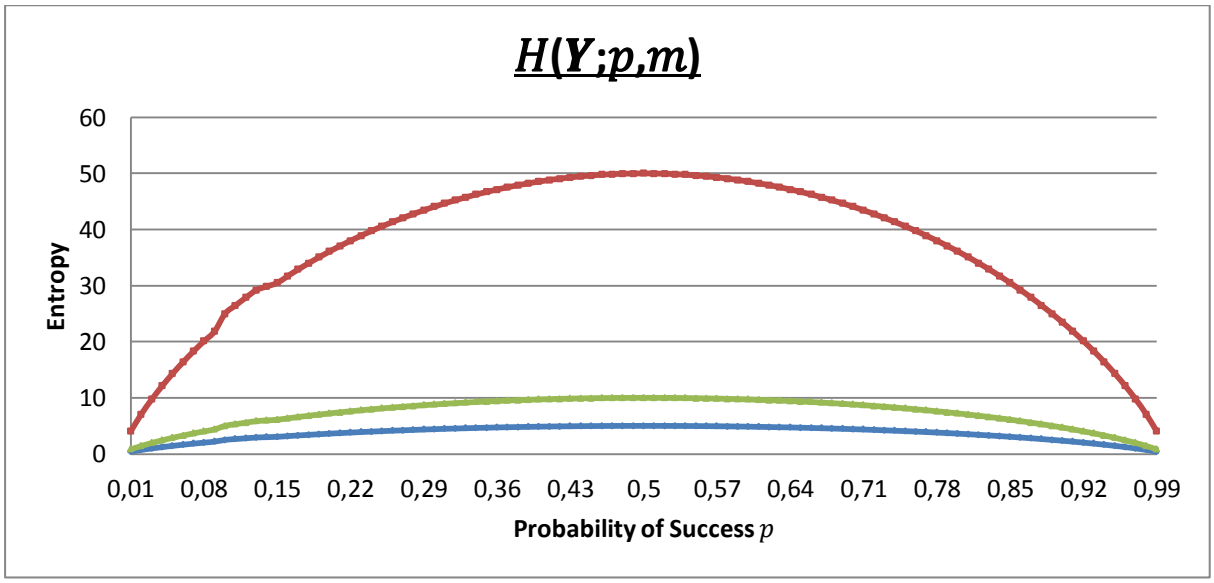


Figure 1.5 The Entropy Function $H(\boldsymbol{Y};p,m)$ Calculated for the Block Sizes $m = 5$, $m = 10$ and $m = 50$ for p - values ranging between $(0.01, 0.99)$ (for $p \in [0,1]$ and m fixed), $H(\boldsymbol{Y};p,m)$ is a continuous function of p.

The entropy per block can be transformed into the entropy per output value in order to obtain an element-wise contribution to the entropy [34], by dividing $H(\boldsymbol{Y};p,m)$ by the block size m, this can be seen in Figure 1.6

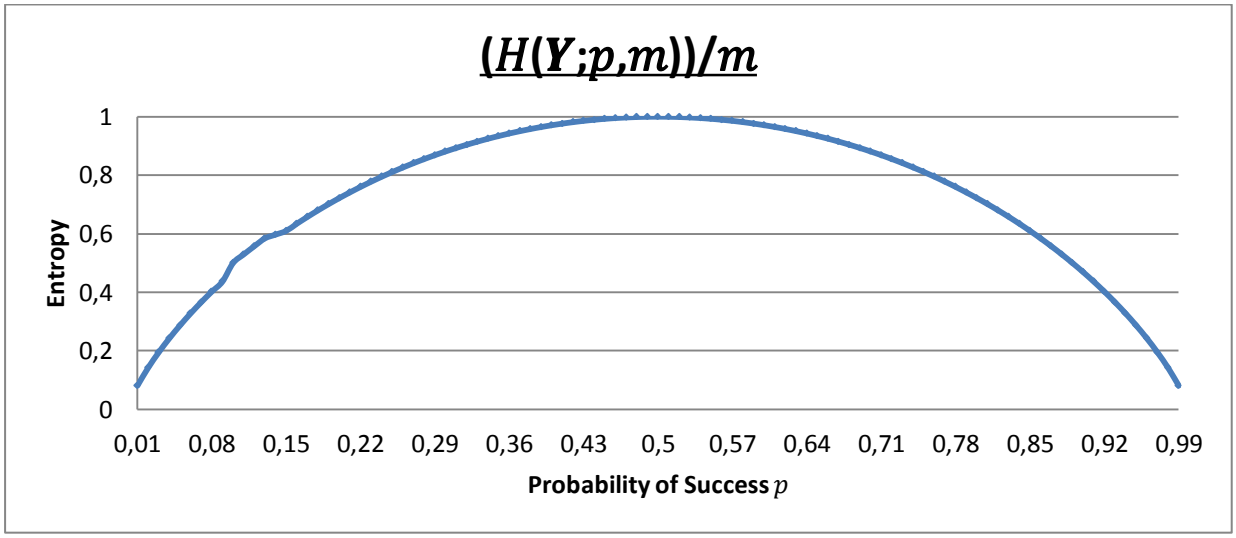


Figure 1.6 The Scaled Entropy Function The result is an entropy rate that is a continuous function of p, but independent of the size m, this is can be related to the independence of the Bernoulli random variables and will be further discussed in the next chapter.

[34] This corresponds to dividing the code rate by the block size m in order to determine the code rate per output value, or element-wise code rate as discussed earlier in the chapter.

As seen in Figure 1.6, the element-wise contribution to the entropy of a Bernoulli random variable will be largest for $p = 0.5$ and decrease towards zero as p tends towards either zero or one, i.e. where the outcome of the experiment is certain. The maximum element-wise entropy where the block elements are identically Bernoulli distributed occurs when the probability function is most homogeneous and the minimum occurs when the probability function is least homogeneous.

Example 1.7

In this example we will examine the resulting effects to the code rate and entropy rate due to increasing block size and changing probability structure of a binary string.

Blocks of size $m = 2, 3, 4$ will be encoded using Huffman encoding for $p = 0.6, 0.7, 0.8$ and 0.9. The code rate r will be calculated using the formula

$$r = \mathbb{E}(l) = \frac{1}{m}\sum_{j=1}^{k^{[m]}} l\left(\mathfrak{E}(\boldsymbol{y}_j)\right) \cdot f_Y(\boldsymbol{y}_j)$$

where $f_Y(\boldsymbol{y}_j) = p^{m-j} \cdot (1-p)^j$, $k = 2$ (Bernoulli) and $k^{[m]} = k^m$.

Table 1.3.1 Block of Size $m = 2$ and $p = 0.6$

Output Vector	**Probabilities due to Huffman Process**			
	Initial Probabilities	Step1	Step2	Step3
(0 0)	0.16	0.4	0.4	1
(0 1)	0.24	0.24	0.60	
(1 0)	0.24	0.36		
(1 1)	0.36			
Huffman Encoding				
	Step1	Step2	Step3	Encoding
(0 0)	0		0	00
(0 1)	1		0	01
(1 0)		0	1	10
(1 1)		1	1	11

The expected length $\mathbb{E}(l)$ is 1, while the entropy rate is 0.971. This is a trivial example showing that with limited heterogeneity, Huffman encoding will not achieve any compression.

In Table 1.3.2 the probability p is incremented by 0.1 to $p = 0.7$ and the resultant encoding is determined.

Table 1.3.2 Block of Size $m = 2$ and $p = 0.7$

Output Vector	**Probabilities due to Huffman Process**			
	Initial Probabilities	Step1	Step2	Step3
(0 0)	0.09	0.3	0.51	1
(0 1)	0.21	0.21	0.49	
(1 0)	0.21	0.49		
(1 1)	0.49			
Huffman Encoding				
	Step1	Step2	Step3	Encoding
(0 0)	0	1	1	110
(0 1)	1	1	1	111
(1 0)		0	1	10
(1 1)			0	0

The expected length $\mathbb{E}(l)$ is 0.905, while the entropy rate is 0.8813.

In Table 1.3.3 the probability p is incremented by 0.1 to $p = 0.8$ and the resultant encoding is determined.

Table 1.3.3 Block of Size $m = 2$ and $p = 0.8$

Output Vector	**Probabilities due to Huffman Process**			
	Initial Probabilities	Step1	Step2	Step3
(0 0)	0.04	0.2	0.36	1
(0 1)	0.16	0.16	0.64	
(1 0)	0.16	0.64		
(1 1)	0.64			
Huffman Encoding				
	Step1	Step2	Step3	Encoding
(0 0)	0	1	0	010
(0 1)	1	1	0	011
(1 0)		0	0	00
(1 1)			1	1

The expected length $\mathbb{E}(l)$ is 0.78, while the entropy rate is 0.7219.

In Table 1.3.4 the probability p is incremented by 0.1 to $p = 0.9$ and the resultant encoding is determined.

Table 1.3.4 Block of Size $m = 2$ and $p = 0.9$

Output Vector	**Probabilities due to Huffman Process**			
	Initial Probabilities	Step1	Step2	Step3
(0 0)	0.01	0.1	0.19	1
(0 1)	0.09	0.09	0.81	
(1 0)	0.09	0.81		
(1 1)	0.81			
Huffman Encoding				
	Step1	Step2	Step3	Encoding
(0 0)	0	1	0	010
(0 1)	1	1	0	011
(1 0)		0	0	00
(1 1)			1	1

The expected length $\mathbb{E}(l)$ is 0.645, while the entropy rate is 0.4690.

The Huffman encoder assigns the same number of bits to each output vector for $p = 0.7, 0.8$ and $p = 0.9$, the decrease in expected length is then only attributable to the increasing heterogeneity in the probability structure of the block.

If m is increased to 3, the Huffman encoder has greater flexibility in encoding the output vectors, this means that gains in efficiency are possible in the case of limited heterogeneity as in Table 1.4.1.

Table 1.4.1 Block of Size $m = 3$ and $p = 0.6$

Output Vector	**Probabilities due to Huffman Process**							
	Initial Probabilities	Step1	Step2	Step3	Step4	Step5	Step6	Step7
(0 0 0)	0.064	0.16	0.16	0.16	0.304	0.304	0.592	1
(1 0 0)	0.096	0.096	0.192	0.192	0.192	0.408	0.408	
(0 1 0)	0.096	0.096	0.144	0.288	0.288	0.288		
(0 0 1)	0.096	0.144	0.144	0.144	0.216			
(1 1 0)	0.144	0.144	0.144	0.216				
(1 0 1)	0.144	0.144	0.216					
(0 1 1)	0.144	0.216						
(1 1 1)	0.216							
Huffman Encoding								
	Step1	Step2	Step3	Step4	Step5	Step6	Step7	Encoding
(0 0 0)	0			1		1	1	1110
(1 0 0)	1			1		1	1	1111
(0 1 0)		0			0		0	000
(0 0 1)		1			0		0	001
(1 1 0)			0			0	1	100
(1 0 1)			1			0	1	101
(0 1 1)				0		1	1	110
(1 1 1)					1		0	01

The expected length $\mathbb{E}(l)$ is 0.9813, while the entropy rate is 0.9710. A decrease in code rate is now possible for the case $p = 0.6$ compared to Table 1.3.1 due to the increased block size.

Table 1.4.2 Block of Size $m = 3$ and $p = 0.7$

Output Vector	**Probabilities due to Huffman Process**							
	Initial Probabilities	Step1	Step2	Step3	Step4	Step5	Step6	Step7
(0 0 0)	0.027	0.09	0.09	0.09	0.216	0.363	0.363	1
(1 0 0)	0.063	0.063	0.126	0.126	0.294	0.294	0.637	
(0 1 0)	0.063	0.063	0.147	0.294	0.147	0.343		
(0 0 1)	0.063	0.147	0.147	0.147	0.343			
(1 1 0)	0.147	0.147	0.147	0.343				
(1 0 1)	0.147	0.147	0.343					
(0 1 1)	0.147	0.343						
(1 1 1)	0.343							
Huffman Encoding								
	Step1	Step2	Step3	Step4	Step5	Step6	Step7	Encoding
(0 0 0)	0			0	1		0	0100
(1 0 0)	1			0	1		0	0101
(0 1 0)		0		1	1		0	0110
(0 0 1)		1		1	1		0	0111
(1 1 0)			0			0	1	100
(1 0 1)			1			0	1	101
(0 1 1)					0		0	00
(1 1 1)						1	1	11

The expected length $\mathbb{E}(l)$ is 0.9086, while the entropy rate is 0.8813. Notice that the code rate for the block of size $m = 3$ and $p = 0.7$ is greater than that of $m = 2$ and $p = 0.7$ for Table 1.3.2, Huffman encoding decreases the code rate when the block size is increased, this however does not necessarily occur in a monotone fashion. The entropy remains the same for $m = 2, 3$ (and 4), this is due to the independence of Bernoulli random variables constituting the block and was illustrated in Figure 1.6.

Table 1.4.3 Block of Size $m = 3$ and $p = 0.8$

Output Vector	**Probabilities due to Huffman Process**							
	Initial Probabilities	Step1	Step2	Step3	Step4	Step5	Step6	Step7
(0 0 0)	0.008	0.04	0.04	0.104	0.232	0.232	0.488	1
(1 0 0)	0.032	0.032	0.064	0.128	0.128	0.256	0.512	
(0 1 0)	0.032	0.032	0.128	0.128	0.128	0.512		
(0 0 1)	0.032	0.128	0.128	0.128	0.512			
(1 1 0)	0.128	0.128	0.128	0.512				
(1 0 1)	0.128	0.128	0.512					
(0 1 1)	0.128	0.512						
(1 1 1)	0.512							
Huffman Encoding								
	Step1	Step2	Step3	Step4	Step5	Step6	Step7	Encoding
(0 0 0)	0		0	0		0	0	00000
(1 0 0)	1		0	0		0	0	00001
(0 1 0)		0	1	0		0	0	00010
(0 0 1)		1	1	0		0	0	00011
(1 1 0)				1		0	0	001
(1 0 1)					0	1	0	010
(0 1 1)					1	1	0	011
(1 1 1)							1	1

The expected length $\mathbb{E}(l)$ is 0.728, while the entropy rate is 0.7219.

Table 1.4.4 Block of Size $m = 3$ and $p = 0.9$

Output Vector	**Probabilities due to Huffman Process**							
	Initial Probabilities	Step1	Step2	Step3	Step4	Step5	Step6	Step7
(0 0 0)	0.001	0.01	0.01	0.028	0.109	0.109	0.271	1
(1 0 0)	0.009	0.009	0.018	0.081	0.081	0.162	0.729	
(0 1 0)	0.009	0.009	0.081	0.081	0.081	0.729		
(0 0 1)	0.009	0.081	0.081	0.081	0.729			
(1 1 0)	0.081	0.081	0.081	0.729				
(1 0 1)	0.081	0.081	0.729					
(0 1 1)	0.081	0.729						
(1 1 1)	0.729							
Huffman Encoding								
	Step1	Step2	Step3	Step4	Step5	Step6	Step7	Encoding
(0 0 0)	0		0	0		0	0	00000
(1 0 0)	1		0	0		0	0	00001
(0 1 0)		0	1	0		0	0	00010
(0 0 1)		1	1	0		0	0	00011
(1 1 0)				1		0	0	001
(1 0 1)					0	1	0	010
(0 1 1)					1	1	0	011
(1 1 1)							1	1

The expected length $\mathbb{E}(l)$ is 0.5327, while the entropy rate is 0.469.

By comparing the encoding for Table 1.4.3 and Table 1.4.4, the Huffman encoder assigns the same number of bits to each output vector for $p = 0.8$ and $p = 0.9$, the decrease in expected length is then only attributable to the increasing heterogeneity in the probability structure of the block.

By extending the block size m to 4, the Huffman encoder has even greater flexibility in encoding the output vectors, this means that greater gains in efficiency are possible.

The optimal codeword length is included in Table 1.5.1 up to Table 1.5.4, the bit length of the Huffman encoding can be compared to the optimal codeword length, in order to determine where losses in efficiency occur. The optimal codeword length is calculated as $-\frac{1}{\log_2\left(f_Y(y_j)\right)}$ [35].

[35] The derivation of this value can be found in section 2.4.

Table 1.5.1 Block of Size $m = 4$ and $p = 0.6$

Initial Probabilities p	Output Vector	Huffman Encoding	Optimal Codeword Length
0.0256	(0 0 0 0)	10000	5.287712
0.0384	(1 0 0 0)	10001	4.70275
0.0384	(0 1 0 0)	11000	4.70275
0.0384	(0 0 1 0)	11001	4.70275
0.0384	(0 0 0 1)	0010	4.70275
0.0576	(0 0 1 1)	0011	4.117787
0.0576	(0 1 1 0)	0100	4.117787
0.0576	(1 1 0 0)	0101	4.117787
0.0576	(1 0 1 0)	0110	4.117787
0.0576	(1 0 0 1)	0111	4.117787
0.0576	(0 1 0 1)	1001	4.117787
0.0864	(1 1 1 0)	1101	3.532825
0.0864	(1 1 0 1)	1110	3.532825
0.0864	(1 0 1 1)	1111	3.532825
0.0864	(0 1 1 1)	000	3.532825
0.1296	(1 1 1 1)	101	2.947862

The expected length $\mathbb{E}(l)$ is 0.9812, while the entropy rate is 0.971. Since the optimal codeword lengths are all fractions, and the codewords are restricted to integer values, the Huffman encoding in Table 1.5.1 is suboptimal.

Table 1.5.2 Block of Size $m = 4$ and $p = 0.7$

Initial Probabilities p	Output Vector	Huffman Encoding	Optimal Codeword Length
0.0081	(0 0 0 0)	1111110	6.947862
0.0189	(1 0 0 0)	1111111	5.72547
0.0189	(0 1 0 0)	110001	5.72547
0.0189	(0 0 1 0)	110000	5.72547
0.0189	(0 0 0 1)	111110	5.72547
0.0441	(0 0 1 1)	11001	4.503078
0.0441	(0 1 1 0)	11010	4.503078
0.0441	(1 1 0 0)	11011	4.503078
0.0441	(1 0 1 0)	11100	4.503078
0.0441	(1 0 0 1)	11101	4.503078
0.0441	(0 1 0 1)	11110	4.503078
0.1029	(1 1 1 0)	011	3.280685
0.1029	(1 1 0 1)	010	3.280685
0.1029	(1 0 1 1)	001	3.280685
0.1029	(0 1 1 1)	000	3.280685
0.2401	(1 1 1 1)	10	2.058293

The expected length $\mathbb{E}(l)$ is 0.8918, while the entropy rate is 0.8813.

Table 1.5.3 Block of Size $m = 4$ and $p = 0.8$

Initial Probabilities p p	Output Vector	Huffman Encoding	Optimal Codeword Length
0.0016	(0 0 0 0)	11000110	9.287712
0.0064	(1 0 0 0)	11000111	7.287712
0.0064	(0 1 0 0)	1100000	7.287712
0.0064	(0 0 1 0)	1100001	7.287712
0.0064	(0 0 0 1)	1100010	7.287712
0.0256	(0 0 1 1)	110010	5.287712
0.0256	(0 1 1 0)	110011	5.287712
0.0256	(1 1 0 0)	110100	5.287712
0.0256	(1 0 1 0)	110101	5.287712
0.0256	(1 0 0 1)	110110	5.287712
0.0256	(0 1 0 1)	110111	5.287712
0.1024	(1 1 1 0)	1110	3.287712
0.1024	(1 1 0 1)	1111	3.287712
0.1024	(1 0 1 1)	100	3.287712
0.1024	(0 1 1 1)	101	3.287712
0.4096	(1 1 1 1)	0	1.287712

The expected length $\mathbb{E}(l)$ is 0.7408, while the entropy rate is 0.7219.

Table 1.5.4 Block of Size $m = 4$ and $p = 0.9$

Initial Probabilities p	Output Vector	Huffman Encoding	Optimal Codeword Length
0.0001	(0 0 0 0)	0110010110	13.28771
0.0009	(1 0 0 0)	0110010111	10.11779
0.0009	(0 1 0 0)	011001000	10.11779
0.0009	(0 0 1 0)	011001001	10.11779
0.0009	(0 0 0 1)	011001010	10.11779
0.0081	(0 0 1 1)	0110011	6.947862
0.0081	(0 1 1 0)	0110100	6.947862
0.0081	(1 1 0 0)	0110101	6.947862
0.0081	(1 0 1 0)	0110110	6.947862
0.0081	(1 0 0 1)	0110111	6.947862
0.0081	(0 1 0 1)	011000	6.947862
0.0729	(1 1 1 0)	0111	3.777937
0.0729	(1 1 0 1)	000	3.777937
0.0729	(1 0 1 1)	001	3.777937
0.0729	(0 1 1 1)	010	3.777937
0.6561	(1 1 1 1)	1	0.608012

The expected length $\mathbb{E}(l)$ is 0.4925, while the entropy rate is 0.469.

For $m > 4$, the Huffman encoding became very difficult to do by hand and too sizable to show in the document, so it was programmed in SAS IML for $m = 5, 6, 7, 8, 9, 10$ the code rate and entropy were calculated for each case. The SAS IML code is included in the Table below [36].

Code 1.2 SAS 9.2 Code for Table 1.6

```
*Generating Joint Probability Distributions*
*Based on Independent Identically Distributed Bernoulli Variables*;
Proc iml;
*n0={1,2,3,4,5,6,7,8,9}*;
*n0={0.6,0.7,0.8,0.9}*;
n=n0;
p=p0;
count=0;
v=j((2**n),1,0);
doi=0 to n;
do j=1 to comb(n,i);
count=count+1;
v[count]=(p)**(n-i)*(1-p)**(i);
end;
end;
```

[36] Please note that the SAS coding used was for the mini-dissertation's purpose only and is not economic with regards to time efficiency.

```
*Huffman Encoding Counting Algorithm*
Proc iml;
row={};
v=row`;
call sort(v,1);
doi=1 to nrow(v);
index=index//i;
end;
*Huffman Coding Process*;
v index=index||v;
one_vec={1,1}//j(nrow(v)-2,1,0);
m=j(nrow(v),1,0);
do until (stop=1);
call sort(v_index,2);
seeker1=v_index[1,1]; seeker2=v_index[2,1];
replace=(one_vec`*v_index[,2]);
Temp=v_index[,1]||one_vec;
v_index=v_index[,1]||(v_index[,2]-(v_index[,2]#Temp[,2])+(Temp[,2]));
v_index[1,2]=replace;
call sort(Temp,1);
c=j(nrow(v),1,0);
do j=1 to ncol(m);
if m[seeker1,j]>0 then c=c||m[,j];
if m[seeker2,j]>0 then c=c||m[,j];
end;
m_1=(c[,+]+temp[,2]>0);
call sort(v_index,2);
m=m||m 1;
ifncol(m)=nrow(v) then stop=1;
else stop=0;
end;
*End of Huffman Coding Process*;

*Determining the Average Bit Rate*;
call sort(v,1);
m_final=m[,+];
bitrate=(v`*m_final)/(log2(nrow(v)));
print bitrate;
quit;
```

Table 1.6 Comparison of Code Rates and Entropy for varying m and p values

Block Size	Probability p	Code Rate	Entropy
5	0.6	0.977664	0.9709506
5	0.7	0.888996	0.8812909
5	0.8	0.73792	0.7219281
5	0.9	0.480194	0.4689956
6	0.6	0.9762987	0.9709506
6	0.7	0.8881745	0.8812909
6	0.8	0.725248	0.7219281
6	0.9	0.4701568	0.4689956
7	0.6	0.9753655	0.9709506
7	0.7	0.8844696	0.8812909
7	0.8	0.7317559	0.7219281
7	0.9	0.4743416	0.4689956
8	0.6	0.9744358	0.9709506
8	0.7	0.8858608	0.8812909
8	0.8	0.7322282	0.7219281
8	0.9	0.4757994	0.4689956
9	0.6	0.9744167	0.9709506
9	0.7	0.8842474	0.8812909
9	0.8	0.7251399	0.7219281
9	0.9	0.4749981	0.4689956
10	0.6	0.9736992	0.9709506
10	0.7	0.8842745	0.8812909
10	0.8	0.7282013	0.7219281
10	0.9	0.4766912	0.4689956

The graph below (Figure 1.7) compares the entropy and the code rate as function of block size and probability p given in Tables 1.3.1 through to Table 1.6.

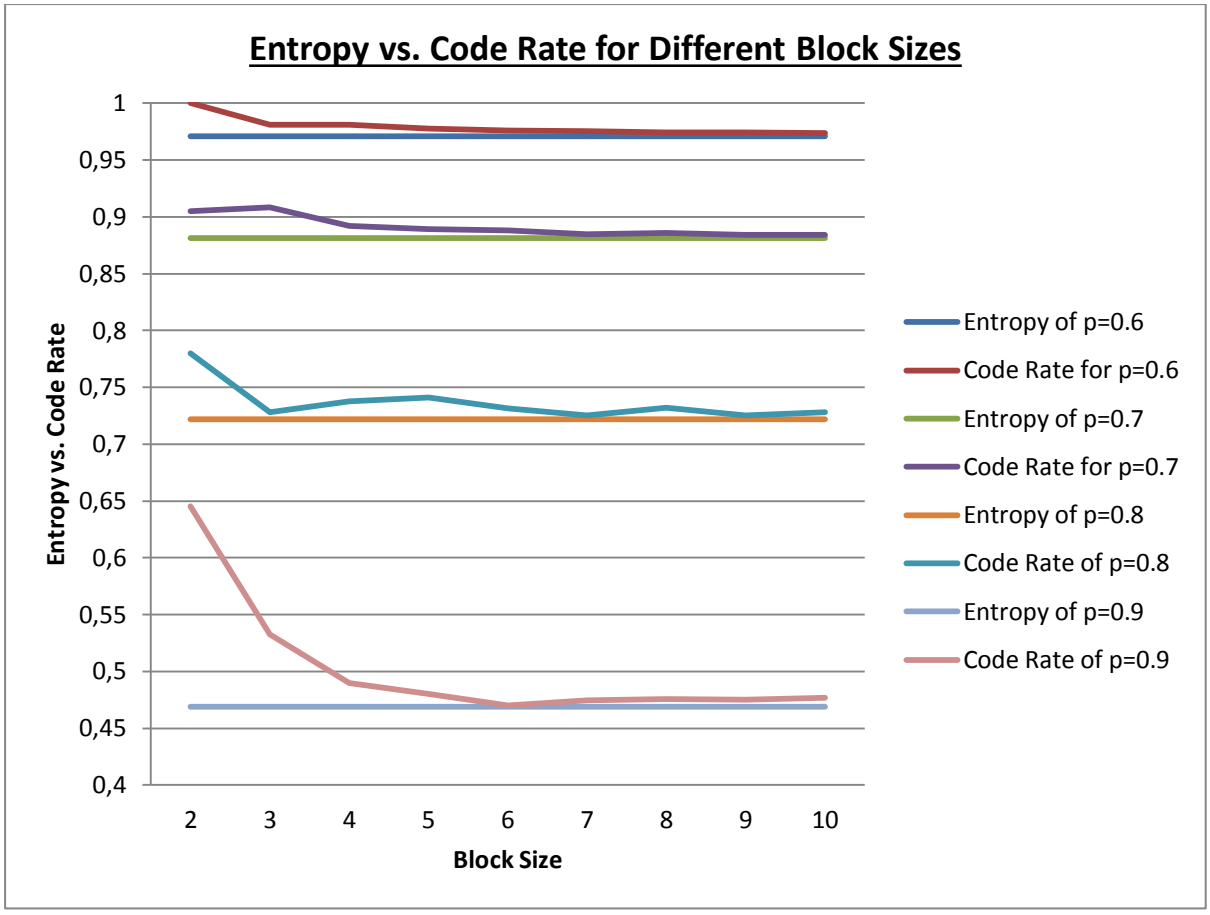


Figure 1.7 Entropy vs. Code Rate for Different Block Sizes An increase in block size m, for a particular success probability $p \neq 0.5$ will result in a decrease in code rate r, this decrease may not be monotone, but as the block size m gets larger, it appears as though the code rate will converge to the entropy of the random variable[37].

The entropy rates are smaller for smaller values of p, however they remain unchanged for increasing block sizes m for a particular p. An increase in block size m results in a decrease in code rate, however this is not a monotone decrease as seen in Figure 1.7. It should also be noted that the decrease in code rate to the entropy rate occurs slower for increasing probabilities p.

The graph below (Figure 1.8) compares the code rates for increasing probabilities p for the block sizes $m = 2, 3, 4$.

[37] This will be proven in the next chapter for all discrete random variables.

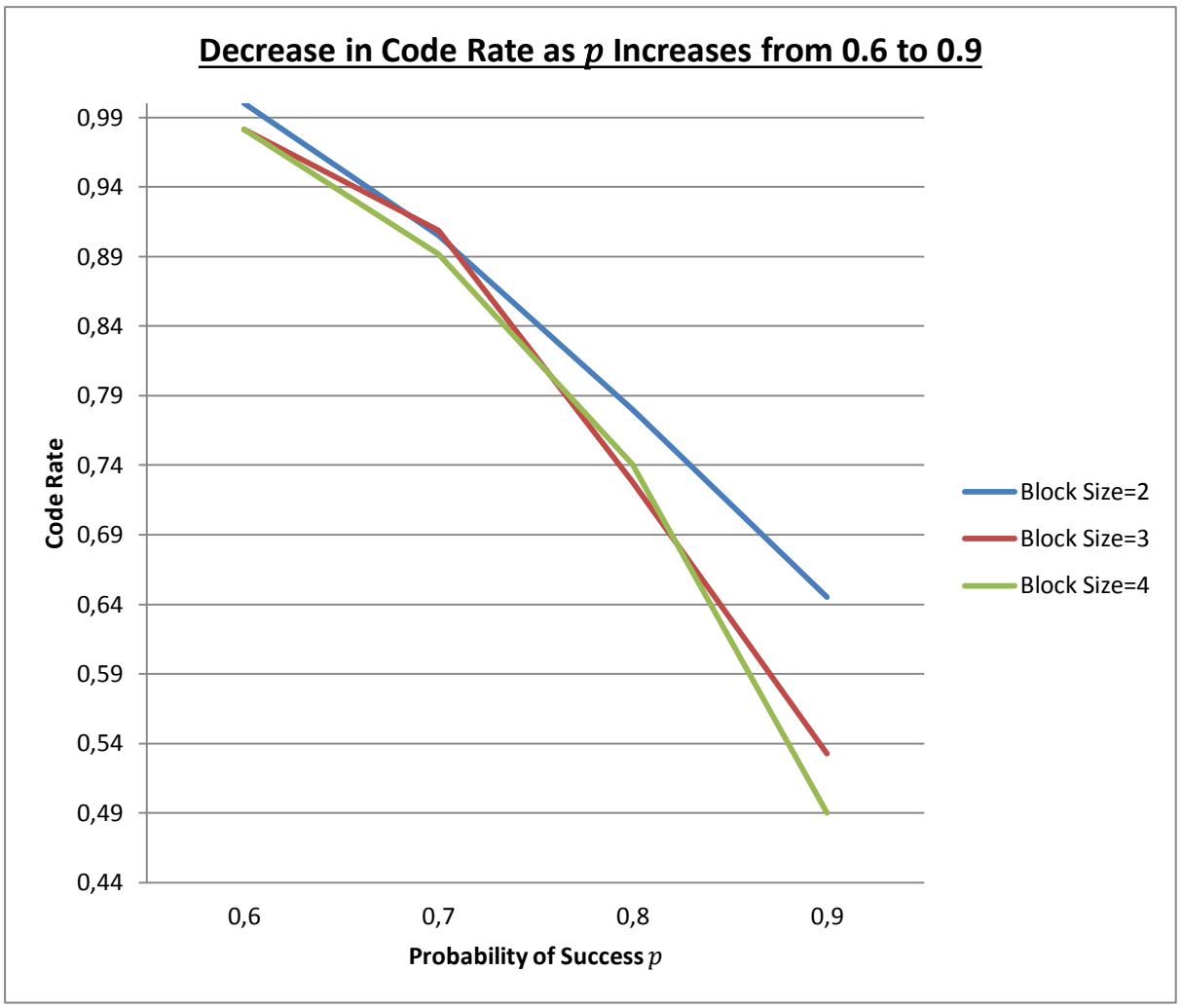


Figure 1.8 Decrease in Code Rate for Increasing Heterogeneity As the probability function shows greater heterogeneity more efficient encoding results from Huffman encoding, which results in a decrease in the code rate.

Figure 1.8 shows that an increase in block size m from $m = 2$ to $m = 3$ results in a *large* decrease in code rate for probabilities p of 0.7 and upwards, while a *smaller* decrease in code rate occurs when the block size is extended to $m = 4$.

1.5 Study2: Huffman Encoding applied to Bernoulli data with Dependence Structure

In 1.4 Study1 we examined the code rate and entropy rate a binary string made up of independently identically distributed Bernoulli distributed random variables. In 1.5 Study2, we will examine the effect of a dependence structure on the entropy, by *creating* a conditional distribution between the random variable constituents of the binary string.

Consider the random variable Y_1 with probability mass function

$$f_{Y_1}(y_1) = \begin{cases} p_1 & if\ \ Y_1 = 1 \\ 1 - p_1 & if\ \ Y_1 = 0 \end{cases}$$

and the conditional probability of $Y_2|Y_1$

$$f_{Y_2|Y_1}(y_2|y_1) = \begin{cases} p_{11} & if\ \ Y_2 = 1|Y_1 = 1 \\ 1 - p_{11} & if\ \ Y_2 = 0|Y_1 = 1 \\ p_{01} & if\ \ Y_2 = 1|Y_1 = 0 \\ 1 - \ p_{01} & if\ \ Y_2 = 0|Y_1 = 0 \end{cases}$$

Then the joint probability mass function $f_{Y_1,Y_2}(y_1, y_2)$ and the marginal probability mass function of Y_2 are

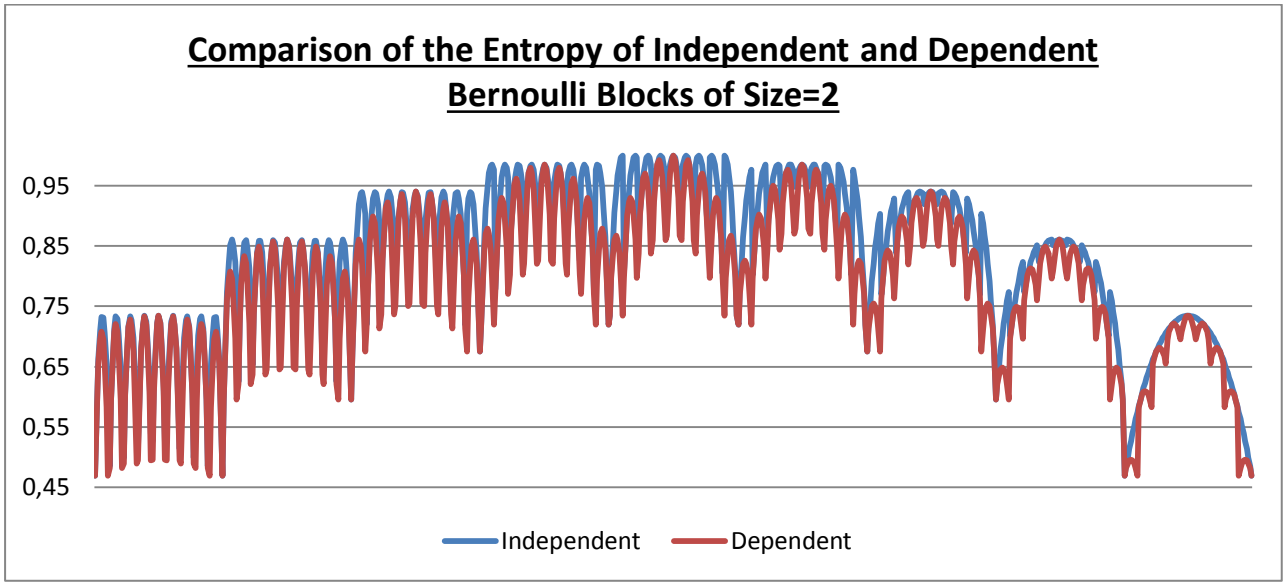
$$f_{Y_1,Y_2}(y_1, y_2) = \begin{cases} p_{11} \cdot p_1 & if\ \ Y_2 = 1\ and\ Y_1 = 1 \\ (1 - p_{11}) \cdot p_1 & if\ \ Y_2 = 0\ and\ Y_1 = 1 \\ p_{01} \cdot (1 - p_1) & if\ \ Y_2 = 1\ and\ Y_1 = 0 \\ (1 - \ p_{01}) \cdot (1 - p_1) & if\ \ Y_2 = 0\ and\ Y_1 = 0 \end{cases}$$

$$f_{Y_1}(y_1) \cdot f_{Y_2}(y_2) = \begin{cases} (\ p_{11} + \ p_{01}) \cdot p_1 & if\ \ Y_2 = 1\ and\ Y_1 = 1 \\ ((1 - \ p_{01}) + (1 - p_{11})) \cdot p_1 & if\ \ Y_2 = 0\ and\ Y_1 = 1 \\ (\ p_{11} + \ p_{01}) \cdot (1 - p_1) & if\ \ Y_2 = 1\ and\ Y_1 = 0 \\ ((1 - \ p_{01}) + (1 - p_{11})) \cdot (1 - p_1) & if\ \ Y_2 = 0\ and\ Y_1 = 0 \end{cases}$$

All possible combinations of p_1, p_{11} and p_{01} are picked by looping through the set $p = \{0.1, 0.2, 0.3, \ldots 0.9\}$. For each combination of p_1, p_{11} and p_{01}, the joint probability mass function $f_{Y_1,Y_2}(y_1, y_2)$ and the product of the marginal probability mass functions $f_{Y_1}(y_1) \cdot f_{Y_2}(y_2)$ are used to calculate the entropy. The entropy calculation is based on the equation in 1.4.2 and is determined for both the independent probabilities and the dependent probabilities.

Code 1.3 SAS 9.2 Code [38]

```
Proc iml;
P={0.10.20.30.40.50.60.70.80.9,
0.90.80.70.60.50.40.30.20.1};
one=j(2,1,1);
doi=1 to 9;
do j=1 to 9;
do k=1 to 9;
Conditional_mat=(p[,j]`//p[,k]`);
Marginal_mat=p[,i];
Joint_mat= Marginal_mat#Conditional_mat;
Joint_Ind_mat= Joint_mat[+,]@Joint_mat [,+];
Self_info_Dep=log2(Joint_mat);
Self_info_Indep=log2(Joint_Ind_mat);
Entropy_Dep_mat= Self_info_Dep#(Joint_mat);
Entropy_Indep_mat=Self_info_Indep#(Joint_Ind_mat);
Entropy_Dep=(Entropy_Dep_mat *one)`*one;
Entropy_Indep=(Entropy_Indep_mat*one)`*one;
Vec=Vec//(Entropy_Indep||Entropy_Dep);
end;end;end;Vec=Vec/2;quit;
```



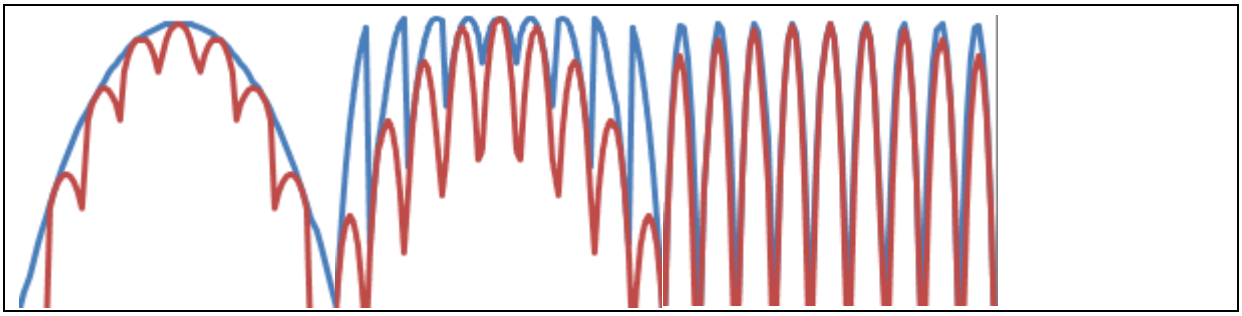


Figure 1.9 Comparison of the Entropy of Independent and Dependent Bernoulli Blocks The joint distributions of dependent random variables and their respective joint independent counterparts.

[38] There is quite a lot of repetition of entropy values in SAS IML output, so Figure 2.7 was chosen out of several other images, that had no repetition, due its *aesthetic* appeal. The output, however is consistent with all the other images and correctly represents the fact (which is proven in the next chapter), that the entropy of a block of random variables with a dependence relation among its constituents has a lower entropy than a block of independently identically distributed random variables.

The dependent Bernoulli strings result in lower or equal entropies for each combination of p_1, p_{11} and p_{01}.

Thus far we have shown that a dependence structure between random variables in a binary string reduces the entropy. How would the code rate be affected by a dependence structure between the random variables constituting the binary string?

In order to compare code rates and entropies for block sizes $N = 2$ and 3 for dependent variables and independent variables, Table 1.7 for $(N = 2)$ and Table 1.8 for $(N = 3)$ were setup. The probabilities for both tables were hand-picked and follow no specific order.

Table 1.7

Marginal_X1		Marginal_X2		Expected	Joint Dependent Prob. Dist.				Huffman Code				Expected	Joint-Indep. Dist.				Huffman Code				Expected	Marginal	Joint-Dep.	Joint-Indep.
1	0	1	0	Block Len.	11	10	01	00	11	10	01	00	Block Len.	11	10	01	00	11	10	01	00	Block Len.	Entropy	Entropy	Entropy
0.1	0.9	0.9	0.1	2	0.05	0.05	0.85	0.05	3	3	1	2	1.25	0.09	0.01	0.81	0.09	3	3	1	2	1.29	0.937991187	0.84758468	0.937991187
0.1	0.9	0.8	0.2	2	0.05	0.05	0.75	0.15	3	3	1	2	1.35	0.08	0.02	0.72	0.18	3	3	1	2	1.38	1.190923688	1.154015773	1.190923688
0.1	0.9	0.7	0.3	2	0.05	0.05	0.65	0.25	3	3	1	2	1.45	0.07	0.03	0.63	0.27	3	3	1	2	1.47	1.350286493	1.336160254	1.350286493
0.1	0.9	0.6	0.4	2	0.05	0.05	0.55	0.35	3	3	1	2	1.55	0.06	0.04	0.54	0.36	3	3	1	2	1.56	1.439946188	1.436666482	1.439946188
0.1	0.9	0.5	0.5	2	0.05	0.05	0.45	0.45	3	3	2	1	1.65	0.05	0.05	0.45	0.45	3	3	2	1	1.65	1.468995594	1.468995594	1.468995594
0.2	0.8	0.9	0.1	2	0.15	0.05	0.75	0.05	2	3	1	3	1.35	0.18	0.02	0.72	0.08	2	3	1	3	1.38	1.190923688	1.154015773	1.190923688
0.2	0.8	0.8	0.2	2	0.1	0.1	0.7	0.1	2	3	1	3	1.50	0.16	0.04	0.64	0.16	2	3	1	3	1.56	1.44385619	1.356779649	1.44385619
0.2	0.8	0.7	0.3	2	0.1	0.1	0.6	0.2	3	3	1	2	1.60	0.14	0.06	0.56	0.24	3	3	1	2	1.64	1.603218994	1.570950594	1.603218994
0.2	0.8	0.6	0.4	2	0.1	0.1	0.5	0.3	3	3	1	2	1.70	0.12	0.08	0.48	0.32	3	3	1	2	1.72	1.692878689	1.685475297	1.692878689
0.2	0.8	0.5	0.5	2	0.1	0.1	0.4	0.4	3	3	2	1	1.80	0.1	0.1	0.4	0.4	3	3	1	2	1.80	1.721928095	1.721928095	1.721928095
0.3	0.7	0.9	0.1	2	0.25	0.05	0.65	0.05	2	3	1	3	1.45	0.27	0.03	0.63	0.07	2	3	1	3	1.47	1.350286493	1.336160254	1.350286493
0.3	0.7	0.8	0.2	2	0.2	0.1	0.6	0.1	2	3	1	3	1.60	0.24	0.06	0.56	0.14	2	3	1	3	1.64	1.603218994	1.570950594	1.603218994
0.3	0.7	0.7	0.3	2	0.1	0.2	0.6	0.1	3	2	1	3	1.60	0.21	0.09	0.49	0.21	2	3	1	3	1.81	1.762581798	1.570950594	1.762581798
0.3	0.7	0.7	0.3	2	0.2	0.1	0.5	0.2	3	3	1	2	1.80	0.21	0.09	0.49	0.21	2	3	1	3	1.81	1.762581798	1.760964047	1.762581798
0.3	0.7	0.6	0.4	2	0.1	0.2	0.5	0.2	3	3	1	2	1.80	0.18	0.12	0.42	0.28	3	3	1	2	1.88	1.852241494	1.760964047	1.852241494
0.3	0.7	0.6	0.4	2	0.2	0.1	0.4	0.3	3	3	1	2	1.90	0.18	0.12	0.42	0.28	3	3	1	2	1.88	1.852241494	1.846439345	1.852241494
0.3	0.7	0.5	0.5	2	0.1	0.2	0.4	0.3	3	3	1	2	1.90	0.15	0.15	0.35	0.35	3	3	2	1	1.95	1.881290899	1.846439345	1.881290899
0.4	0.6	0.9	0.1	2	0.35	0.05	0.55	0.05	2	3	1	3	1.55	0.36	0.04	0.54	0.06	2	3	1	3	1.56	1.439946188	1.436666482	1.439946188
0.4	0.6	0.8	0.2	2	0.3	0.1	0.5	0.1	2	3	1	3	1.70	0.32	0.08	0.48	0.12	2	3	1	3	1.72	1.692878689	1.685475297	1.692878689
0.4	0.6	0.7	0.3	2	0.2	0.2	0.5	0.1	2	3	1	3	1.80	0.28	0.12	0.42	0.18	2	3	1	3	1.88	1.852241494	1.760964047	1.852241494
0.4	0.6	0.6	0.4	2	0.1	0.3	0.5	0.1	3	2	1	3	1.70	0.24	0.16	0.36	0.24	2	2	2	2	2.00	1.941901189	1.685475297	1.941901189
0.4	0.6	0.7	0.3	2	0.3	0.1	0.4	0.2	2	3	1	3	1.90	0.28	0.12	0.42	0.18	2	3	1	3	1.88	1.852241494	1.846439345	1.852241494
0.4	0.6	0.6	0.4	2	0.2	0.2	0.4	0.2	2	2	2	2	2.00	0.24	0.16	0.36	0.24	2	2	2	2	2.00	1.941901189	1.921928095	1.941901189
0.4	0.6	0.5	0.5	2	0.1	0.3	0.4	0.2	3	2	1	3	1.90	0.2	0.2	0.3	0.3	2	2	2	2	2.00	1.970950594	1.846439345	1.970950594
0.4	0.6	0.6	0.4	2	0.3	0.1	0.3	0.3	2	2	2	2	2.00	0.24	0.16	0.36	0.24	2	2	2	2	2.00	1.941901189	1.895461844	1.941901189
0.4	0.6	0.5	0.5	2	0.2	0.2	0.3	0.3	2	2	2	2	2.00	0.2	0.2	0.3	0.3	2	2	2	2	2.00	1.970950594	1.970950594	1.970950594
0.4	0.6	0.4	0.6	2	0.1	0.3	0.3	0.3	2	2	2	2	2.00	0.16	0.24	0.24	0.36	2	2	2	2	2.00	1.941901189	1.895461844	1.941901189

Table 1.7 A Comparison of Code Rates with Dependency Structure for $m = 2$ Compares the code rate (expected block len.) and entropy for a block of size $m = 2$ whose constituents are independently distributed and a binary string whose constituents have a dependence structure, introduced by a conditional probability distribution (not included on Table 1.7). The three (out-of-four) shades of grey tables reading from left-to-right are the marginal probabilities, the joint dependent probabilities and the joint independent probabilities. The fourth grey table (far right) consists of the representative entropies for the marginal, dependent and independent probability distributions. The Huffman code column contains the codeword length of the **bold** vector at the top of each column.

Table 1.8

Marginal_X1		Marginal_X2		Marginal_X3		Expected	Joint Dependent Prob. Dist.								Huffman Code Length								Expected	Marginal	Joint-Dep.
1	0	1	0	1	0	Block Len.	000	001	010	011	110	111	100	101	000	001	010	011	110	111	100	101	Block Len.	Entropy	Entropy
0.5	0.5	0.5	0.5	0.5	0.5	3	0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125	3	3	3	3	3	3	3	3	3.00	3	3
0.5	0.5	0.4	0.6	0.5	0.5	3	0.05	0.25	0.1	0.1	0.1	0.1	0.25	0.05	4	2	3	4	4	3	2	4	2.80	2.9709506	2.760964
0.46	0.54	0.76	0.24	0.47	0.53	3	0.08	0.08	0.08	0.3	0.3	0.08	0.07	0.01	3	4	4	2	2	3	4	4	2.64	2.7878203	2.543207
0.525	0.475	0.75	0.25	0.5	0.5	3	0.05	0.05	0.35	0.025	0.025	0.35	0.075	0.075	5	5	1	5	5	2	4	4	2.40	2.809474	2.3190353
0.51	0.49	0.12	0.88	0.49	0.51	3	0.4	0.03	0.03	0.03	0.03	0.03	0.05	0.4	1	5	5	5	5	4	4	2	2.12	2.5287837	2.0324729
0.325	0.675	0.4	0.6	0.325	0.675	3	0.45	0.025	0.1	0.1	0.1	0.1	0.025	0.1	1	5	4	4	3	3	5	4	2.50	2.7904228	2.4454618
0.71	0.29	0.285	0.715	0.71	0.29	3	0.075	0.07	0.075	0.07	0.07	0.07	0.07	0.5	3	4	4	4	4	4	4	1	2.43	2.599617	2.4033203
0.29	0.71	0.3	0.7	0.28	0.72	3	0.55	0.04	0.04	0.08	0.03	0.15	0.1	0.01	1	4	5	3	6	3	3	6	2.14	2.605463	2.0983331
0.77	0.23	0.19	0.81	0.69	0.31	3	0.08	0.01	0.09	0.05	0.02	0.03	0.12	0.6	3	6	3	4	6	5	3	1	2.00	2.3726562	1.9605878
0.21	0.79	0.78	0.22	0.69	0.31	3	0.07	0.01	0.06	0.65	0.05	0.02	0.13	0.01	4	6	4	1	4	5	2	6	1.85	2.3948237	1.7605509
0.88	0.12	0.8	0.2	0.155	0.845	3	0.045	0.045	0.02	0.01	0.7	0.07	0.08	0.03	4	4	5	5	1	3	3	4	1.78	1.8735016	1.6540011
0.14	0.86	0.86	0.14	0.19	0.81	3	0.02	0.07	0.75	0.02	0.03	0.06	0.01	0.04	4	3	1	5	4	3	5	4	1.65	1.8699491	1.4530807
0.905	0.095	0.87	0.13	0.86	0.14	3	0.03	0.01	0.035	0.02	0.015	0.8	0.06	0.03	4	5	4	4	5	1	3	3	1.54	1.5946195	1.2440864
0.8775	0.1225	0.0575	0.9425	0.095	0.905	3	0.0125	0.07	0.03	0.01	0.0125	0.005	0.85	0.01	4	3	3	4	4	5	1	5	1.37	1.306887	0.948762
0.065	0.935	0.055	0.945	0.95	0.05	3	0.015	0.9	0.01	0.01	0.015	0.02	0.01	0.02	4	1	5	5	4	3	4	3	1.28	0.9406466	0.7436395

Marginal_X1		Marginal_X2		Marginal_X3		Expected	Joint Independent Prob. Dist.								Huffman Code Length								Expected	Marginal	Joint-Indep.
1	0	1	0	1	0	Block Len.	000	001	010	011	110	111	100	101	000	001	010	011	110	111	100	101	Block Len.	Entropy	Entropy
0.5	0.5	0.5	0.5	0.5	0.5	3	0.1250	0.1250	0.1250	0.1250	0.1250	0.1250	0.1250	0.1250	3	3	3	3	3	3	3	3	3.00	3	3
0.5	0.5	0.4	0.6	0.5	0.5	3	0.1500	0.1500	0.1000	0.1000	0.1000	0.1000	0.1500	0.1500	3	3	3	3	3	3	3	3	3.00	2.9709506	2.9709506
0.46	0.54	0.76	0.24	0.47	0.53	3	0.0687	0.0609	0.2175	0.1929	0.1853	0.1643	0.0585	0.0519	4	4	2	2	3	3	4	4	2.83	2.7878203	2.7878203
0.525	0.475	0.75	0.25	0.5	0.5	3	0.0594	0.0594	0.1781	0.1781	0.1969	0.1969	0.0656	0.0656	4	4	3	3	2	2	4	4	2.86	2.809474	2.809474
0.51	0.49	0.12	0.88	0.49	0.51	3	0.2199	0.2113	0.0300	0.0288	0.0312	0.0300	0.2289	0.2199	2	3	5	5	5	5	2	2	2.57	2.5287837	2.5287837
0.325	0.675	0.4	0.6	0.325	0.675	3	0.2734	0.1316	0.1823	0.0878	0.0878	0.0423	0.1316	0.0634	2	3	2	4	3	4	3	4	2.74	2.7904228	2.7904228
0.71	0.29	0.285	0.715	0.71	0.29	3	0.0601	0.1472	0.0240	0.0587	0.0587	0.1437	0.1472	0.3604	4	3	4	4	4	3	2	2	2.69	2.599617	2.599617
0.29	0.71	0.3	0.7	0.28	0.72	3	0.3578	0.1392	0.1534	0.0596	0.0626	0.0244	0.1462	0.0568	2	3	2	4	4	4	4	4	2.84	2.605463	2.605463
0.77	0.23	0.19	0.81	0.69	0.31	3	0.0578	0.1285	0.0135	0.0302	0.0454	0.1009	0.1933	0.4304	4	3	6	6	5	3	3	1	2.42	2.3726562	2.3726562
0.21	0.79	0.78	0.22	0.69	0.31	3	0.0539	0.1199	0.1910	0.4252	0.0508	0.1130	0.0143	0.0319	4	3	3	1	5	3	6	6	2.44	2.3948237	2.3948237
0.88	0.12	0.8	0.2	0.155	0.845	3	0.0203	0.0037	0.0811	0.0149	0.5949	0.1091	0.1487	0.0273	6	7	4	7	1	3	2	5	1.93	1.8735016	1.8735016
0.14	0.86	0.86	0.14	0.19	0.81	3	0.0975	0.0229	0.5991	0.1405	0.0975	0.0229	0.0159	0.0037	3	4	1	3	3	5	6	6	1.93	1.8699491	1.8699491
0.905	0.095	0.87	0.13	0.86	0.14	3	0.0017	0.0106	0.0116	0.0711	0.1102	0.6771	0.0165	0.1012	6	6	5	3	3	1	4	3	1.72	1.5946195	1.5946195
0.8775	0.1225	0.0575	0.9425	0.095	0.905	3	0.1045	0.0110	0.0064	0.0007	0.0457	0.0048	0.7485	0.0786	2	5	6	7	4	7	1	3	1.51	1.306887	1.306887
0.065	0.935	0.055	0.945	0.95	0.05	3	0.0442	0.8394	0.0026	0.0489	0.0002	0.0034	0.0031	0.0584	4	1	7	3	7	5	6	2	1.33	0.9406466	0.9406466

Table 1.8 A Comparison of Code Rates with Dependency Structure for $m = 3$ Compares the code rate (expected block len.) and entropy for a block of size $m = 3$ whose constituents are independently distributed and a binary string whose constituents have a dependence structure, introduced by a conditional probability distribution (not included on Table 1.8).The top table consists of the marginal probabilities and the joint dependent probabilities with respective expected block lengths and entropy code rates. The bottom table consists of the same marginal probabilities as the top table, but calculates the independent probability distribution and respective expected block lengths and entropy code rates (for both the marginal and joint independent probabilities).The Huffman code column contains the codeword length of the **bold** vector at the top of each column.

Several remarks can be made with regards to Table 1.7 and Table 1.8.

Firstly, the entropy code rates for dependent random variables are lower than their independent and marginal counterparts (as shown in Study 2), this can be seen by comparing the marginal entropy code rate to the joint independent and dependent entropy code rates and the joint independent and dependent entropy code rates to one another.

Secondly, the code rate per block decreases with increasing block size, i.e. the code rate for marginal probabilities – block size $m = 1$, compared to the joint independently distributed and dependently distributed – block size $m = 2$, is greater (as shown in Study 1).

Lastly, the code rate decreases when there is dependence structure in the binary string compared to where there is none.

1.6 Conclusions of Chapter One

- Data compression is the reduction of bits required for the encoding of data.
- Compression systems reduce redundancy in data much of which is statistical in nature.
- There are two types of data compression, lossless and lossy data compression, lossless data compression implies that the data can be perfectly reconstructed, while lossy implies that there will be error in the reconstructed values.
- Data can be encoded as individual random variables, or as blocks (or random vectors).
- A binary codeword is a sequence or sub-string of "$0's$"and "$1's$", it is a binary representative of the output value y_i or vector $\boldsymbol{y}^T = (y_1\ y_2\ \dots\ y_m)$ of dimension m, sampled from the information source.
- The code rate r is the number of bits required to uniquely decode each individual output value y_i or block $\boldsymbol{y}^T = (y_1\ y_2\ \dots\ y_m)$. The code rate is a measure of encoding efficiency.
- Fixed rate encoding uses the same number of bits to encode each of the output values, while variable rate encoding uses a varying number of bits to encode each of the output values.
- Unique decipherability is a necessary condition for lossless encoding and instantaneous decoding is a good encoding property that is satisfied by the prefix condition.
- The binary tree encoding algorithm and Huffman encoding both satisfy the prefix condition.
- Huffman Coding is optimal in the sense that it is able to encode data at a code rate that may achieve the entropy lower bound.

The effects of heterogeneity, block size and statistical dependence were examined for varying size blocks of Bernoulli distributed random variables (vectors) on the code rate and entropy. The following observations were made

- Increased heterogeneity leads to a decrease in code rate and entropy of a particular Bernoulli probability mass function.

- Increased block size will result in a decrease code rate and entropy, that is if the probability distribution is not homogeneous and the constituent random variables of the block are not independent. If the constituent random vectors making up the block are independent, then the code rate will still decrease with an increase in block size.

- Heterogeneity is directly related to the size of the block required to bring the code rate within a neighbourhood of the entropy measure of the particular joint probability mass function when Huffman encoding is used, an increase in heterogeneity will require an increase in block size in order to achieve a decrease in code rate within a particular neighbourhood of entropy.

- Dependency relationships between the random variables within a block affect the entropy and code rate of encoded random vectors. Dependent random variables incur a lower entropy than the independent random variables obtained from the marginal probability mass function of the dependent random variables.

- If there is substantial heterogeneity in the probabilities of the underlying probability mass function, the difference between the code rate obtained by a Huffman block encoding of smaller size blocks and the entropy code rate is larger. In other words, larger blocks are required to obtain marginally better code rates for strongly heterogeneous probability distributions than for more homogeneous probabilities.

Chapter Two:
Information Theory

2.1 What is information?

Information is a message, a pattern, a sensory input or abstract data. Information has a physical underlying representation [39].

The measure of the amount of information is the measure of the freedom of choice, uncertainty or entropy of a message or data string being selected out of all possible messages.

According to [40] the basis of randomness lies in probability theory, while uncertainty is related to information theory.

The information content of a (discrete) random variable is dependent on the amount of uncertainty as to the outcome of an experiment. If the random variable is highly homogeneous, say for example it is a uniformly distributed, with a few number of outcomes, then the information content will be high due to the level of uncertainty. An increase in the number of outcomes of a uniformly distributed random variable is directly related to the level of uncertainty [41].

If however certain outcomes are more probable than others, in other words, the random variable is more heterogeneous (than a uniform random variable) then the uncertainty pertaining to the outcome of an experiment will be less. An increase in the number of possible outcomes of a non-uniform random variable will increase the level of uncertainty by a rate that is less than that of a uniform random variable and this increase is inversely related to the heterogeneity of the random variable.

2.2 Information in a String

In terms of the ideas presented in the previous chapter, the above statements can be applied to a random string of data.

The amount of information in a random string is dependent on the underlying probability structure of the string, but independent of its encoding. We assume that the probability distribution is discrete unless stated otherwise.

[39] As pointed out, it does have a physical representation, however, it does not necessarily have a particular interpretation.
[40] Cambel, A.B. Applied Chaos Theory: A Paradigm for Complexity. Academic press. 1993. Page 8.
[41] An increase in the number of outcomes results in a logarithmic increase in uncertainty, the logarithmic nature will be discussed later in this chapter.

The information content of a string is quantified by the joint probability mass function of the random string. It is therefore related to the *number* of output values and the homogeneity of the data contained within the string.

Therefore the joint probability mass function of the set of all possible output strings is very heterogeneous, then the information content is low and if the probability mass function is very homogeneous, then the information content is high. The information content will increase with the increase of the length of the string.

A string with high information content will in general require a greater number of bits to store or transmit than one with lower information content.

If a string or message is encoded it is typically made up of an information component and coding redundancy space, the code rate then indicates the amount of the space required to store (or transmit) the information contained within the string and the redundant component of the encoding process. If an encoding process has less redundancy (than another), then the process is said to be more efficient, if the coding redundancy is zero, then the code rate is said to be optimal.

2.3 An Intuitive Introduction to Uncertainty and Entropy

Uncertainty and the Additive Property

The uncertainty of a random variable is defined as the number of bits required to encode any outcome of an experiment.

Consider a discrete uniformly distributed random variable Y with an output set $\{y_i\}_{i=1}^{k}$ of size k. The number of bits required to encode any y_i is $log_2 k$. This can be written as $-log_2\left(\frac{1}{k}\right) = -log_2(f_Y(y_i))\ \forall\ i = 1, 2, \dots, k.$

An increase in size of the output set, k results in an increase in the number of bits that are required to encode any y_i.

For more than one random variable, consider a random vector of m independent discrete uniform random variables $\boldsymbol{Y}^T = (Y_1\ Y_2\ Y_3\ \dots\ Y_m)$, the output set has the size $k_1 \times k_2 \times k_3 \times \dots \times k_m$ to encode any output vector of the random vector $\boldsymbol{Y}$ requires $log_2(k_1 \times k_2 \times k_3 \times \dots \times k_m) = \sum_{i=1}^{k} log_2(k_i)$ bits. The uncertainty of independent (uniform) random variables is equal to the sum of their individual uncertainties, this is referred to as the additive property of uncertainty. [42]

The entropy H of a random variable is its expected uncertainty.

[42] The random variables need not be uniform, nor identically distributed. This will be shown later in this chapter.

The Branching Property

Consider a random variable Y which has three outcomes $\{y_i\}_{i=1}^{3}$ each with a corresponding probability $\{p_1, p_2, p_3\}$. If we were to group outcomes y_2 and y_3 and rename it b_2 and rename y_1 into b_1, then by grouping the two outputs, we have

$$f_{Y|B}(y_1|b_j) = \begin{cases} 1 \; if \; j = 1 \\ 0 \; if \; j = 2 \end{cases}$$

$$f_{Y|B}(y_2|b_j) = \begin{cases} 0 & if \; j = 1 \\ \dfrac{p_2}{p_2 + p_3} & if \; j = 2 \end{cases}$$

$$f_{Y|B}(y_3|b_j) = \begin{cases} 0 & if \; j = 1 \\ \dfrac{p_3}{p_2 + p_3} & if \; j = 2 \end{cases}$$

$$f_B(b_j) = \begin{cases} p_1 & if \; j = 1 \\ p_2 + p_3 & if \; j = 2 \end{cases}$$

The expected uncertainty or entropy of picking y_1, y_2 or y_3 in an experiment maybe determined by considering the experiment in two stages.

The first stage requires picking b_1 or b_2, the expected uncertainty of picking either b is $H(p_1, p_2 + p_3)$.

The second stage requires picking y_1, y_2 or y_3 from out of the previously picked group. If b_1 was picked in the first stage, then it is certain that y_1 will be chosen from it. If b_2 was picked, then y_2 or y_3 may be picked with the uncertainty given by $H\left(\frac{p_2}{p_2+p_3}, \frac{p_3}{p_2+p_3}\right)$.

Therefore the entropy or the expected uncertainty of picking y_1, y_2 or y_3 following the two stage process is given by

$$H(p_1, p_2, p_3) = H(p_1, p_2 + p_3) + p_1 \cdot H(1) + (p_2 + p_3) \cdot H\left(\frac{p_2}{p_2 + p_3}, \frac{p_3}{p_2 + p_3}\right).$$

$H(1) = 0$ since the uncertainty related to picking y_1 from b_1 is zero, so

$$H(p_1, p_2, p_3) = H(p_1, p_2 + p_3) + (p_1) \cdot 0 + (p_2 + p_3) \cdot H\left(\frac{p_2}{p_2 + p_3}, \frac{p_3}{p_2 + p_3}\right).$$

If we replace p_1 with a_1, p_2 with $\lambda \cdot a_2$ and p_3 with $(1 - \lambda) \cdot a_2$ then we have

$$H(a_1, \lambda \cdot a_2, (1 - \lambda) \cdot a_2) = H(a_1, a_2) + (a_2) \cdot H\big(\lambda, (1 - \lambda)\big).$$

This is known as the recursion formula or branching formula and will be used to derive the entropy equation.

Entropy

The entropy of a random variable is a measure of the amount of information required on the average to describe a random variable [43].

The entropy is the measure of the amount of uncertainty before sampling and the amount of information obtained through sampling [44].

As seen earlier in 3.3, the uncertainty of a uniformly distributed random variable is $log_2 k$ or more generally $-log_2(f_Y(y_i)) \forall i = 1, 2, \dots, k$, the more general definition is applicable to non-uniform random variables as well. The expected uncertainty for the random variable Y is given below, it is a special case of definition given in Chapter One, section 1.4.1 where $m = 1$

$$\mathbb{E}_Y[-log_2(f_Y(y))] = -\sum_{i=1}^{k} f_Y(y_i) \cdot log_2(f_Y(y_i))$$

and $-0 \cdot log_2(0) = 0$ for convention.

The derivation of the above expression will follow in the next two sections, 3.4 and 3.5.

An alternative, simpler and more intuitive derivation is included in section 3.12 the simpler derivation assumes independence and identical distribution with probabilities all rational.

[43] Cover, T. M. and Thomas, J. A. Elements of Information Theory. Wiley Series in Telecommunications and Signal Processing. Second Edition. Page 15.
[44] Roman, S. Coding and Information Theory. Springer-Verlag. 1997. Page 11.

2.4 The Derivation of Entropy

The derivation of entropy $H(\cdot)$ is done with the help of four conditions relating to information as a function of the probabilities associated with each outcome value of a discrete random variable.

Let Y be a discrete random variable with associated probability mass function $f_Y(y)$ and outcome value set $\{y_i\}_{i=1}^{k}$.

Consider a function $H(\cdot)$ for which the four conditions hold [45].

1. Regularity: $H(p, 1-p)$ is Lebesgue integrable on the interval $0 \leq p \leq 1$.
2. Symmetry: $H(p_1, p_2, p_3, \dots, p_k)$ is symmetrical in each of its variables.
3. Recursion: for every $0 \leq \lambda < 1$,
$$H\big(p_1, p_2, p_3, \dots, p_{j-1}, \lambda \cdot p_j, (1-\lambda) \cdot p_j\big) = H\big(p_1, p_2, p_3, \dots, p_{j-1}, p_j\big) + \Big(p_j \cdot H(\lambda, 1-\lambda)\Big).$$
4. Normalization: $H\left(\frac{1}{2}, \frac{1}{2}\right) = 1$.

The four conditions are satisfied by the function

$$H(p_1, p_2, p_3, \dots, p_k) = -\sum_{i=1}^{k} p_i \cdot log_2(p_i)$$

which is the entropy of the random variable Y.

Condition (1) ensures that the function is bounded and continuous on the interval $0 \leq p \leq 1$ except possibly at a countable number of values.

Condition (2) implies that the ordering of the probabilities, do not affect the entropy of the random variable.

Condition (3) was described above and is a property of uncertainty.

Property (4) ensures that the entropy is 1 when there is maximum homogeneity and two outcomes.

Starting at the recursion formula (3) for $j = 2$.

For notational simplicity let $p = p_1$ and $1 - p = p_2$ and $u = \lambda \cdot (1-p)$ and $v = (1-\lambda) \cdot (1-p)$ where $p \neq 0$ then it follows that

$$H(p, u, v) = H(p, u+v) + \left((u+v) \cdot H\left(\frac{u}{(u+v)}, \frac{v}{(u+v)}\right)\right) \qquad (P1)$$

[45] There are many different derivations of the entropy function based on similar axioms, this derivation follows the one described Mathai, A.M. and Rathie, P.N. Basic Concepts in Information Theory and Statistics. Wiley. 1975. Pages 6-8.

due to the symmetry property (2), $H(p,u,v) = H(u,p,v)$ where

$$H(u,p,v) = H(u,p+v) + \left((p+v)\cdot H\left(\frac{p}{(p+v)},\frac{v}{(p+v)}\right)\right).$$

Therefore

$$H(p,u+v) + \left((u+v)\cdot H\left(\frac{u}{(u+v)},\frac{v}{(u+v)}\right)\right)$$
$$= H(u,p+v) + \left((p+v)\cdot H\left(\frac{p}{(p+v)},\frac{v}{(p+v)}\right)\right).$$

Let $g(p) = H(p,1-p)$ with $0<p<1, 0<u<1$

$$g(p) + (1-p)\cdot g\left(\frac{u}{(1-p)}\right) = g(u) + (1-u)\cdot g\left(\frac{p}{(1-u)}\right). \qquad (P2)$$

Then by integrating over the variable u from 0 to $1-p$ (we replace u with t as is good practice).

$$\int_0^{1-p}\left(g(p) + (1-p)\cdot g\left(\frac{t}{(1-p)}\right)\right)dt = \int_0^{1-p}\left(g(t) + (1-t)\cdot g\left(\frac{p}{(1-t)}\right)\right)dt.$$

Using the substitutions

$$m = \frac{t}{(1-p)} \quad dm = \frac{dt}{(1-p)}$$
$$n = \frac{p}{(1-t)} dn = -\frac{p}{(1-t)^2}dt$$

$$(1-p)\cdot g(p) + (1-p)^2\cdot\int_0^1 g(m)dm = \int_0^{1-p} g(t)dt + p^2\cdot\int_p^1 n^{-3}\cdot g(n)dn. \qquad (P3)$$

Now since $g(p)$ is Lebesgue integrable for $0\le p\le 1$, then all terms of the above equation are absolutely continuous, except possibly for $g(p)$ itself, however, since the equality holds for all $0\le p\le 1$, then $g(p)$ is also absolutely continuous on $0\le p\le 1$.

So then by differentiating on the LHS and RHS in terms of p

$$\frac{d}{dp}\left((1-p)\cdot g(p) + (1-p)^2\cdot\int_0^1 g(m)dm\right) = \frac{d}{dp}\left(\int_0^{1-p} g(t)dt + p^2\cdot\int_p^1 n^{-3}\cdot g(n)dn\right)$$
$$(1-p)\cdot g'(p) - g(p) - 2(1-p)\cdot\int_0^1 g(m)dm$$
$$= -g(1-p) + 2p\cdot\int_p^1 n^{-3}\cdot g(n)dn - p^{-1}\cdot g(p).$$

But due to the symmetry property (2)

$$g(p) = H(p, 1-p) = H(1-p, p) = g(1-p).$$

Then by cancelling $g(p)$ on the LHS and $g(1-p)$ on the RHS

$$(1-p)\cdot g'(p) - 2(1-p)\cdot \int_0^1 g(m)dm = 2p\cdot \int_p^1 n^{-3}\cdot g(n)dn - p^{-1}\cdot g(p). \quad (P4)$$

This can be rewritten as

$$g'(p) = 2\cdot \int_0^1 g(m)dm + 2\left(\frac{p}{1-p}\right)\cdot \int_p^1 n^{-3}\cdot g(n)dn - \frac{1}{p\cdot(1-p)}\cdot g(p). \quad (P5)$$

By differentiating $g'(p)$ in terms of p and replacing $g'(p)$with $(P5)$ and $g(p)$ with $(P3)$ this reduces to

$$g''(p) = -2\cdot\left(\frac{1}{p\cdot(1-p)}\right)\cdot \int_0^1 g(t)dt. \quad (P6)$$

Then by integrating over p, and the fact that

$$\frac{1}{p\cdot(1-p)} = \frac{1}{p} + \frac{1}{1-p}.$$

An alternative expression for $g'(p)$ can be obtained (where c_1 is a constant of integration).

$$g'(p) = -2\cdot(ln(p) - ln(1-p))\cdot \int_0^1 g(t)dt + c_1.$$

By integrating over p again we get

$$g(p) = c_1\cdot p + c_2 - 2\cdot[p\cdot ln(p) + (1-p)\cdot ln(1-p)]\cdot \int_0^1 g(t)dt. \quad (P7)$$

From symmetry $g(x) = g(1-x)$

$$g(1-p) = c_1\cdot(1-p) + c_2 - 2\cdot[(1-p)\cdot ln(1-p) + p\cdot ln(p)]\cdot \int_0^1 g(t)dt. \quad (P8)$$

Then by equating the two equations $(P7)$ and $(P8)$, it follows that $c_1, c_2 = 0$, therefore

$$g(p) = -2\cdot[p\cdot ln(p) + (1-p)\cdot ln(1-p)]\cdot \int_0^1 g(t)dt.$$

$$g(p) = -2\cdot log_2 e\cdot[p\cdot log_2(p) + (1-p)\cdot log_2(1-p)]\cdot \int_0^1 g(t)dt. \quad (P9)$$

Due to normalization (4), $H\left(\frac{1}{2}, \frac{1}{2}\right) = 1$, then $g\left(\frac{1}{2}\right) = 1$, so by substituting $p = \frac{1}{2}$ into $(P9)$

$$-2\cdot log_2 e\cdot\left[\left(\frac{1}{2}\right)\cdot log_2\left(\frac{1}{2}\right) + \left(1-\left(\frac{1}{2}\right)\right)\cdot log_2\left(1-\left(\frac{1}{2}\right)\right)\right]\cdot \int_0^1 g(t)dt = 1.$$

$$\Rightarrow \int_0^1 g(t)dt = \frac{1}{2 \cdot log_2 e}. \qquad (P10)$$

Therefore by combining $(P9)$and $(P10)$ we get

$$g(p) = -p \cdot log_2(p) - (1-p) \cdot log_2(1-p) \qquad (P11)$$

where $0 \leq p \leq 1$ and allowing $0 \cdot \log_2(0) = 0$.

$$H(p, 1-p) = -p \cdot log_2(p) - (1-p) \cdot log_2(1-p).$$

For $0 \leq p \leq 1$

By replacing p with p_1 and $1-p$ with p_2 we get

$$H(p_1, p_2) = -\sum_{i=1}^{2} p_i \cdot log_2(p_i). \qquad (P12)$$

By applying the recursion formula (3) we get

$$H(p_1, p_2, p_3, \dots, p_k) = -\sum_{i=1}^{k} p_i \cdot log_2(p_i). \qquad (P13)$$

It can be shown that the entropy is bounded below by zero in the minimum uncertainty or certain case where $p_i = 0$ for every i except for one outcome, where $p_j = 1$. Entropy is bounded above by $log_2(k)$ where $Y \sim Unif\left(\frac{1}{k}\right)$ which corresponds to the maximum uncertainty for a discrete random variable.

2.5 The Kraft inequality for a Binary Codebook

The derivation of the entropy formula in section 2.4 deals solely with the discrete random variable and its underlying probability mass function. In order to connect the idea of instantaneous encoding/decoding with the entropy function, the Kraft inequality will firstly be derived and then used to show that the minimum code rate for a uniquely decipherable encoding is the entropy rate of the encoded random variable.

The derivation of the Kraft inequality is completed first and it is given in two parts, **Part One** shows that an instantaneous code with corresponding lengths $\{l_1, l_2 \dots, l_k\}$ must satisfy the Kraft inequality. **Part Two** shows that there is an instantaneous code with corresponding lengths $\{l_1, l_2 \dots, l_k\}$ that satisfies the Kraft inequality.

The Kraft inequality is then used to show that a uniquely decipherable encoding process cannot achieve a lower code rate than the entropy of the encoded random variable.

2.5.1 The Kraft inequality[46]

For a variable length encoding of a random variable Y with an output set $y = \{y_i\}_{i=1}^{k}$, the encoding is uniquely decipherable [47] if the binary codeword lengths $l_i = l\big(\mathfrak{E}(y_i)\big)$ satisfy the inequality

$$\sum_{i=1}^{k} 2^{-l_i} \leq 1.$$

The Kraft inequality is both the necessary and sufficient condition for the existence of a unique decipherable code and hence instantaneous code.

2.5.1.1 Part One

Consider the set of instantaneous binary codewords $C = \{e_1, e_2 \dots, e_k\}$ of an output value set $\{y_j\}_{j=1}^{k}$ with corresponding lengths $\{l_1, l_2 \dots, l_k\}$.

Any codeword [48] $e_i \in C$ can be written as $e_i = \left(e_{i,1}e_{i,2} \dots e_{i,l_i-1}e_{i,l_i}\right)$ as described in the previous chapter.

Let the maximum length of a binary codeword in C be $l_{Max} = max\left(\{l_j\}_{j=1}^{k}\right)$, then the total number of possible binary codewords of length l_{Max} is $2^{l_{Max}}$, with a structure given by

$$e_{Max} = e_{i,1}e_{i,2} \dots e_{i,l_i-1}e_{i,l_i}e_{i,l_i+1}\, e_{i,l_i+2} \dots e_{i,l_{Max}}$$

The expression above, shows that if e_i has the same first l_i values as the codeword e_{Max}, then it is a prefix of e_{Max}. For each codeword e_i there are $2^{(l_{Max}-l_i)}$ codewords that are prefixes of e_{Max} and will not be permitted into the codebook C.

The codebook C is either empty or contains a positive integer number of codewords, therefore

$$0 \leq 2^{l_{Max}} - 2^{l_{Max}} \cdot \sum_{i=1}^{k} 2^{-l_i}.$$

Binary codewords of length $l_j \leq l_{Max}$ permitted to be in C.

The inequality can be factored

$$0 \leq 2^{l_{Max}} \left(1 - \sum_{i=1}^{k} 2^{-l_i}\right).$$

Since $2^{l_{Max}}$ is non-zero and positive , the Kraft inequality is obtained, by dividing by $2^{l_{Max}}$.

[46] The proof given below is similar to one sketched in Roman, S. Coding and Information Theory. Springer-Verlag. 1997. Pages 44-46.

[47] The Kraft inequality will be proven for the specific case of instantaneous coding, however this holds for the more general case of uniquely decipherable coding.

[48] For notational simplicity, the transpose vector notation for the codeword will be dropped.

$$0 \leq \left(1 - \sum_{i=1}^{k} 2^{-l_i}\right).$$

$$\sum_{i=1}^{k} 2^{-l_i} \leq 1.$$

2.5.1.2 Part Two

We show that a codebook comprising of codewords that satisfies both the Kraft inequality and the prefix condition can be constructed.

Let N_j be the number of codewords of length j. The number of codewords of length 1 is therefore N_1.

Note that $N_1 \leq 2$, if

Case of codeword up till length 1

$N_1 = 2$ the codebook is $C = \{0, 1\}$
$N_1 = 1$ the codebook is $C = \{0\}$ or $C = \{1\}$
$N_1 = 0$ the codebook is $C = \{\emptyset\}$

If $N_1 < 2$, then $N_2 \leq 2^2 - 2 \cdot N_1$, if

Case of codeword up till length 2

$N_1 = 1$ and $N_2 = 2$ the codebook is $C = \{0{,}10{,}11\}$ or $C = \{1{,}01{,}00\}$
$N_1 = 1$ and $N_2 = 1$the codebook is $C = \{0{,}10\}$ or $C = \{0{,}11\}$ or $C = \{1{,}01\}$ or $C = \{1{,}00\}$
$N_1 = 0$ then $N_2 \leq 4$ and the codebook is $C = \{11{,}10{,}01{,}00\}$.

If $N_1 < 2$ and $N_2 < 2^2 - 2 \cdot N_1$, then $N_3 \leq 2^3 - 2^2 \cdot N_1 - 2 \cdot N_2$, that is if

Case of codeword up till length 3

$N_1 = 1$ and $N_2 = 1$ and $N_3 = 3$ then the codebook
$C = \{0, 10{,}110{,}111\}$ or $C = \{1, 01{,}001{,}000\}$ or $C = \{0, 11{,}100{,}101\}$ or $C = \{1, 00{,}011{,}010\}$.

Case of codeword up till length n

The encoding satisfies the inequalities

$N_1 < 2$ and $N_2 < 2^2 - 2 \cdot N_1$ and so on, to the number of strings of length n

$$N_n \leq 2^n - 2^{n-1} \cdot N_1 - 2^{n-2} \cdot N_2 - \cdots - 2 \cdot N_{n-1}.$$

What is left to be shown is that this encoding scheme satisfies the Kraft inequality.

Multiplying the previous expression on both sides of the inequality by 2^{-n} leads to

$$2^{-n} \cdot N_n \leq 1 - 2^{-1} \cdot N_1 - 2^{-2} \cdot N_2 - \cdots - 2^{1-n} \cdot N_{n-1}.$$

This can be written as

$$2^{-n} \cdot N_n + 2^{1-n} \cdot N_{n-1} + 2^{2-n} \cdot N_{n-2} - \cdots - 2^{-1} \cdot N_1 \leq 1.$$

This can be rewritten in sigma notation as

$$\sum_{i=1}^{n} 2^{-i} \cdot N_i \leq 1$$
$$\sum_{j=1}^{n} \sum_{i=1}^{N_j} 2^{-j} \leq 1.$$

Since the total number of codewords is $\sum_{i=1}^{n} N_i = k$, we obtain the Kraft inequality

$$\sum_{i=1}^{k} 2^{-l_i} \leq 1.$$

2.5.2 Code Rate for Instantaneous Encoding[49]

Let Y be a discrete random variable with associated probability mass function f_Y and output value set $\{y_i\}_{i=1}^{k}$.The code rate is the average codeword length and is given by r, so with a little algebra,

$$r = \mathbb{E}(l_i) = \sum_{i=1}^{k} l_i \cdot f_Y(y_i) = -\sum_{i=1}^{k} log_2\, 2^{-l_i} \cdot f_Y(y_i).$$

An instantaneous encoding satisfies the Kraft inequality, so by substituting the Kraft inequality into the code rate formula,

$$r \geq -\sum_{i=1}^{k} log_2 \left(\frac{2^{-l_i}}{\sum_{j=1}^{k} 2^{-l_j}} \right) \cdot f_Y(y_i).$$

Let

$$p^*(l_i) = \left(\frac{2^{-l_i}}{\sum_{j=1}^{k} 2^{-l_j}} \right).$$

[49]Based on the methodology from the book by Gersho , A. and Gray, R. M. Vector Quantization And Signal Compression. Springer. 1992. Pages 267-269.

$$r = \sum_{i=1}^{k} l_i \cdot f_Y(y_i) \geq -\sum_{i=1}^{k} \left(log_2\left(p^*(l_i)\right)\right) \cdot f_Y(y_i).$$

When the set of output values $\{y_i\}_{i=1}^{k}$ are encoded with codewords of lengths l_i that satisfy the Kraft inequality, the resulting code rate is less than or equal to the code rate of encoding using a codewords of length l_i that do not satisfy the Kraft inequality.

The set codewords of lengths l_i that satisfy the Kraft inequality constitute a subset of all possible codewords. The inequality shown above holds for all codewords of length l_i, this implies that the assignment of codewords satisfying the above inequality is not unique and there are (possibly many) other codebooks comprising of codewords whose lengths satisfy the above inequality.

Note that $p^*(l_i)$ can be considered as a probability mass function, since

$$\sum_{i=1}^{k} \left(\frac{2^{-l_i}}{\sum_{j=1}^{k} 2^{-l_j}}\right) = 1 \text{ and } 0 < \left(\frac{2^{-l_i}}{\sum_{j=1}^{k} 2^{-l_j}}\right) \leq 1 \; \forall \; i.$$

What remains to be shown is that there exists a lower bound of the expression

$$-\sum_{i=1}^{k} \left(log_2\left(p^*(l_i)\right)\right) \cdot f_Y(y_i)$$

and that this lower bound maybe achieved if the codeword lengths satisfy a particular condition. The lower bound is the entropy of the random variable Y.

2.5.3 Determining the Probability Mass Function obtaining the Lower Bound

According to the book, [50] by considering the difference between the entropy of the random variable Y and the code rate of a codebook satisfying the Kraft inequality

$$-\sum_{i=1}^{k} \left(log_2\left(f_Y(y_i)\right)\right) \cdot f_Y(y_i) - \left(-\sum_{i=1}^{k} \left(log_2\left(p^*(l_i)\right)\right) \cdot f_Y(y_i)\right)$$

$$= \sum_{i=1}^{k} \left(log_2\left(\frac{1}{f_Y(y_i)}\right)\right) \cdot f_Y(y_i) - \sum_{i=1}^{k} \left(log_2\left(\frac{1}{p^*(l_i)}\right)\right) \cdot f_Y(y_i)$$

$$= \sum_{i=1}^{k} \left(log_2\left(\frac{p^*(l_i)}{f_Y(y_i)}\right)\right) \cdot f_Y(y_i).$$

The Taylor series expansion of $log_e(x)$ shows that $log_e(x) \leq x - 1$ for $x \in (0,1]$ with equality when $x = 1$ therefore,

[50] Gersho , A. and Gray, R. M. Vector Quantization And Signal Compression. Springer. 1992. Pages 267-269.

$$log_e\left(\frac{p^*(l)}{f_Y(y)}\right) \leq \left(\frac{p^*(l)}{f_Y(y)}\right) - 1.$$

Then by a changing the base of the logarithm

$$\frac{1}{log_e 2} \cdot \sum_{i=1}^{k} \left(log_e\left(\frac{p^*(l_i)}{f_Y(y_i)}\right)\right) \cdot f_Y(y_i)$$

and substituting the above expression into the logarithmic inequality we get

$$\frac{1}{log_e 2} \cdot \sum_{i=1}^{k} \left(log_e\left(\frac{p^*(l_i)}{f_Y(y_i)}\right)\right) \cdot f_Y(y_i) \leq \frac{1}{log_e 2} \cdot \sum_{i=1}^{k} \left(\left(\frac{p^*(l_i)}{f_Y(y_i)}\right) - 1\right) \cdot f_Y(y_i).$$

By multiplying into the brackets on the RHS of the above expression

$$\frac{1}{log_e 2} \cdot \sum_{i=1}^{k} \left(log_e\left(\frac{p^*(l_i)}{f_Y(y_i)}\right)\right) \cdot f_Y(y_i) \leq \frac{1}{log_e 2} \cdot \sum_{i=1}^{k} (p^*(l_i) - f_Y(y_i)).$$

Using the fact that both p^* and f_Y are both probability mass functions

$$\sum_{i=1}^{k} (p^*(y_i) - f_Y(y_i)) = \sum_{i=1}^{k} f_Y(y_i) - \sum_{i=1}^{k} p^*(y_i) = 0.$$

Thus we have

$$-\sum_{i=1}^{k} \left(log_2(f_Y(y_i))\right) \cdot f_Y(y_i) - \left(-\sum_{i=1}^{k} \left(log_2(p^*(l_i))\right) \cdot f_Y(y_i) \right) \leq 0$$

$$-\sum_{i=1}^{k} \left(log_2(f_Y(y_i))\right) \cdot f_Y(y_i) \leq -\sum_{i=1}^{k} \left(log_2(p^*(l_i))\right) \cdot f_Y(y_i).$$

The LHS is the entropy of the random variable Y and is the minimum code rate for encoding the output set of a random variable with a uniquely decipherable code, as shown above.

The equality holds if it is possible to encode using lengths l_i that match the probability mass function

$$f_Y(y_i) = p^*(l_i) = \left(\frac{2^{-l_i}}{\sum_{j=1}^{k} 2^{-l_j}}\right)$$

The rest of the chapter deals with the properties of entropy and code rate of blocks of random variables.

2.6 Entropy of Random Vectors of dimension m

In the previous chapter, the joint entropy of a random vector $\boldsymbol{Y}$ was defined as

$$H(\boldsymbol{Y};\boldsymbol{\theta},m)=-\sum_{y_m}\dots\sum_{y_3}\sum_{y_2}\sum_{y_1}\left(f_{\boldsymbol{Y}}\begin{pmatrix}y_1\\y_2\\\vdots\\y_m\end{pmatrix}\right)\cdot log_2\left(f_{\boldsymbol{Y}}\begin{pmatrix}y_1\\y_2\\\vdots\\y_m\end{pmatrix}\right)$$

With dimension m and a discrete joint probability mass function $f_{\boldsymbol{Y}}(\boldsymbol{y})$ where $\boldsymbol{Y}=\begin{pmatrix}Y_1\\Y_2\\\vdots\\Y_m\end{pmatrix}$ with finite output set Y_i, where Y_i is a discrete random variable with probability mass function $f_{Y_i}(y_i)$ with $y_i\in\{y_{i1},y_{i2},y_{i3},\dots,y_{ik_i}\}$ for $i=1,2,3,\dots,m$. The summation $\sum_{y_i}\ i=1,2,\dots,m$ is over the outcome set of the i^{th} element of $\boldsymbol{Y}$.

The following properties 2.7 – 2.10 and derivations of entropy are found in the book [51] and will re-derived for the purpose of understanding the concept of joint entropy and interpreting the results from the previous chapter.

2.7 Independent $Y_i's$

If the $Y_i's$ are independent, then

$$f_{\boldsymbol{Y}}\begin{pmatrix}y_1\\y_2\\\vdots\\y_m\end{pmatrix}=\prod_{i=1}^{m}f_{Y_i}(y)\text{ and }log_2\left(\prod_{i=1}^{m}f_{Y_i}(y)\right)=\sum_{i=1}^{m}log_2\left(f_{Y_i}(y_j)\right)$$

$$H(\boldsymbol{Y};\boldsymbol{\theta},m)=-\sum_{y_m}\dots\sum_{y_3}\sum_{y_2}\sum_{y_1}\left(\prod_{i=1}^{m}f_{Y_i}(y_j)\right)\cdot\left(\sum_{i=1}^{m}log_2\left(f_{Y_i}(y_j)\right)\right).$$

Consider the case where $m=2$

$$H(\boldsymbol{Y};\boldsymbol{\theta},2)=-\sum_{y_2}\sum_{y_1}\left(\prod_{i=1}^{2}f_{Y_i}(y_j)\right)\cdot\left(\sum_{i=1}^{2}log_2\left(f_{Y_i}(y_j)\right)\right)$$

$$=-\sum_{y_2}\sum_{y_1}\left(f_{Y_1}(y_1)\cdot f_{Y_2}(y_2)\right)\cdot\left(log_2\left(f_{Y_1}(y_1)\right)+log_2\left(f_{Y_2}(y_2)\right)\right)$$

$$=-\sum_{y_2}\sum_{y_1}\left(f_{Y_1}(y_1)\cdot f_{Y_2}(y_2)\right)\cdot log_2\left(f_{Y_1}(y_1)\right)-\sum_{y_2}\sum_{y_1}\left(f_{Y_1}(y_1)\cdot f_{Y_2}(y_2)\right)\cdot log_2\left(f_{Y_2}(y_2)\right)$$

$$=-\sum_{y_2}f_{Y_2}(y_2)\cdot\sum_{y_1}\left(f_{Y_1}(y_1)\right)\cdot log_2\left(f_{Y_1}(y_1)\right)-\sum_{y_1}f_{Y_1}(y_1)\cdot\sum_{y_2}\left(f_{Y_2}(y_2)\right)\cdot log_2\left(f_{Y_2}(y_2)\right)$$

[51] Cover, T. M. and Thomas, J. A. Elements of Information Theory. Wiley Series in Telecommunications and Signal Processing. Second Edition. Chapter Two. Pages 13-23.

$$= -\sum_{y_1}\left(f_{Y_1}(y_1)\right)\cdot log_2\left(f_{Y_1}(y_1)\right) - \sum_{y_2}\left(f_{Y_2}(y_2)\right)\cdot log_2\left(f_{Y_2}(y_2)\right)$$

$$= H(Y_1;\theta_1,1) + H(Y_2;\theta_2,1) = \sum_{i=1}^{2} H(Y_i;\theta_i,1).$$

Similarly for any m

$$H(\boldsymbol{Y};\boldsymbol{\theta},m) = \sum_{i=1}^{m} H(Y_i;\theta_i,1).$$

If the $Y_i's$ are independently identically distributed, then it follows from above that

$$H(\boldsymbol{Y};\boldsymbol{\theta},m) = m\cdot H(Y;\theta,1).$$

2.8 Conditionally Dependent $Y_i's$

Let $\boldsymbol{Y} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$ where Y_1 and Y_2 are not independent, then the conditional entropy is given by

$$H(Y_2|Y_1;\boldsymbol{\theta},2) = \sum_{y_1} f_{Y_1}(y_1)\cdot\left(-\sum_{y_2} f_{Y_2|Y_1}(y_2|y_1)\cdot log_2\left(f_{Y_2|Y_1}(y_2|y_1)\right)\right)$$

$$= -\sum_{y_1}\sum_{y_2} f_{Y_1}(y_1)\cdot f_{Y_2|Y_1}(y_2|y_1)\cdot log_2\left(f_{Y_2|Y_1}(y_2|y_1)\right)$$

$$= -\sum_{y_1}\sum_{y_2} f_{Y_1,Y_2}(y_1,y_2)\cdot log_2\left(f_{Y_2|Y_1}(y_2|y_1)\right)$$

$$H(Y_2|Y_1;\boldsymbol{\theta},2) = \mathbb{E}_{Y_1,Y_2}\left[log_2\left(f_{Y_2|Y_1}(y_2|y_1)\right)\right].$$

This can be extended to a vector $\boldsymbol{Y} = \begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_m \end{pmatrix}$ of dimension m

$$H(Y_m|Y_1,Y_2,Y_3,\dots,Y_{m-1};\boldsymbol{\theta},m)$$
$$= -\sum_{y_1}\sum_{y_2}\dots\sum_{y_m} f_{\boldsymbol{Y}}(y_1,y_2,\dots,y_m)\cdot log_2\left(f_{Y_m|Y_1,Y_2,\dots,Y_{m-1}}(y_m|y_1,y_2,\dots,y_{m-1})\right)$$

$$= \mathbb{E}_{Y_1,Y_2,\dots,Y_m}\left[log_2\left(f_{Y_m|Y_1,Y_2,\dots,Y_{m-1}}(y_m|y_1,y_2,\dots,y_{m-1})\right)\right].$$

2.9 The Chain Rule for Entropy

Consider the joint entropy for the case where $m = 2$

$$H(\boldsymbol{Y};\boldsymbol{\theta},2) = -\sum_{y_1}\sum_{y_2} f_{Y_1Y_2}(y_1,y_2)\cdot log_2\left(f_{Y_1Y_2}(y_1,y_2)\right)$$

$$= -\sum_{y_1}\sum_{y_2} f_{Y_1Y_2}(y_1,y_2)\cdot log_2\left(f_{Y_1}(y_1)\cdot f_{Y_2|Y_1}(y_2|y_1)\right)$$

$$= -\sum_{y_1}\sum_{y_2} f_{Y_1Y_2}(y_1,y_2)\cdot log_2\left(f_{Y_1}(y_1)\right) - \sum_{y_1}\sum_{y_2} f_{Y_1Y_2}(y_1,y_2)\cdot log_2\left(f_{Y_2|Y_1}(y_2|y_1)\right)$$

$$= -\sum_{y_1} f_{Y_1}(y_1)\cdot log_2\left(f_{Y_1}(y_1)\right) - \sum_{y_1}\sum_{y_2} f_{Y_1Y_2}(y_1,y_2)\cdot log_2\left(f_{Y_2|Y_1}(y_2|y_1)\right)$$

$$\therefore H(\boldsymbol{Y};\boldsymbol{\theta},2) = H(Y_1;\boldsymbol{\theta},1) + H(Y_2|Y_1;\boldsymbol{\theta},2).$$

This can be extended to a vector $\boldsymbol{Y} = \begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_m \end{pmatrix}$ of dimension m

$$H(\boldsymbol{Y};\boldsymbol{\theta},m) = \sum_{i=1}^{m} H(Y_i|Y_1,Y_2,\dots,Y_{i-1};\boldsymbol{\theta},m).$$

2.10 Mutual Information and Information Inequality

Mutual information is the measure of the amount of information that one random variable contains about another random variable. It represents the reduction of uncertainty of one random variable due to knowledge of the other variable and is given by $I(Y_1,Y_2)$ [52].

2.10.1 Mutual Information

Consider the random vector $\boldsymbol{Y} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$ of dimension $m = 2$

$$I(Y_1,Y_2) = \sum_{y_1}\sum_{y_2} f_{Y_1,Y_2}(y_1,y_2)\cdot log_2\left(\frac{f_{Y_1,Y_2}(y_1,y_2)}{f_{Y_1}(y_1)\cdot f_{Y_2}(y_2)}\right)$$

$$= \sum_{y_1}\sum_{y_2} f_{Y_1,Y_2}(y_1,y_2)\cdot log_2\left(\frac{f_{Y_2|Y_1}(y_2|y_1)\cdot f_{Y_1}(y_1)}{f_{Y_1}(y_1)\cdot f_{Y_2}(y_2)}\right)$$

$$= -\sum_{y_1}\sum_{y_2} f_{Y_1,Y_2}(y_1,y_2)\cdot log_2\left(f_{Y_2}(y_2)\right) + \sum_{y_1}\sum_{y_2} f_{Y_1,Y_2}(y_1,y_2)\cdot log_2\left(f_{Y_2|Y_1}(y_2|y_1)\right)$$

[52] Cover, T. M. and Thomas, J. A. Elements of Information Theory. Second Edition. Wiley Series in Telecommunications and Signal Processing. Second Edition. 2002. Page 19.

$$= -\sum_{y_2} f_{Y_2}(y_2) \cdot log_2\left(f_{Y_2}(y_2)\right) + \sum_{y_1}\sum_{y_2} f_{Y_1Y_2}(y_1, y_2) \cdot log_2\left(f_{Y_2|Y_1}(y_2|y_1)\right)$$

$$= -\mathbb{E}_{Y_2}\left[log_2\left(f_{Y_2}(y_2)\right)\right] + \mathbb{E}_{Y_1,Y_2}\left[log_2\left(f_{Y_2|Y_1}(y_2|y_1)\right)\right]$$

$$\therefore I(Y_1, Y_2) = H(Y_2; \boldsymbol{\theta}, 1) - H(Y_2|Y_1; \boldsymbol{\theta}, 2).$$

It [53] can be shown that

$$I(Y_1, Y_2) = H(Y_2; \boldsymbol{\theta}, 1) - H(Y_2 - Y_1|Y_1; \boldsymbol{\theta}, 2).$$

2.10.2 Mutual Information Inequality

Consider the negative of the mutual information

$$-I(Y_1, Y_2) = -\sum_{y_1}\sum_{y_2} f_{Y_1,Y_2}(y_1, y_2) \cdot log_2\left(\frac{f_{Y_1,Y_2}(y_1, y_2)}{f_{Y_1}(y_1) \cdot f_{Y_2}(y_2)}\right)$$

$$= \sum_{y_1}\sum_{y_2} f_{Y_1,Y_2}(y_1, y_2) \cdot log_2\left(\frac{f_{Y_1}(y_1) \cdot f_{Y_2}(y_2)}{f_{Y_1,Y_2}(y_1, y_2)}\right)$$

$$\leq log_2\left(\sum_{y_1}\sum_{y_2} f_{Y_1,Y_2}(y_1, y_2) \cdot \left(\frac{f_{Y_1}(y_1) \cdot f_{Y_2}(y_2)}{f_{Y_1,Y_2}(y_1, y_2)}\right)\right).$$

Since log_2 is (strictly) concave using Jensen's Inequality [54].

$$-I(Y_1, Y_2) \leq log_2\left(\sum_{\forall Y_1}\sum_{\forall Y_2} f_{Y_1}(y_1) \cdot f_{Y_2}(y_2)\right)$$

$$= log_2\left(\sum_{\forall Y_1} f_{Y_1}(y_1) \cdot \sum_{\forall Y_2} f_{Y_2}(y_2)\right)$$

$$= 0.$$

Thus

$$I(Y_1, Y_2) \geq 0$$

$$0 \leq I(Y_1, Y_2) = H(Y_2; \boldsymbol{\theta}, 1) - H(Y_2|Y_1; \boldsymbol{\theta}, 2)$$

$$H(Y_2; \boldsymbol{\theta}, 1) \geq H(Y_2|Y_1; \boldsymbol{\theta}, 2).$$

That is if Y_1 contains information about Y_2, then the uncertainty related to Y_2 is reduced.

[53] Wiegand, T. Rate Distortion Theory & Quantization. Digital Image Communication. Page 14.
[54] www.encyclopediaofmath.org/index.php/Jensen_inequality.

2.11 Blocking Random Variables

In the previous chapter, the concept of a block of random variables of size m was introduced, and was defined as a random vector of dimension m composed of random variables that are grouped together.

The reason for blocking random variables will become apparent in the next two sections (section 2.11.1 and 2.11.2) that have been adapted from those given in [55].

2.11.1 Blocking Reduces Entropy

Consider the joint entropy of $\boldsymbol{Y}$ a random vector of dimension m, according to the mutual information inequality

$$\begin{aligned} H(\boldsymbol{Y};\boldsymbol{\theta},m) &= H(Y_1,Y_2,\dots,Y_{m-1};\boldsymbol{\theta},m-1) + H(Y_m|Y_1,Y_2,\dots,Y_{m-1};\boldsymbol{\theta},m) \\ &\leq H(Y_1,Y_2,\dots,Y_{m-1};\boldsymbol{\theta},m-1) + H(Y_m;\theta,1) \\ &\leq H(Y_1,Y_2,\dots,Y_{m-2};\boldsymbol{\theta},m-2) + H(Y_m;\theta,1) + H(Y_{m-1};\theta,1) \\ &\quad \dots \\ &\leq \sum_{i=1}^{m} H(Y_i;\theta,1). \end{aligned}$$

Blocking random variables together reduces the entropy below that of the individual random variable constituents, equality holds if the m random variables in the block are independent. This confirms what was shown in the previous chapter with Bernoulli random variables. This result can be argued from a geometric perspective as well.

2.11.2 Blocking Reduces Code Rate

In the previous chapter a reduction in code rate was even noticeable for the case of independently distributed random variables. This reduction was due to the convergence of the code rate to the entropy lower bound as block size increases. This aspect will be shown next. The authors show that an increase in block size reduces the code rate, the derivation based on the book will be given below.

Consider the probability density function $f_Y(y)$ for a particular random variable Y, if it is possible to encode each output value y_i with a codeword of length l_i for $i = 1, 2, \dots, k$ so that it satisfies

$$f_Y(y_i) = \frac{2^{-l_i}}{\sum_{j=1}^{k} 2^{-l_j}}$$

then the code rate will be equal to the entropy.

If there is no codeword of integer length l_i that allows for the encoding as shown in the above expression, then the encoding will not be optimal, that is, attaining entropy.

[55] Gersho, A. and Gray, R. M. Vector Quantization and Signal Compression. Springer. 1992. Pages 276-277.

In this case, codewords with integer valued lengths l_i can be assigned to the probabilities $f_Y(y_i)$ such that

$$-log_2(f_Y(y_i)) \leq l_i < -log_2(f_Y(y_i)) + 1.$$

This satisfies the Kraft inequality, since the LHS can be written as

$$\sum_{i=1}^{k} 2^{-l_i} \leq \sum_{i=1}^{k} 2^{log_2(f_Y(y_i))} = \sum_{i=1}^{k} f_Y(y_i) = 1.$$

Also by taking the expected values, we have

$$-\sum_{i=1}^{k} f_Y(y_i) \cdot log_2(f_Y(y_i)) \leq \sum_{i=1}^{k} (f_Y(y_i) \cdot l_i) < -\sum_{i=1}^{k} f_Y(y_i) \cdot log_2(f_Y(y_i)) + 1.$$

Which is

$$H(Y;\theta,1) \leq \mathbb{E}(l) < H(Y;\theta,1) + 1.$$

This shows that there is an instantaneous encoding that is within one bit of the entropy code rate. The increased efficiency of blocking can now be established, since the previous expression also holds for a random vector $\boldsymbol{Y} = \begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_m \end{pmatrix}$ of dimension m, with a discrete joint probability mass function $f_{\boldsymbol{Y}}(\boldsymbol{y})$ where Y_i has a probability mass function $f_{Y_i}(y_i)$ for $y_i \in \{y_{i1}, y_{i2}, y_{i3}, \dots, y_{ik_i}\}, i = 1, 2, 3, \dots, m.$

There are $k^{[m]} = \prod_{i=1}^{m} k_i$ vector permutations possible.

In the same way as before, by assigning code words of length l_i to the probabilities $f_{\boldsymbol{Y}}(\boldsymbol{y}_i)$ such that

$$-log_2\left(f_{\boldsymbol{Y}}(\boldsymbol{y}_j)\right) \leq l_i < -log_2\left(f_{\boldsymbol{Y}}(\boldsymbol{y}_j)\right) + 1.$$

Then by taking the expected values over the joint distribution $f_{\boldsymbol{Y}}(\boldsymbol{y})$ we get

$$H(\boldsymbol{Y};\boldsymbol{\theta},m) \leq \mathbb{E}(l^B) < H(\boldsymbol{Y};\boldsymbol{\theta},m) + 1.$$

Where $\mathbb{E}(l^B)$ is the expected block length, this differs from $\mathbb{E}(l)$, the element-wise code rate, since $k^{[m]}$ blocks of length m are represented by the codebook, as explained in the previous chapter, the element-wise code rate is calculated as

$$\mathbb{E}(l) = \frac{1}{m} \cdot \sum_{j=1}^{k^{[m]}} l\left(\mathfrak{C}(\boldsymbol{y}_j)\right) \cdot f_{\boldsymbol{Y}}(\boldsymbol{y}_j).$$

The expected length per output value (or element-wise bit rate) is $\mathbb{E}(l) = \frac{1}{m} \cdot \mathbb{E}(l^B)$ and

$$\frac{1}{m} \cdot H(\boldsymbol{Y}; \boldsymbol{\theta}, m) \leq \frac{\mathbb{E}(l^B)}{m} < \frac{1}{m} \cdot H(\boldsymbol{Y}; \boldsymbol{\theta}, m) + \frac{1}{m}.$$

By increasing the block size m, code rates converge towards the joint entropy code rate of the random vector $\boldsymbol{Y}$ of size m.

Combining the expression given above with the inequality obtained in section 2.11.1, namely

$$H(\boldsymbol{Y}; \boldsymbol{\theta}, m) \leq \sum_{i=1}^{m} H(Y_i; \theta, 1).$$

Then for a large m the element-wise bit rate can be reduced below the individual entropy values calculated for each random variable Y_i averaged over the block size m

$$\frac{\mathbb{E}(l^B)}{m} < \frac{1}{m} \cdot \sum_{i=1}^{m} H(Y_i; \theta, 1).$$

2.12 Alternative Entropy Derivation [56]

Consider a vector $\boldsymbol{Y} = \begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_m \end{pmatrix}$ of size m where each Y_i is independently identically $Unif\left(\frac{1}{k}\right)$ distributed. The number of bits required to uniquely encode every output variable Y_i is given by $log_2 k$ and to encode anyone of the k^m possible strings will require $log_2(k^m) = m \cdot log_2 k$ bits.

If however, each Y_i is non-uniform then, as seen in the previous chapter, it is possible to encode more probable outcomes with shorter codewords and less probable outcomes with longer codewords.

So if Y_i is non-uniform but still independently identically distributed [57] with k outcome values $y_{i,j}$ with $j = 1, 2, 3, \dots, k$ where $P(Y_i = y_{i,j}) = p_j$ for every $i = 1, 2, 3, \dots, m$ then there are k^m possible strings, each with a probability $\prod_{j=1}^{k} (p_j)^{n_j}$ where n_j is the number of occurrences of the outcome value $y_{i,j}$.

This can be considered in the context of a multinomial distribution given by

$$f_Y(\boldsymbol{y}, m, p_1, p_2, p_3, \dots, p_K) = \begin{cases} \left(\frac{m!}{n_1! n_2! \dots n_m!}\right) {p_1}^{n_1} {p_2}^{n_2} {p_3}^{n_3} \dots {p_k}^{n_k} & when \sum_{i=1}^{k} n_i = m \\ 0 & otherwise \end{cases}.$$

[56] Proof found in text by Rozanov,Y.A. Probability Theory: A Concise Course. Dover. Pages 115-116.
[57] This is not a necessary prerequisite for the entropy calculation, it is however a necessary consideration in this proof.

The random vector $\boldsymbol{Y}$ can take on any one of m^k possible outcomes each with probability given by the above expression.

If m is very large, then by the strong law of large numbers [58],

$$\lim_{m\to\infty}\left(\frac{1}{m}\cdot\sum_{i=1}^{m}n_i\right)=p_i$$

If m is large there are $\left(\frac{m!}{n_1!n_2!n_3!\ldots.n_k!}\right)$ overwhelmingly likely outcomes each with the probability ${p_1}^{n_1}{p_2}^{n_2}{p_3}^{n_3}\ldots{p_k}^{n_k}$, where $n_j=m\cdot p_j$ and $m=\sum_{j=1}^{k}n_j$ [59].

In other words, if the vector $\boldsymbol{Y}$ has a very large dimension m then most of the information would be accounted for by considering a subset of size $\left(\frac{m!}{n_1!n_2!n_3!\ldots.n_k!}\right)$ possible output vectors. Only a little information would be unaccounted for if the rest of the output vectors are ignored.

If only this subset is considered, then this corresponds once again to a uniform distribution with $\left(\frac{m!}{n_1!n_2!n_3!\ldots.n_k!}\right)$ outcomes each with the probability $\left(\frac{n_1!n_2!n_3!\ldots.n_k!}{m!}\right)$ of occurring.

In order to calculate the element-wise bit rate, we divide through by the string length m, so the element-wise bit rate is $\frac{1}{m}\cdot log_2\left(\frac{m!}{n_1!n_2!n_3!\ldots.n_k!}\right)$ bits to encode each output value.

By replacing all factorials by Stirling's formula [60]

$$n!\simeq\sqrt[2]{2\cdot\pi\cdot n}\cdot n^n\cdot e^{-n}.$$

We get

$$\frac{1}{m}\cdot log_2\left(\frac{m!}{n_1!\,n_2!\,n_3!\ldots.n_k!}\right)$$

$$=\frac{1}{2m}\cdot log_2(2\pi)-\frac{k}{2m}\cdot log_2(2\pi)+log_2(m)-\frac{1}{m}\cdot\sum_{i=1}^{k}n_i\cdot log_2(n_i).$$

And by replacing n_i with $m\cdot p_i$

$$=\left(\frac{1-k}{2m}\right)\cdot log_2(2\pi)-\sum_{i=1}^{k}p_i\cdot log_2(p_i).$$

[58] Rozanov,Y.A. Probability Theory: A Concise Course. Dover. Page 115.
[59] Only rational probability values are considered.
[60] Stirling's Formula, Y.A. Rozanov, Probability Theory: A Concise Course. Page 10.

Therefore, the element-wise bit rate of encoding a random variable Y with k outcome values with m large is given by the entropy of Y

$$H(Y;\boldsymbol{\theta},1) = -\sum_{i=1}^{k} p_i \cdot log_2(p_i).$$

2.13 Conclusions of Chapter Two

- Entropy is a measure of the amount of information required on the average to describe a random variable.
- The uncertainty of a random variable is defined as the number of bits required to encode any outcome of an experiment.
- The entropy $H(Y;\theta,1)$ of a discrete random variable Y with probability mass function f_Y is $-\sum_{i=1}^{k} f_Y(y_i) \cdot log_2\big(f_Y(y_i)\big)$.
- The Kraft Inequality provides a bound on the lengths of uniquely decipherable variable rate encoding schemes, and therefore instantaneous codes.
- Blocks of random variables, can achieve lower rates than individual random variables when encoded with uniquely decipherable variable rate encoding, even if the random variables are independent.
- If a dependency structure exists amongst the random variables constituting a block, then the block entropy rate is lower than the sum of the individual entropy rates.
- If a dependency structure exists amongst the random variables constituting a block, then the code rates achievable, may be below that of the individual entropy rates averaged over the block size m.

Chapter Three:
SCALAR QUANTIZATION

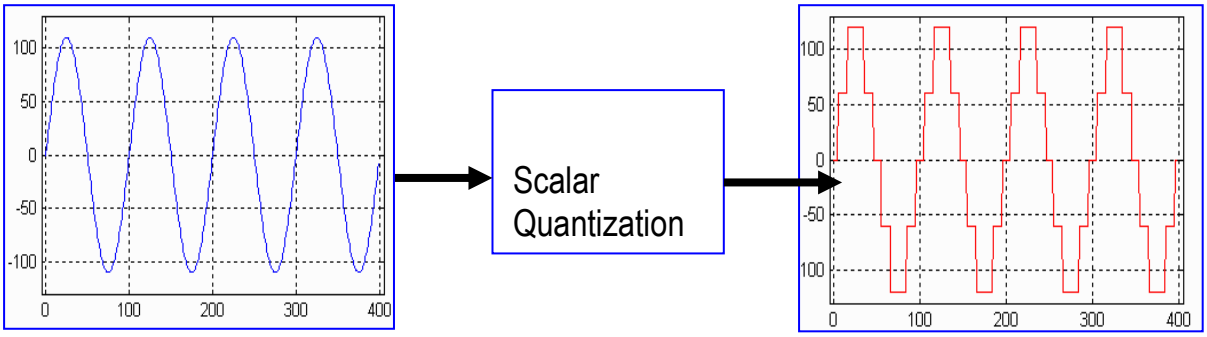


Figure 3.1 Scalar Quantization Applied to a Continuously Sampled Signal (left block) with quantized output (right block). [61]

In the previous chapter it was shown that lossless data compression is bounded below by the joint entropy which was calculated from the joint probability mass function of the random vector encoded. Lossy data compression, is able to reduce the code rate below that achieved by lossless data compression by means of data quantization, this however at the cost of approximation error.

3.1 Quantization in Data Compression

Data quantization is a process which is applied to quantitative continuous/discrete data that reduces the data size to a pre-set value by means of approximation, as described in chapter one as lossy data compression.

Data quantization achieves increased data compression by discarding the information content, in order that the quantized data may be encoded at a lower rate, thereby reducing the storage capacity of the encoded data. The reduction in information content is due to a combination of the restructuring of the random variable's probability mass/density function and the decrease in its output set size through quantization.

3.2 Scalar and Vector Quantization

In general, quantization is applied to scalar or vector data sampled from a discrete or continuous random variable or vector, which results in discrete, countable random outcome values or vectors. In this chapter, the focus will be on the quantization of a continuous random variable X (scalar) with a density function f_X. A value x from the support of f_X is quantized by replacing it with a suitable reproduction point y. The value y approximates x.

[61] Kolesnikov, A. Image Compression, Lecture 8, Scalar Quantization. Department of Computer Science, University of Joensuu, Joensuu, Finland.

The quantization of X results in a discrete random variable Y with a discrete probability mass function f_Y and a reproduction point set $\{y_i\}_{i=1}^{k}$.

3.3 Quantizer Design

The quantizer design partitions the support of a random variable and allocates all values within a partition to a specific value known as a reproduction point.

The quantizer design is usually done prior to the process of quantization, although there are many applications where the quantizer is designed "on-the-run", as the data is analysed the quantizer evolves in tandem with the data feed.

The number of reproduction points k and a performance measure that penalizes the dissimilarity between the input value x and output reproduction point y are often chosen before the quantizer is designed. These performance measures are often called distortion measures in the context of data quantization.

Three scalar quantizer designs will be discussed in this chapter namely: a uniform scalar quantizer, a companded scalar quantizer and a Lloyd Max locally optimal quantizer.

3.4 Scalar Quantization

Three important aspects of a scalar quantizer are: the reproduction set, the boundary value set and the quantizer rule Q.

3.4.1 Partitioning the Support of X and the Boundary Values

Consider the continuous random variable X with probability density function f_X with support on the interval $I \subseteq \mathbb{R}$. Let I_j be a sub-interval of I with $I = \bigcup_{j=1}^{k} I_j$ and $I_s \cap I_t = \emptyset$ for all $s \neq t$. If the sub-interval I_j is bounded, then it is known as a granular sub-interval, if it is not, then is known as an overloaded sub-interval.

The sub-interval is defined as open, half-open or closed by the inclusion or exclusion of its boundary values.

Open interval $I_j = (a_{j-1}, a_j)$.

Half- open interval $I_j = [a_{j-1}, a_j), I_j = (a_{j-1}, a_j]$.

Closed interval $I_j = [a_{j-1}, a_j]$.

If every sub-interval I_j is closed or half- open and the reproduction point $y_j \in I_j$ then the scalar quantizer is called regular [62].

[62] Gersho, A. and Gray, R. M. Vector Quantization and Signal Compression. Springer. 1992. Page 135.

3.4.2 Scalar Quantizer Q

The scalar quantizer is the mapping Q of the interval I onto the reproduction set R where $R = \{y_i\}_{i=1}^{k}$. The mapping Q is a composite function made up of an index encoder and an index decoder [63].

Mapping the random variable X onto an index i is achieved by the index encoder $E_I(\cdot)$: $\mathbb{R} \rightarrow \mathbb{N}$, while the mapping of the index i onto the reproduction point set $R = \{y_i\}_{i=1}^{k}$ is achieved by the index decoder $D_I(\cdot)$: $\mathbb{N} \rightarrow R$ [64].

So for a continuous random variable X, if $x \in I_j$ then $Q(x) = y_j$, since $E_I(x) = j$ and $D_I(j) = y_j$.

3.4.3 Probability Mass Function of the Reproduction Set

For the quantized random variable $Y = Q(X)$ for the random variable X with probability density function f_X, the reproduction set $R = \{y_i\}_{i=1}^{k}$ has a discrete probability mass function

$$f_Y(y_j) = \begin{cases} \int_{I_j} dF_X & if\ x \in I_j \\ 0 & if\ x \notin I_j \end{cases}.$$

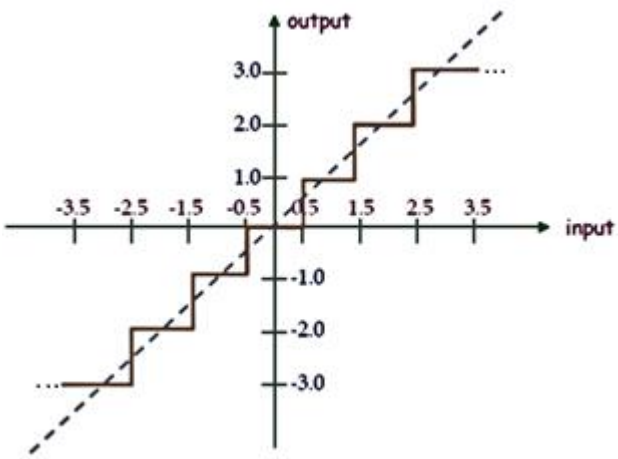


Figure 3.2 Regular Scalar Quantizer The random variable X is plotted against an regular scalar quantizer $Y = Q(X)$. Under-predictions occur on the sub-intervals $\{(j, j + 0.5)\}_{j\in\mathbb{Z}}$, exact predictions at the set of points $\{j\}_{j\in\mathbb{Z}}$ and over predictions occur on the set of sub-intervals$\{(j + 0.5, j + 1)\}_{j\in\mathbb{Z}}$. The reproduction point set $R = \{y_j\}_{j\in\mathbb{Z}}$ is the set of points $\{j\}_{j\in\mathbb{Z}}$, while the boundary values $\{a_j\}_{j\in\mathbb{Z}}$ are the set of points $\{j + 0.5\}_{j\in\mathbb{Z}}$.

[63] It is important to notice that there is a difference between and index encoder and the encoder referred to in chapter one and two.
[64] Quantization as a composite mapping will be discussed in Chapter Four in the more general case of vector quantization.

3.5 Distortion Measures of Scalar Quantization

A scalar distortion measure is the degree of dissimilarity or the approximation error between the input random variable X and output quantized random variable Y.

The distortion measure used is dependent on the application, the type of data and the input data distribution.

3.5.1 Distortion Measure

A distortion measure $d: X \times X \to \mathbb{R}^+ \cup \{0\}$ and $x, y, z \in X$

is a metric if

$$d(x,y) \geq 0 \qquad (positivity)$$

$$d(x,y) = 0 \Leftrightarrow x = y \qquad (identity)$$

$$d(x,y) = d(y,x) \qquad (symmetry)$$

$$d(x,z) \leq d(x,y) + d(y,z) \qquad (triangle\ inequality)$$

3.5.2 Two Commonly Used Distortion Measures [65]

Consider the random variable X with probability function f_X and the scalar quantizer Q that approximates X with $Q(X)$. Then $Q(x)$ is the quantized value of x and the distortion due to x being approximated by $Q(x)$ is given by $d\big(x, Q(x)\big)$.

Squared Error Distortion (also called noise energy)

The squared error distortion is the most commonly used distortion and is given by

$$d\big(x, Q(x)\big) = \big(x - Q(x)\big)^2$$

Its popularity is due to the wide knowledge of mean squared error theory. It is also the square of the Euclidian distance between x and $Q(x)$ as used in neutral geometry.

Hamming Distortion (also called 0/1 loss)

The Hamming distortion is used if X is a discrete random variable, and is given by

$$d\big(x, Q(x)\big) = \begin{cases} 0\ if\ x = Q(x) \\ 1\ if\ x \neq Q(x) \end{cases}$$

[65] See Deza, E. Deza, M. Dictionary of Distances. Elsevier Science & Technology Books. 2006. for more distance measures.

3.5.3 Entropy Constrained Distortion

The entropy constrained distortion measure which is the linear combination of a distortion $d(x, Q(x))$ and the uncertainty $-log_2(f_Y(y))$ where $Q(x) = y$ is

$$d^*(x, Q(x)) = d(x, Q(x)) + \lambda \cdot (-log_2(f_Y(y))).$$

Where λ is a weighting or penalty factor. As the distortion decreases, a related increase in information content will result, and hence a decrease in compression efficiency.

Minimization of $d^*(x, Q(x))$ over all $x \in I$ leads to the constrained problem relating to the rate distortion function which will be briefly discussed in the next chapter.

3.5.4 Combined Distortion

The combined distortion is a function of the distortion measure computed over the support of the input random variable X. The most commonly used combined distortion is the mean or expected distortion.

3.5.5 Mean Distortion

Let X be a continuous random variable with probability density function $f_X(x)$, the mean distortion is given by

$$D = \mathbb{E}[d(X, Q(X))] = \int_{-\infty}^{\infty} d(x, Q(x)) \cdot f_X(x)dx.$$

Or for the sub-intervals $I_j, j = 1, 2, \dots, k$

$$D = \mathbb{E}[d(X, Q(X))] = \sum_{i=1}^{k} \int_{a_{i-1}}^{a_i} d(x, y_i) \cdot f_X(x)dx.$$

The mean distortion may be divided into granular and overloaded regions

$$D = \int_{-\infty}^{a} d(x, Q(x)) \cdot f(x)dx + \sum_{j=1}^{k} \int_{a_{j-1}}^{a_j} d(x, Q(x)) \cdot f(x)dx + \int_{b}^{\infty} d(x, Q(x)) \cdot f(x)dx.$$

Then since both $d(x, Q(x)) \geq 0$ and $f(x) \geq 0 \; \forall \; x \in \mathbb{R}$

$$D \geq \sum_{j=1}^{k} \int_{a_{j-1}}^{a_j} d(x, Q(x)) \cdot f(x)dx.$$

If the sub-interval $[a; b]$ is chosen so that $\int_{-\infty}^{a} f(x)dx + \int_{b}^{\infty} f(x)dx \approx 0,$

then distortion will be approximately equal to $\sum_{j=1}^{k} \int_{a_{j-1}}^{a_j} d(x, Q(x)) \cdot f(x)dx$ on average.

3.5.6 Sample Mean Distortion

For a set of observations $\{x_j\}_{j=1}^{N}$ quantized according to the set of reproduction points $R = \{y_i\}_{i=1}^{k}$ the mean distortion D can be estimated by

$$\overline{D} = \sum_{i=1}^{k}\sum_{j=1}^{N} S_i(x_j) \cdot d(x_j, y_i) \cdot f_X(x_j)$$

where $S_i(x_j) = \begin{cases} 1 \; if \; x_j \in (a_{i-1}, a_i) \\ 0 \; if \; x_j \notin (a_{i-1}, a_i) \end{cases}$.

This is an estimate of the combined mean distortion.

3.6 Three Methods of Scalar Quantization

Three scalar quantization methods will be considered: uniform scalar quantization, companded scalar quantization and Lloyd Max scalar quantization.

3.6.1 Uniform Scalar Quantization

Uniform scalar quantization is a scalar quantization technique that uses equal length sub-intervals to partition the support I of X.

The step-size between two adjacent reproduction points is Δ .

- $\Delta = y_i - y_{i-1}$ for $i = 2, 3 \ldots, k$.
- $y_i = \frac{(a_{i-1}+a_i)}{2}$ for $i = 1, 2, 3 \ldots, k$.

If the support of X is the bounded interval $I = (a, b)$ then $\Delta = \frac{b-a}{k}$ the input random variable has a bounded sample space with maximum error $\frac{\Delta}{2}$.

If the support is the unbounded interval $I \subseteq \mathbb{R}$, then the quantizer design for overloaded intervals $(-\infty; a_0]$ and/or $[a_k; \infty)$ is often done independently of the bounded interval $[a_0; a_k]$. This is due to large errors that may occur in quantizing observations from the unbounded region.

Consider a random variable X with density function f_X, the uniform scalar quantizer has the following properties

- The uniform scalar quantizer is regular.
- The boundary values of a uniform scalar quantizer are equally spaced.
- The reproduction points are the midpoints of the two adjacent boundary values.

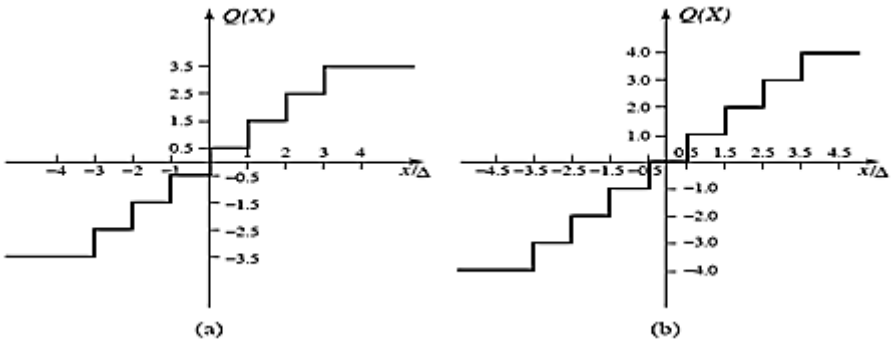


Figure 3.3[66] Midrise and Midtread Scalar Quantizer (a) is known as midrise uniform scalar quantization, while (b) is known as midtread uniform scalar quantization. The midtread uniform scalar quantizer shown above is equivalent to rounding to the nearest integer and maps the zero value onto zero, while the midrise uniform quantizer does not map onto the zero value.

Uniform scalar quantization is robust, it is often simpler than its non-uniform counterparts and near-optimal for a large reproduction value sets [67].

Uniform scalar quantization will be considered for uniform and non-uniform probability functions in the next section.

3.6.1.1 Uniform Quantization Applied to a Uniform Distribution

If X follows a $Uniform(a, b)$, then for k reproduction points, the step size is given by

$$\Delta = \frac{b-a}{k}.$$

The reproduction points y_i and the boundary values a_i can be calculated as

$y_i = y_{i-1} + \Delta$ and $a_i = a + i \cdot \Delta$ for $i = 2, 3 \ldots, k$.

The quantization error $\epsilon = \big(X - Q(X)\big)$ is $Unif\left(-\frac{\Delta}{2}, \frac{\Delta}{2}\right)$ then

$\mathbb{E}(\epsilon) = 0$ and $\mathbb{E}(\epsilon^2) = \frac{\Delta^2}{12}$ and $\mathbb{E}(X \cdot \epsilon) = -\frac{\Delta^2}{12}$.

3.6.1.2 Uniform Quantization Applied to a Non-Uniform Distribution

If the continuous random variable X is not uniform and has a support that is not bounded in $\mathbb{R}$, then uniform scalar quantization can be applied to a bounded sub-interval $[a; b]$ of I and the overloaded sub-interval/s can be dealt with independently by either discarding the observations that occur outside of the interval, or by quantizing them with a larger step-size.

[66] Image obtained from PDF Cleveland State University CIS 658 Fall 2005.
[67] Gersho, A. and Gray, R. M. Vector Quantization and Signal Compression. Springer. 1992. Pages 151, 153 and 299.

3.6.2 Non-Uniform Scalar Quantization

The difference between non-uniform and uniform scalar quantization is that the step-size (sub-interval lengths) between the reproduction points for a non-uniform quantizer is not constant.

Two major advantages of non-uniform scalar quantization over uniform scalar quantization are [68].

1. For a given reproduction point set size k, one can increase the range that can be accommodated by a reproduction point set from a source with an overloaded region.
2. For a specific input probability density function, a non-uniform quantizer can be developed to match the probability density function, which will result in lower combined distortion levels.

One such non-uniform scalar quantization technique is companding.

3.6.3 Companding

Companding applies a non-linear transformation to a random variable, and then applies uniform scalar quantization to the transformed random variable. The inverse transform is then applied to the quantized random variable. The companding process is illustrated in figure 3.4.

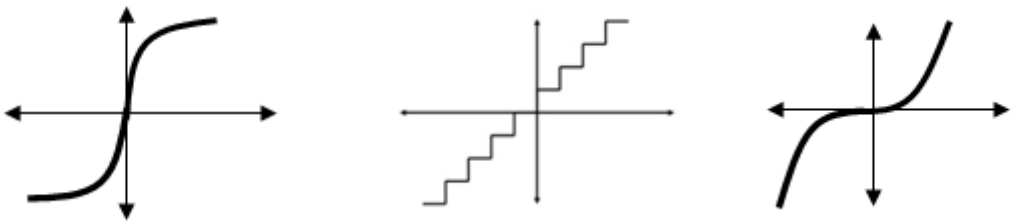


Figure 3.4: Compressor (Far Left), Uniform Quantizer (Middle), Expander (Far Right)

The input random variable X is transformed by an invertible non-linear function E into a bounded random variable $\tilde{X}$. $\tilde{X}$ is then quantized by uniform scalar quantization. The resultant quantized random variable $\tilde{Y}$ is then transformed by the inverse of the non-linear function E^{-1} the expander.

A signal to be transmitted that requires compression by means of companding prior to transmission is first compressed by the function E, it is then uniformly scalar quantized and encoded before transmitting it. At the receiver, the compressed signal is decoded and then expanded by the function E^{-1}.

[68] Gersho, A. and Gray, R. M. Vector Quantization and Signal Compression. Springer. 1992. Page 156.

The effect of companding can be interpreted as using a tightly packed reproduction set on the sub-intervals of high probability, while using a sparse and loosely packed reproduction point set on intervals of lower probability, this is due to the shape of the compressor curve.

Three companding methods will be considered: Probability Integral Transform compandor, piece-wise compandor and logarithmic compandor.

3.6.3.1 Probability Integral Transform Compandor

Consider the distribution function F_X , we know that the distribution function compresses X onto the unit interval $[0,1]$. That is $E = F_X(X)$ then $E(X) \sim Unif(0,1)$.

If F_X is one-to-one, the probability integral transform provides a compressor E and expandor E^{-1} where E maps the support of a random variable X onto the unit interval $[0,1]$ and E^{-1}maps the interval $[0,1]$ onto the reproduction point set R.

If F is not one-to-one, the probability integral transform still holds, however the use of E as a compressor is problematic, as E^{-1} may not be unique. One way of solving this is by approximating F by a smooth curve such that $\frac{dF}{dx} > 0$ for $x \in I$.

If X is a continuous random variable with a distribution function F_X with an unbounded support, with either the minimum, maximum or both boundary points undefined, then for a reproduction point set of size k, the boundary points $\{a_i\}_{i=1}^{k-1}$ can be determined as follows

$$\int_{-\infty}^{a_i} f_X(x)dx = \frac{i}{k} \text{ where } i = 1, 2, \dots, k-1$$

with $a_0 = a$ or $a_k = b$ where applicable, that is a_i is the $\left(\frac{i}{k}\right)^{th}$ percentile of the distribution function F_X.

The reproduction points $\{y_i\}_{i=1}^{k}$ are determined by

$$\int_{-\infty}^{y_i} f_X(x)dx = \frac{i-1}{k} + \frac{1}{2 \cdot k} \text{ where } i = 1,2, \dots, k.$$

If X has a bounded support $I = [a, b]$, then the boundary points can be obtained by substituting $a = a_0$ and $b = a_k$ into the above formula.

If the probability density function of X is either not known or the integration intractable, then the empirical cumulative distribution function $\hat{F}$ can be used, where $\hat{F}$ is given by

$$\hat{F}(b) = \frac{1}{M}\sum_{j=1}^{M} \phi(b - x_j)$$

$$\phi(x) = \begin{cases} 1 \; if \; x \geq 0 \\ 0 \; if \; x < 0. \end{cases}$$

The cumulative probabilities can be divided into k sub-intervals, with step size $= \frac{1}{k}$, with the number of observations M such that $k << M$ and the empirical cumulative distribution function can be used to design the non-uniform quantizer.

Let a_i be a boundary point where $\hat{F}(a_i) = \frac{i}{k}$ for $i = 1,2, \dots, k-1$ and let y_i be a reproduction point where $\hat{F}(y_i) = \frac{i-1}{k} + \frac{1}{2 \cdot k}$ where $i = 1,2, \dots, k$.

In order to estimate a_0, the first order statistic can be used $a_0 = \hat{x}_{[1]}$ and similarly for the estimation of a_N, the last order statistic $a_N = \hat{x}_{[M]}$ may be used, where $\hat{x}_{[i]}$ is the estimated i^{th} order statistic.

The probability integral transform is a conceptual application to the non-uniform compression problem and can be implemented in many situations, however, since most data signals are generally non-stationary and have a varying dynamic range. An easier solution would be to find an invertible function independent of the data signal that is defined for the whole real line. One such function is the logarithmic compressor.

3.6.3.2 Logarithmic Companding

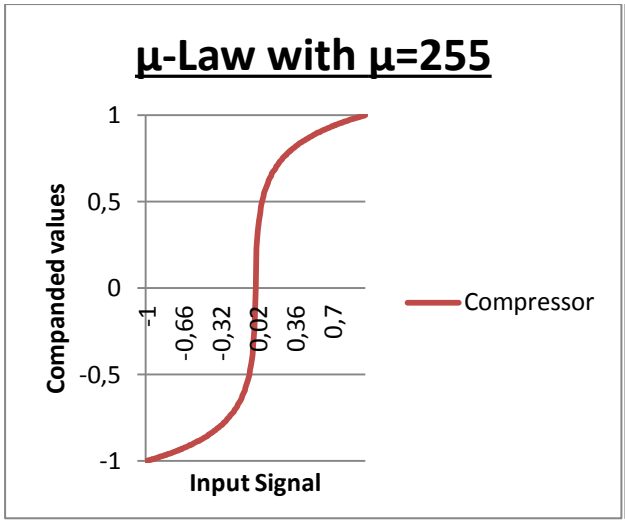
The section follows from the article [69].

The logarithmic compandor is a transformation which is applied to the random variable X. Two such logarithmic compandors are the μ-law compandor and the A-law compandor, where μ and A are parameters of the respective functions. Both of these compandors have been used extensively in telecommunication.

A logarithmic curve is used as the non-linear transformation, since perceived sound intensity is logarithmic in form, this leads to the companded signal being louder (increase in perceived intensity) than the original signal.

Logarithmic companding in telecommunications serves the dual purpose of efficiently encoding and increasing the perceived intensity of the analog signal [70].

[69] Brokish, C. W. Lewis, M. A-Law and mu-Law Companding Implementations using the TMS320C54x, MTSA. Texas Instruments. 1997.
[70] Gersho, A. and Gray, R. M. Vector Quantization and Signal Compression. Springer. 1992. Pages 159-160.



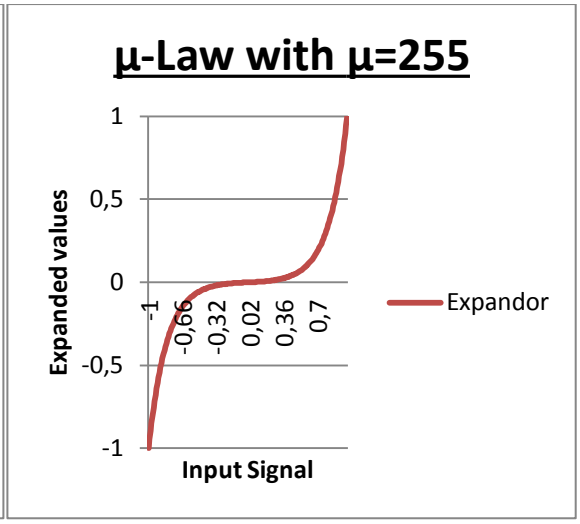


Figure 3.5 The Logarithmic Compressor Curve based on the μ-law (Left) Figure 3.6 The Logarithmic Expander Curve based on the μ-law (Right)

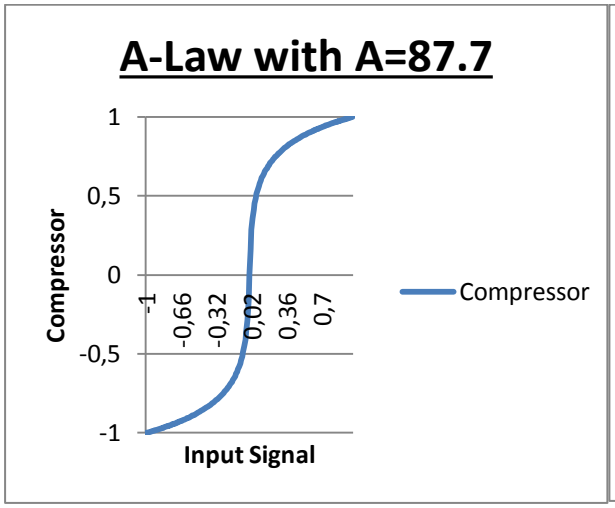
The μ-law

The compressor E and expandor E^{-1} with the compression intensity parameter μ

$$E_\mu(x) = \frac{\ln(1+\mu\cdot|x|)}{\ln(1+\mu)}\cdot sgn(x) \qquad \text{for } 0 \le |x| \le 1.$$

$$E_\mu^{-1}(x) = \left(\frac{1}{\mu}\right)\cdot((1+\mu)^{|x|}-1)\cdot sgn(x) \qquad \text{for } 0 \le |x| \le 1.$$

The μ -law algorithm is used to modify the dynamic range of an analog signal. The μ -law is used in the USA and in Japanese digital systems. The μ = 255 for 8-bit systems ($\mu = 2^8 - 1$) [71].



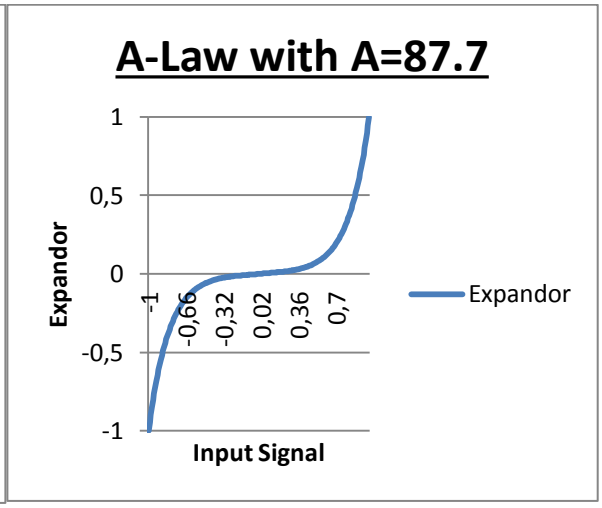


Figure 3.7 The Logarithmic Compressor Curve based on the A-law (Left) Figure 3.8 The Logarithmic Expander Curve based on the A-law (Right)

[71] Gersho, A. and Gray, R. M. Vector Quantization and Signal Compression. Springer. 1992. Page 160.

In figure 3.5 and figure 3.7, the compressor E of the logarithmic compandor is symmetric and approximates the logarithmic curve for positive values. By comparing the figures above, the A-law algorithm has a smaller dynamic range than the μ -law algorithm, this leads to less distortion for smaller analog signals, but at the cost of less dynamic range for larger analog signals.

The A-law

The compressor E and expandor E^{-1} with the compression intensity parameter A

$$E_A(x) = \begin{cases} \dfrac{A \cdot |x|}{1 + \ln(A)} \cdot sgn(x) & \text{for } 0 \leq |x| < \dfrac{1}{A} \\ \dfrac{(1 + \ln(A \cdot |x|))}{1 + \ln(A)} \cdot sgn(x) & \text{for } \dfrac{1}{A} \leq |x| \leq 1 \end{cases}.$$

$$E_A^{-1}(x) = \begin{cases} \dfrac{(1 + \ln(A)) \cdot |x|}{A} \cdot sgn(x) & \text{for } 0 \leq |x| < \dfrac{1}{(1 + \ln(A))} \\ \dfrac{\exp(|x| \cdot (1 + \ln(A)) - 1)}{A} \cdot sgn(x) & \text{for } \dfrac{1}{(1 + \ln(A))} \leq |x| < 1 \end{cases}$$

The A-law is used in European digital systems. In Europe the $A = 87.7\ (or\ 87.6)$ [72].

3.6.3.3 Piece-wise Companding

The piece-wise compandor has a range made up of several segments, each of which consists of uniformly spaced intervals, with differing step sizes for different segments to accommodate the dynamic range of the input signal. Piece-wise uniform scalar quantizers are often used in the place of the compandor's compressor curve.

According to the book, [73] given any finite regular scalar quantizer, there exists a piece-wise compandor model to match it.

Assume that Q is a finite regular scalar quantizer, with the set of ordered boundary points $\{a_i\}_{i=1}^{k}$ and the reproduction point set $\{y_i\}_{i=1}^{k}$, and then define the set of points in the Cartesian plane.

$$\boldsymbol{\gamma}_i = (y_i\ , i \cdot \Delta + c), i = 1, 2, \dots, k$$

where y_i is the x co-ordinate and $i \cdot \Delta + c$ is the y co-ordinate. With $\Delta > 0$ constant, and $c \in \mathbb{R}$ constant.

[72] Gersho, A. and Gray, R. M. Vector Quantization and Signal Compression. Springer. 1992. Page 160.
[73] The proof and result follows from the one given in the book, Gersho, A. and Gray, R. M. Vector Quantization and Signal Compression. Springer. 1992. Pages 157-158.

Similarly the set of points

$$\boldsymbol{\alpha}_i = \left(a_i\,, i \cdot \Delta + \frac{\Delta}{2} + c\right), i = 1, 2, \ldots, k-1.$$

can be plotted on the Cartesian plane, where a_i is the x co-ordinate and $i \cdot \Delta + \frac{\Delta}{2} + c$ is the y co-ordinate.

The point $\boldsymbol{\gamma}_1$ is connected to $\boldsymbol{\alpha}_1$ using a straight line, then $\boldsymbol{\alpha}_1$is connected to $\boldsymbol{\gamma}_2$ using a straight line and $\boldsymbol{\gamma}_2$ is then connected to $\boldsymbol{\alpha}_2$, and continuing in this way, each point $\boldsymbol{\alpha}_i$ is connected to the points $\boldsymbol{\gamma}_{i-1}$ and $\boldsymbol{\gamma}_i$ by means of lines with varying gradients.

The resulting graph E will be piecewise continuous and monotone increasing over the interval (y_1, y_k).

For any predefined set of boundary points $\{a_i\}_{i=1}^{k}$, reproduction points $\{y_i\}_{i=1}^{k}$ and a finite regular scalar quantizer one can obtain a set of uniformly spaced, transformed reproduction points and boundary points in a one-to-one relationship with their untransformed counterparts.

Therefore, for any finite regular quantizer Q, there exists a compandor E to match it.

A smooth monotone increasing function can replace the piecewise defined function described above. In the book [74] piece-wise uniform scalar quantizers are considered as a practical version of logarithmic compandors.

Companding is used in several applications other than telephony, in particular problems where high dynamic range causes visual and audio inconsistencies in image and audio compression.

There are many discussion on the topic, one interesting on the use of companding in the compression of HDR images is Li, Y. Sharan, L. Adelson, E. H. Compressing and Companding High Dynamic Range Images with Subband Arcitectures. 2005.

[74] Gersho, A. and Gray, R. M. Vector Quantization and Signal Compression. Springer. 1992. Pages 160-162 entitled piece-wise uniform scalar quantizers.

3.6.4 Lloyd-Max Scalar Quantization: The Optimal Problem

Another approach to designing a scalar quantizer for the random variable X comes from the minimization of the optimal problem, which is given below.

If X is any continuous random variable with probability density function f_X and $Q(X)$ is a scalar quantizer then the optimal problem can be considered as finding the boundary values $\{a_i\}_{i=0}^{k}$ and the reproduction points $\{y_i\}_{i=1}^{k}$ satisfying

$$D_{optimal} = \min_{\{a_i\},\{y_i\}} \left(\sum_{i=1}^{k} \int_{a_{i-1}}^{a_i} d(x, y_i) \cdot f_X(x) dx \right).$$

If the optimal problem is successfully minimized, then the resulting (generally non-uniform) scalar quantizer design will be optimal in terms minimizing the combined distortion.

In general, there is no closed form for solving the above mentioned minimization problem. There are however, effective algorithms to deal with the optimal problem, one such algorithm is the Lloyd-Max algorithm.

3.6.4.1 Deriving the Lloyd-Max Algorithm (Partial Solutions)

If the distortion measure d is the squared error distortion, then the optimal problem is given by

$$D_{optimal} = \min_{\{a_i\},\{y_i\}} \sum_{i=1}^{k} \int_{a_{i-1}}^{a_i} (y_i - x)^2 \cdot f_X(x) dx.$$

Consider the following two partial problems,

1. Conditioning on boundary values $\{ a_i\}_{i=0}^{k}$ and finding the optimal reproduction set $\{y_i\}_{i=1}^{k}$.
2. Conditioning on the reproduction set $\{y_i\}_{i=1}^{k}$ and finding the optimal boundary values $\{a_i\}_{i=0}^{k}$.

Conditioning on the Boundary Values

Assuming that $\{I_i\}_{i=1}^{k} = \{(a_{i-1}, a_i)\}_{i=1}^{k}$ are fixed, then minimizing D with respect to $\{y_i\}_{i=1}^{k}$. That is find $\{y_i\}_{i=1}^{k}$ such that

$$D = \min_{\{y_i\}} \left(\sum_{i=1}^{k} \int_{a_{i-1}}^{a_i} (y_i - x)^2 \cdot f_X(x) dx \right)$$

$$= \min_{\{y_i\}} \left(\sum_{i=1}^{k} D_i \right)$$

where $D_i = \int_{a_{i-1}}^{a_i} (y_i - x)^2 \cdot f_X(x)dx \ \forall\ i = 1,2,3,\dots,k.$

By differenting D_i in terms of (y_i) and setting the derivative equal to zero

$$2\int_{a_{i-1}}^{a_i} (y_i - x) \cdot f_X(x)dx = 0$$

$$y_i \int_{a_{i-1}}^{a_i} f_X(x)dx = \int_{a_{i-1}}^{a_i} x \cdot f_X(x)dx$$

from which it follows that

$$y_i = \frac{\int_{a_{i-1}}^{a_i} x \cdot f_X(x)dx}{\int_{a_{i-1}}^{a_i} f_X(x)dx}.$$

This can be written as

$$y_i = \frac{\int_{-\infty}^{\infty} x \cdot f_X(x) \cdot S_i(x)dx}{f_X(x \in (a_{i-1}, a_i))}$$

where the indicator function is given by

$$S_i(x_j) = \begin{cases} 1 \ if & x_j \in (a_{i-1}, a_i) \\ 0 & x_j \notin (a_{i-1}, a_i) \end{cases}$$

$$= \int_{-\infty}^{\infty} x \cdot f_X\big(x | x \in (a_{i-1}, a_i)\big)dx.$$

This is the conditional expectation over the sub-interval I_i and is known as the centroid condition.

$$y_i = \mathbb{E}(X | X \in (a_{i-1}, a_i)) \ \forall\ i = 1, 2, \dots, k.$$

<u>Conditioning on the Reproduction Set</u>

Assuming that $\{y_i\}_{i=1}^k$ are fixed, then minimizing D.

$$D = \min_{\{a_i\}} \left(\sum_{i=1}^{k} \int_{a_{i-1}}^{a_i} (y_i - x)^2 \cdot f_X(x)dx \right)$$

Notice that if we define the interval I_j such that $I_j = \left\{x: (y_j - x)^2 \leq (y_i - x)^2\right\}$ for all $i = 1,2,\dots,k$. Then for any density function f_X we have

$$\int_{I_j} (y_j - x)^2 \cdot f_X(x)dx \leq \int_{I_j} (y_i - x)^2 \cdot f_X(x)dx.$$

Summing over all the intervals I_j and the reproduction points y_j

$$\sum_{j=1}^{k}\int_{I_j}(y_j-x)^2\cdot f_X(x)dx \leq \sum_{j=1}^{k}\int_{I_j}(y_i-x)^2\cdot f_X(x)dx.$$

Therefore by choosing the intervals interval I_j such that $I_j=\left\{x:(y_j-x)^2\leq(y_i-x)^2\right\}$ for all $i=1,2,\dots,k$, then D can be minimized for the set of reproduction points $\{y_i\}_{i=1}^{k}$ for any density function f_X.

The set of boundary points $\{a_j\}_{j=1}^{k}$ corresponding to the intervals $I_j=(a_{j-1},a_j)$ must satisfy the equation

$(y_{j-1}-a_{j-1})^2=(y_j-a_{j-1})^2$ for all $j=2,3,\dots,k$.

This is the midpoint of the reproduction points y_{j-1} and y_j defined by

$a_{j-1}=\frac{y_{j-1}+y_j}{2}$ for all $j=2,3,\dots,k$.

This is known as the nearest neighbour condition.

3.6.4.2 Application of the Lloyd-Max Algorithm to Empirical data

According to the book, [75] direct application of the Lloyd-Max Algorithm to empirical data is often not tractable, specifically determining the reproduction set using the centroid condition, often numerical integration and Monte Carlo simulation are used in tandem to solve this problem.

The direct application of the Lloyd-Max Algorithm to the problem requires the distribution function which is typically not known.

The unbiased estimator of the conditional expected value $\mathbb{E}(X|X\in(a_{i-1},a_i))$ is the conditional sample mean from which estimated boundary points can be determined without estimating the probability function.

Since determining the conditional sample mean depends on the yet to be determined boundary points, a set of k sample means can arbitrarily [76] be assigned and the optimal boundary points can then be calculated to match the sample means. This unsupervised, iterative process is commonly known as the k-means algorithm or the Linde-Buzo-Gray algorithm and will be discussed below.

[75] Discussion follows the discussion presented in the book, Gersho, A. and Gray, R. M. Vector Quantization and Signal Compression. Springer. 1992. Pages 190-191.

[76] There are several considerations when initializing the reproduction set, see 1. Meila, M. and Heckerman, D. An experimental comparison of several clustering methods. 1998. and 2. Bradley , P. S. Microsoft Research and Fayyad, U. M. Microsoft Research. Refining Initial Points for K-Means Clustering. 1998.

3.6.4.3 The Linde-Buzo-Gray Algorithm

1. Starting with initial reproduction set $\{y_{t,i}\}_{i=1}^{k}$ for iteration $t = 1$.
2. Apply nearest neighbour partitioning using $\{y_{t,i}\}_{i=1}^{k}$ to obtain the boundary points for each of the intervals $I_{t,i}$

   $$a_{t,i-1} = \frac{y_{t,i-1} + y_{t,i}}{2} \; \forall \; i = 2,3, \dots, k.$$

   Assign all observations $\{x_j\}$ within the interval $I_{t,i}$ to the nearest reproduction point $y_{t,i}$.
3. Apply the centroid condition to obtain the new reproduction set $\{y_{t+1,i}\}_{i=1}^{k}$ by finding the mean of all the observations $\{x_j\}$ that have been assigned to the reproduction point $y_{t,i}$.

   $$y_{t+1,i} = \frac{\sum_{i=1}^{M} x_j \cdot S_{t,i}(x_j)}{\sum_{i=1}^{M} S_{t,i}(x_j)} \quad i = 1,2, \dots, k$$

   where

   $$S_i(x_j) = \begin{cases} 1 \; if & x_j \in (a_{i-1}, a_i) \\ 0 & x_j \notin (a_{i-1}, a_i) \end{cases}.$$

4. Determine the estimated combined distortion as

   $$D_{t+1} = \frac{1}{M} \cdot \sum_{i=1}^{k} \sum_{j=1}^{M} S_{t+1,i}(x_j) \cdot d(x_j, y_{t+1,i}).$$

5. Stop if some threshold is reached, else return to 2.

If a point x_j is equidistant from two reproduction points y_i and y_{i+1}, then an allocation rule is necessary to assign the point uniquely to one interval.

A threshold is also required to stop the iteration process as the iteration process will converge. The threshold may be compared to the relative decrease in estimated combined distortion $\frac{D_{t+1} - D_t}{D_t}$. Another option is comparing the threshold to the maximum change in the reproduction set points $\max_i(y_{t+1,i} - y_{t,i})$.

3.6.4.4 A Few Properties of Lloyd Max Scalar Quantization

The book [77] gives several properties of Lloyd Max scalar quantizers, these are stated with proofs form the book below.

Property One

A quantizer that satisfies the centroid condition will satisfy

$$\mathbb{E}\big(X - Q(X)\big) = 0.$$

Starting with the centroid condition

$$Q(X) = \mathbb{E}\big(X \big| X \in (a_{i-1}, a_i)\big).$$

$$\mathbb{E}\big(Q(X)\big) = \mathbb{E}\left(\mathbb{E}\big(X \big| X \in (a_{i-1}, a_i)\big)\right) = \mathbb{E}(X).$$

So $\mathbb{E}\big(X - Q(X)\big) = 0.$

Property Two

A quantizer that satisfies the centroid condition will satisfy

$$\mathbb{E}\left(Q(X) \cdot \big(X - Q(X)\big)\right) = 0.$$

Starting with

$$\mathbb{E}\left(Q(X) \cdot \big(X - Q(X)\big)\right)$$

$$= \mathbb{E}\left(\mathbb{E}\big(Q(X) \cdot \big(X - Q(X)\big) | X \in (a_{i-1}, a_i)\big)\right).$$

Notice that if $X \in (a_{i-1}, a_i)$ then $Q(X) = y_i$, so then it follows that

$$= \mathbb{E}\big(\mathbb{E}(y_i \cdot (X - y_i) | X \in (a_{i-1}, a_i))\big)$$

$$= \mathbb{E}\left(\big(y_i \cdot \mathbb{E}(X | X \in (a_{i-1}, a_i)\,)\big)\, - {y_i}^2\right)$$

$$\mathbb{E}\left(Q(X) \cdot \big(X - Q(X)\big)\right) = 0.$$

The reproduction set $\{y_j\}_{j=1}^{k}$ is orthogonal to the quantizer error.

[77] Gersho, A. and Gray, R. M. Vector Quantization and Signal Compression. Springer. 1992. Pages 180-182.

Property Three

A quantizer that satisfies the centroid condition will satisfy

$$\mathbb{E}\left(X \cdot \left(X - Q(X)\right)\right) \geq 0.$$

Starting with

$$\mathbb{E}\left(X \cdot \left(X - Q(X)\right)\right) = \mathbb{E}(X^2) - \mathbb{E}\big(X \cdot Q(X)\big)$$

$$= \mathbb{E}(X^2) - \mathbb{E}\big(Q^2(X)\big).$$

Notice that from property two, $\mathbb{E}(Q(X) \cdot X) = \mathbb{E}\big(Q^2(X)\big)$, it follows then that

$$= \mathbb{E}(X^2) - 2 \cdot \mathbb{E}\big(X \cdot Q(X)\big) + \mathbb{E}\big(Q^2(X)\big)$$

$$= \mathbb{E}\left(\left(X - Q(X)\right)^2\right) \geq 0$$

$$\therefore \mathbb{E}\left(X \cdot \left(X - Q(X)\right)\right) \geq 0.$$

A random variable X is positively correlated with the quantizer error [78].

Property Four

The nearest neighbour condition implies scalar quantizer convexity, that is, if

$x_\gamma \in (x_\alpha, x_\beta)$ and $Q(x_\alpha) = Q\big(x_\beta\big) = y_i$, then $Q\big(x_\gamma\big) = y_i$.

Firstly, if any reproduction point $y_i \in \{y_j\}_{j=1}^{k}$ and any two points $x_\alpha, x_\beta \in I$ where $x_\alpha < x_\beta$ that satisfy $|x_\alpha - y_i|^2 < |x_\alpha - y_j|^2 \; \forall \, j \neq i, j = 1,2, \dots, k$ and similarly $|x_\beta - y_i|^2 < |x_\beta - y_j|^2 \; \forall \, j \neq i, j = 1,2, \dots, k$.

Then for any point $x_\gamma \in (x_\alpha, x_\beta)$ we show that

$$|x_\gamma - y_i|^2 < |x_\gamma - y_j|^2 \; \forall \, j \neq i, j = 1,2, \dots, k.$$

Starting with

$x_\gamma = t \cdot x_\alpha + (1 - t) \cdot x_\beta$ where $t \in (0,1)$ and $\left|x_\alpha - x_\beta\right| = \left|x_\alpha - x_\gamma\right| + \left|x_\gamma - x_\beta\right|$

$$\left|x_\gamma - y_i\right| = \left|t \cdot x_\alpha + (1 - t) \cdot x_\beta - y_i\right|.$$

[78] According to the author, the quantizer error $\left(x - Q(x)\right)$ is a deterministic function of the input data x. Gersho, A. and Gray, R. M. Vector Quantization and Signal Compression. Springer. 1992. Page 182.

$$= \left|(t \cdot x_\alpha - t \cdot y_i) + \left((1-t) \cdot x_\beta - (1-t) \cdot y_i\right)\right|$$

$$= (|t \cdot x_\alpha - t \cdot y_i|) + \left(\left|(1-t) \cdot x_\beta - (1-t) \cdot y_i\right|\right)$$

$$= (t \cdot |x_\alpha - y_i|) + \left((1-t) \cdot \left|x_\beta - y_i\right|\right)$$

$$\leq \left(t \cdot \left|x_\alpha - y_j\right|\right) + \left((1-t) \cdot \left|x_\beta - y_j\right|\right) \text{ for all } j \neq i, j = 1,2,\dots,k.$$

$$\left(t \cdot \left|x_\alpha - y_j\right|\right) + \left((1-t) \cdot \left|x_\beta - y_j\right|\right).$$

$$= \left|\left(t \cdot x_\alpha - t \cdot y_j\right) + \left((1-t) \cdot x_\beta - (1-t) \cdot y_j\right)\right|$$

$$= \left|t \cdot x_\alpha + (1-t) \cdot x_\beta - y_j\right|$$

$$= \left|x_\gamma - y_j\right|.$$

$$|x_\gamma - y_i|^2 < |x_\gamma - y_j|^2 \text{ for all } j \neq i, j = 1,2,\dots,k.$$

Therefore if $x_\gamma \in (x_\alpha, x_\beta)$ and $Q(x_\alpha) = Q(x_\beta) = y_i$, then $Q(x_\gamma) = y_i$.

Property Five

The nearest neighbour condition implies scalar quantizer regularity.

As shown in property four, the nearest neighbour condition implies scalar quantizer convexity. The allocation rule forces the reproduction point y_j to be contained on the open interval (a_{j-1}, a_j), Therefore, a scalar quantizer Q that satisfies the nearest neighbour conditions is regular.

Scalar quantizer regularity is an important property of Lloyd Max scalar quantizers as it is a requirement for optimal scalar quantization as will be discussed in section 3.7.

3.7 High Code Rate Estimate of Distortion

In the book [[79]], the authors discuss the optimality of the uniform scalar quantizer, for a high code rate, over any other scalar quantizers.

The authors first establish an expression for the estimated combined distortion D, then the author uses the point density function $\lambda(x)$ to obtain the distortion integral. Finally, the distortion integral is then used to show that a uniform scalar quantizer combined with an entropy encoder is the best scalar quantizer for a high code rate.

The first and last steps are included with details in order to better understand the results obtained.

3.7.1 Estimated Combined Distortion for High Code Rate Quantization

Consider a random variable X which follows a smooth non-uniform probability distribution f_X, then for a very large reproduction set $\{y_i\}_{i=1}^{k}$ and k small sub-intervals where the granular region comprises of most if not all of the support of X.

If a regular scalar quantizer Q is applied to X then the combined mean squared error distortion D is

$$D = \sum_{i=1}^{k} \int_{a_{i-1}}^{a_i} (x - y_i)^2 \cdot f_X(x)\, dx.$$

Since the granular interval $[a_1, a_{k-1}]$ includes most if not all the probability, we have

$$\int_{a_1}^{a_{k-1}} f_X(x) dx \approx 1.$$

Since k is large and f is smooth, the partitions are small enough to be approximated by a constant probability f_i over the sub-interval (a_{i-1}, a_i), so

$$f_Y(y_i) = \int_{a_{i-1}}^{a_i} f_X(x)\, dx \approx f_i \cdot (a_i - a_{i-1}).$$

In other words,

$$f_i \approx \frac{f_Y(y_i)}{(a_i - a_{i-1})}.$$

The combined mean squared error distortion D is given by,

[79] The High Resolution Approximation and the Uniform Scalar Quantizer follows directly from the proof in Gersho, A. and Gray, R. M. Vector Quantization and Signal Compression. Springer. 1992. Pages 161-162 and Pages 298-300.

$$D \approx \sum_{i=1}^{k} f_Y(y_i) \cdot \int_{a_{i-1}}^{a_i} \frac{(x-y_i)^2}{(a_i - a_{i-1})}\, dx.$$

If the codeword y_i is chosen as the midpoint of the interval $(a_i - a_{i-1})$, which is the nearest neighbour condition for optimal quantization of a uniform random variable, then the above integral becomes

$$\int_{a_{i-1}}^{a_i} \frac{(x-y_i)^2}{(a_i - a_{i-1})}\, dx = \int_{a_{i-1}}^{a_i} \frac{\left(x - \frac{(a_i + a_{i-1})}{2}\right)^2}{(a_i - a_{i-1})}\, dx$$

$$= \frac{1}{(a_i - a_{i-1})}\left(\int_{a_{i-1}}^{a_i} x^2 - (a_i + a_{i-1}) \cdot x + \left(\frac{(a_i + a_{i-1})}{2}\right)^2\, dx\right)$$

$$= \frac{1}{(a_i - a_{i-1})}\left(\frac{(a_i - a_{i-1})^3}{3} - \frac{(a_i + a_{i-1}) \cdot (a_i - a_{i-1})^2}{2} + (a_i - a_{i-1}) \cdot \left(\frac{(a_i + a_{i-1})}{2}\right)^2\right)$$

$$= \frac{1}{(a_i - a_{i-1})}\left(\frac{(a_i - a_{i-1})^3}{3} - \frac{(a_i + a_{i-1}) \cdot (a_i - a_{i-1})^2}{2} + (a_i - a_{i-1}) \cdot \left(\frac{(a_i + a_{i-1})}{2}\right)^2\right)$$

$$= \left(\frac{(a_i - a_{i-1})^2}{3} - \frac{(a_i + a_{i-1}) \cdot (a_i - a_{i-1})}{2} + \left(\frac{(a_i + a_{i-1})}{2}\right)^2\right)$$

$$= \frac{(a_i - a_{i-1})^2}{12}.$$

By substituting the expression above into the combined mean squared error distortion we get

$$D \approx \sum_{i=1}^{k} f_Y(y_i) \cdot \frac{(a_i - a_{i-1})^2}{12}.$$

3.7.2 Uniform Scalar Quantizer as an Optimal Scalar Quantizer

The authors [80] define the point density function $\lambda(x)$ as given by

$$\lambda(x) = \lim_{k\to\infty} \frac{k(x)}{k}$$

where $k(x)dx$ is the number of quantization levels between $(x, x + dx)$.

The authors also show that for k large the interval length is approximately given by

$$(a_i - a_{i-1}) \approx \frac{1}{k \cdot \lambda(y_i)}.$$

The authors then prove that the combined distortion is given by the distortion integral, given by

$$D \approx \frac{1}{12} \cdot \mathbb{E}_X \left(\frac{1}{\left(k \cdot \lambda(y)\right)^2} \right)$$

where the expectation is taken over the granular region of the random variable X.

The authors finally show that scalar quantizer that achieves the minimum entropy for a given average distortion is the uniform scalar quantizer with a large reproduction set.

As in section 3.7.1, if k is large and f is smooth, then the probability mass function can be approximated by the rectangular areas,

$$f_Y(y_i) \approx (a_i - a_{i-1}) \cdot f_X(y_i).$$

But, as stated earlier, this expression can be modified by the approximation,

$$(a_i - a_{i-1}) \approx \frac{1}{k \cdot \lambda(y_i)}.$$

By substitution, we get

$$f_Y(y_i) \approx \frac{f_X(y_i)}{k \cdot \lambda(y_i)}.$$

In section 2.4 it was shown that the lower bound of the code rate for encoding a discrete random variable Y with probability mass function $f_Y(y)$ is given by the entropy,

$$H(Y, \boldsymbol{\theta}, 1) = -\sum_{i=1}^{k} f_Y(y_i) \cdot log_2\big(f_Y(y_i)\big).$$

[80] Gersho, A. and Gray, R. M. Vector Quantization and Signal Compression. Springer. 1992. Pages 163-164.

By substituting $f_Y(y_i)$ into $H(Y, \boldsymbol{\theta}, 1)$ we have,

$$H(Y, \boldsymbol{\theta}, 1) = -\sum_{i=1}^{k}\left((a_i - a_{i-1}) \cdot f_X(y_i)\right) \cdot log_2\left(\frac{f_X(y_i)}{k \cdot \lambda(y_i)}\right)$$

$$= -\sum_{i=1}^{k}\left((a_i - a_{i-1}) \cdot f_X(y_i)\right) \cdot log_2\left(f_X(y_i)\right) - \sum_{i=1}^{k}\left((a_i - a_{i-1}) \cdot f_X(y_i)\right) \cdot log_2\left(\frac{1}{k \cdot \lambda(y_i)}\right).$$

Allowing the number of reproduction points $k \to \infty$, then we have,

$$\approx -\int_{-\infty}^{\infty}\left(\left(f_X(y)\right) \cdot log_2\left(f_X(y)\right)\right) dy - \int_{-\infty}^{\infty}\left(\left(f_X(y)\right) \cdot log_2\left(\frac{1}{k \cdot \lambda(y)}\right)\right) dy$$

$$= -\int_{-\infty}^{\infty}\left(\left(f_X(y)\right) \cdot log_2\left(f_X(y)\right)\right) dy - \mathbb{E}_X\left(log_2\left(\frac{1}{k \cdot \lambda(X)}\right)\right)$$

$$= -\int_{-\infty}^{\infty}\left(\left(f_X(y)\right) \cdot log_2\left(f_X(y)\right)\right) dy - \frac{1}{2} \cdot \mathbb{E}_X\left(log_2\left(\frac{1}{\left(k \cdot \lambda(X)\right)^2}\right)\right).$$

Due to the convexity of $-log_2(x)$ and by Jensen's Inequality [81] we have

$$\geq -\int_{-\infty}^{\infty}\left(\left(f_X(y)\right) \cdot log_2\left(f_X(y)\right)\right) dy - \frac{1}{2} \cdot \left(log_2\left(\mathbb{E}\left(\frac{1}{\left(k \cdot \lambda(X)\right)^2}\right)\right)\right).$$

Then by substituting the distortion integral into the above expression, we get

$$\approx -\int_{-\infty}^{\infty}\left(\left(f_X(y)\right) \cdot log_2\left(f_X(y)\right)\right) dy - \frac{1}{2} \cdot \left(log_2(12 \cdot D)\right).$$

$H(Y, \boldsymbol{\theta}, 1)$ is bounded below by

$$-\int_{-\infty}^{\infty}\left(\left(f_X(y)\right) \cdot log_2\left(f_X(y)\right)\right) dy - \frac{1}{2} \cdot \left(log_2(12 \cdot D)\right).$$

With the lower bound achievable if and only if $\lambda(X)$ is constant, which is if the quantization point density function is uniform.

This proves that for a large reproduction set, uniform scalar quantization applied in tandem with an entropy encoder is in general better than any other scalar quantizer, independent of the input probability distribution.

[81] Cover, T. M. and Thomas, J. A. Elements of Information Theory. Wiley Series in Telecommunications and Signal Processing. Second Edition. 2002. Page 25.

3.8 Conclusions of Chapter Three

- A scalar quantizer approximates a random variable X with a set of reproduction points $R = \{y_i\}_{i=1}^{k}$ for the purpose of discretizing continuous data or reducing the information content for compression purposes.

- Scalar quantization is made up of three constituents, namely a scalar quantizer Q, a reproduction set $R = \{y_i\}_{i=1}^{k}$ and the set of boundary values which bound the quantization sub-intervals.

- The result of scalar quantization, is a discrete random variable $Y = Q(X)$ with probability mass function that is a function of the probability density/mass function of X.

- The scalar quantizer introduces redundancy by means of approximation, therefore a distortion measure is used to calculate the degree of dissimilarity between the input random variable X and output quantized random variable Y.

- The mean distortion is the combined distortion between X and Y which is calculated over the support of X.

- Scalar quantization can be achieved by both a uniform and non-uniform scalar quantization. The difference between uniform and non-uniform scalar quantization is the uniform and non-uniform length of the quantization sub-intervals respectively, that partition the support of X.

- Three forms of scalar quantization were considered, namely, uniform scalar quantization, companded scalar quantization and Lloyd Max scalar quantization.

- Companding applies a non-linear transformation to the random variable X, before applying uniform scalar quantization to the transformed variable. Three compandors are considered, namely, probability integral transform companding, logarithmic companding and piecewise companding.

- Lloyd Max scalar quantization is the result of the minimization of the optimal problem of minimizing the combined distortion for the Euclidian distortion measure.

- The nearest neighbour condition and centroid condition are obtained by conditionally minimizing the mean square error distortion in the optimal problem.

- Linde-Buzo-Gray algorithm is the iterative procedure that attempts to find an optimal reproduction set and corresponding boundary points.

- The centroid condition results in several properties including the positive correlation between the random variable X and the quantization error.

- The nearest neighbour condition results in scalar quantizer convexity and regularity.

- In the case of a large reproduction set, uniform scalar quantization outperforms any other scalar quantizers when used in conjunction with an entropy encoder.

Chapter Four: Vector Quantization

4.1 Vector Quantization

Vector quantization is the extension of scalar quantization, that is, vector quantization is applied to a random vector $\boldsymbol{X} \in \mathbb{R}^m$ where the vector $\boldsymbol{X}$ is quantized as a single unit. The quantization of a random vector results in a set of discrete, countable random quantized vectors.

A vector quantizer Q of $\boldsymbol{X}$, is defined as the mapping of the random vector $\boldsymbol{X}$ onto the reproduction set R.

$$Q(\boldsymbol{X}): \mathbb{R}^m \to R$$

where $R = \{\boldsymbol{y}_i\}_{i=1}^{k}$ and $\boldsymbol{y}_i \in \mathbb{R}^m$, with discrete joint probability mass function $f_{\boldsymbol{Y}}(\boldsymbol{y})$ where

$$f_{\boldsymbol{Y}}(\boldsymbol{y}_j) = \begin{cases} \int_{\mathcal{C}_j} dF_{\boldsymbol{X}} & if\ \boldsymbol{x} \in \mathcal{C}_j \\ 0 & if\ \boldsymbol{x} \notin \mathcal{C}_j \end{cases}.$$

4.2 Scalar Quantization to Vector Quantization [82]

Two immediate implications of extending from scalar to vector quantizers are the change from intervals to cells and the change of the quantizer function.

4.2.1 Intervals to Cells

Scalar quantization is applied to random variables of dimension $m = 1$, although these random variables may constitute the elements of a random vector, scalar quantization is the element-wise application of a scalar quantizer to a random vector $\boldsymbol{X}$.

The element-wise scalar quantization of the vector components will result in rectangular partitions, due to scalar quantization in each dimension, this is known as product scalar quantization.

For example, if $m = 2$, the random vector $\boldsymbol{X} \in \mathbb{R}^2$ where $\boldsymbol{X}^T = \begin{pmatrix} X_1 \\ X_2 \end{pmatrix}$ may be quantized using the scalar quantizer Q which is applied to each element of the vector $\boldsymbol{X}$ where $Q(X_1)$ is the mapping from I_1 onto the reproduction set $R_1 = \{y_i\}_{i=1}^{k_1}$ and $Q(X_2)$ is the mapping from I_2 onto the reproduction set $R_2 = \{y_j\}_{j=1}^{k_2}$.

[82] The random vector $\boldsymbol{X}$ will be considered to be continuous, all the theory can be extended to discrete random variables with a few variations to the theory, however, in practice the quantization methods discussed in this document remain alike.

The result of combining these two sets into a reproduction set, is the set of reproduction points $R_1 \times R_2 = \left\{\begin{pmatrix} y_i \\ y_j \end{pmatrix}\right\}_{i=1,j=1}^{k_1,k_2}$ and a grid in $\mathbb{R}^2$ of size $I_1 \times I_2$ of $k_1 \times k_2$ (possibly unequal length for non-uniform scalar quantization) rectangles which are known as rectangular cells.

Regular vector quantization on the other hand, results in Voronoi tessellations, which in general are regular, polygonal shapes (or polytopal shapes for $\mathbb{R}^m$, $m > 2$) of differing size, bounded by planes of lower dimension. An example of this can be seen in figure 4.1.

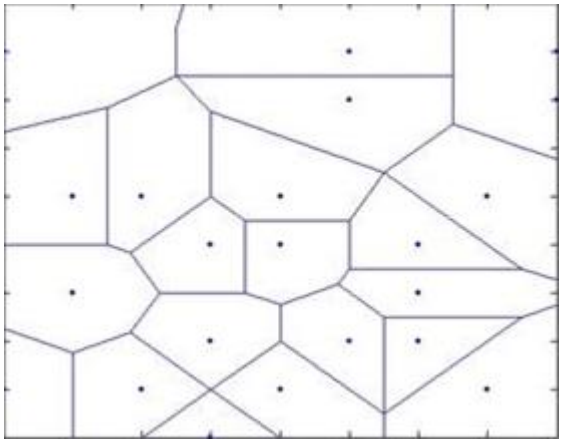


Figure 4.1 A Voronoi Tessellation [83].

A cell [84] $\mathcal{C} \subseteq \mathbb{R}^m$ is a generalization of the interval I in $\mathbb{R}$, this generalization will be restricted to the set of non-overlapping regular polytopes in this mini dissertation.

The region extending over the sample space of the random vector $\boldsymbol{X}$ with a joint probability density function $f_{\boldsymbol{X}}(\boldsymbol{x})$ is divided into k sub-regions or cells.

The cell $\mathcal{C}_i$ with $i \in \{1,2, \ldots , k\}$ is defined as

1. $\mathcal{C}_i = \{ \boldsymbol{X} \in \mathbb{R}^m : Q(\boldsymbol{X}) = \boldsymbol{Y}_i \}$.
2. $\mathcal{C} = \bigcup_{i=1}^{k} \mathcal{C}_i$.
3. $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset \; \forall \; i \neq j$.

with inverse image $\mathcal{C}_i = Q^{-1}(\boldsymbol{y}_i)$ with the reproduction set $\{\boldsymbol{y}_i\}_{i=1}^{k}$.

The cell $\mathcal{C}_i$ will be considered in more detail next.

[83] Image obtained by means of a Google search http://dasl.mem.drexel.edu/Hing/VoronoiTutorial.htm.
[84] Throughout chapter four, the cell will be referred to as $\mathcal{C}$, this is not to be confused with the codebook C from chapters one and two.

4.2.1.1 Boundedness of $\mathcal{C}_i$

The cell $\mathcal{C}_i$ is unbounded if there exists an $\boldsymbol{x} \in \mathcal{C}_i$ such that

$$\boldsymbol{x} \notin (B_r(\boldsymbol{c}_i) \cap \mathcal{C}_i) \; \forall \, r \in \mathbb{N}$$

where the sphere of radius r is given by $B_r(\boldsymbol{c}_i)$ with $\boldsymbol{c}_i$ as the centroid of the cell $\mathcal{C}_i$

$$B_r(\boldsymbol{y}_i) = \{\boldsymbol{x}: \| \boldsymbol{x} - \boldsymbol{c}_i \| < r, r \in \mathbb{N}\}.$$

An unbounded cell is known as overloaded and a set of overloaded cells is known as an overloaded region, while a bounded cell is known as granular and a set of granular cells is known as a granular region.

4.2.1.2 Boundaries of $\mathcal{C}_i$

The boundaries of the cell $\mathcal{C}_i$ are the set of planes P_v

$$P_v = \{ \boldsymbol{x} \in \mathcal{C}_i : \boldsymbol{m}_v^T \cdot \boldsymbol{x} + b_v = 0\}$$

which intersect to form closed bounded polytopals in the granular regions of $\mathbb{R}^m$ and open unbounded regions in the overloaded regions of $\mathbb{R}^m$.

4.2.1.3 Regular Polytopal Cell

The polytopal cell $\mathcal{C}_i$ is defined in [85] as the intersection of L_i half spaces,

$$\mathcal{C}_i = \bigcap_{v=1}^{L_i} H_v$$

$$H_v = \{ \boldsymbol{x} \in \mathbb{R}^m : \boldsymbol{m}_v^T \cdot \boldsymbol{x} + b_v \geq 0\}$$

where L_i is the number of $(m-1)$ dimensional faces of $\mathcal{C}_i$.

A regular polytopal cell is a polytopal cell that is both convex, that is for any $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{C}_i$

$\{\boldsymbol{x} : t \cdot \boldsymbol{x}_1 + (1 - t) \cdot \boldsymbol{x}_2 \; \forall \, t \in (0,1)\} \in \mathcal{C}_i$ and contains its reproduction point vector.

A polytopal cell is convex, this will be shown in property one.

Property One

If $\{\boldsymbol{x}_1, \boldsymbol{x}_2\} \in \mathcal{C}_i$, then $\boldsymbol{m}_v \cdot \boldsymbol{x}_1 + b_v \geq 0$ and $\boldsymbol{m}_v \cdot \boldsymbol{x}_2 + b_v \geq 0 \;\; for\ all\ v = 1,2, \dots, L_i$.

For $a \in (0,1)$ we have,

$a \cdot \boldsymbol{m}_v \cdot \boldsymbol{x}_1 + a \cdot b_v \geq 0$ and $(1-a) \cdot \boldsymbol{m}_v \cdot \boldsymbol{x}_2 + (1-a) \cdot b_v \geq 0 \;\; for\ all \;\; v = 1,2, \dots, L_i$.

By adding the above two inequalities we get,

[85] Gersho, A. and Gray, R. M. Vector Quantization and Signal Compression. Springer. 1992. Page 320.

$$a \cdot \boldsymbol{m}_v \cdot \boldsymbol{x}_1 + a \cdot b_v + (1-a) \cdot \boldsymbol{m}_v \cdot \boldsymbol{x}_2 + (1-a) \cdot b_v \geq 0.$$

$$\boldsymbol{m}_v \cdot \boldsymbol{x}_2 + b_v + a \cdot \boldsymbol{m}_v \cdot (\boldsymbol{x}_1 - \boldsymbol{x}_2) \geq 0.$$

This implies that,

$$\boldsymbol{m}_v \cdot [a \cdot \boldsymbol{x}_1 + (1-a) \cdot \boldsymbol{x}_2] + b_v \geq 0 \; for \; all \;\; v = 1,2,\dots,L_i \; .$$

Therefore the set of half-spaces

$$H_v = \{ \boldsymbol{x} \in \mathbb{R}^m : \boldsymbol{m}_v \cdot \boldsymbol{x} + b_v \geq 0 \}$$

is a convex set and the intersection of any collection of convex subsets of $\mathbb{R}^m$ is a convex set [86]. Therefore the cell $\mathcal{C}_i$ is convex.

Figure 4.2 below illustrates the form of a regular cell in $\mathbb{R}^2$.

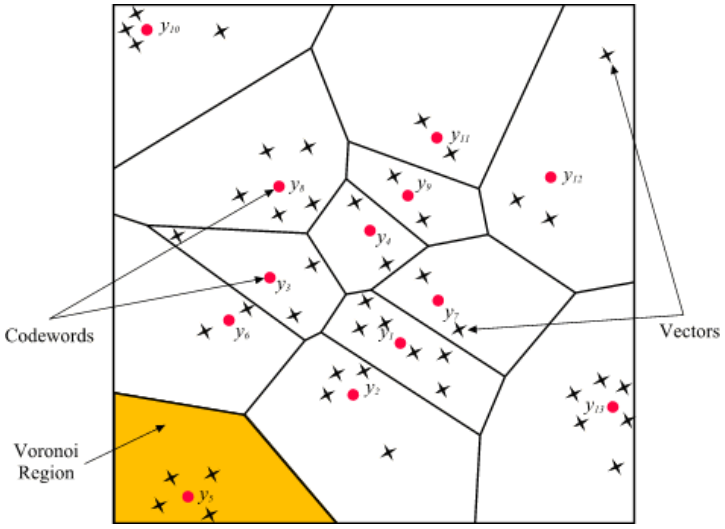


Figure 4.2 [87] Regular Vector Quantization The Voronoi regions are regular cells, their boundaries are linear, convex and they contain their reproduction points $\boldsymbol{y}_i$. The central cells are granular, since they are bounded. The cells on the exterior may be unbounded, and hence overloaded.

4.2.1.4 Decomposing the Vector Quantizer Function

In the book [88], the authors show how the vector quantizer can be decomposed into simpler constituent parts for application.

The vector quantizer Q is the composite function made up of an encoder E and a decoder D. The process of quantization then takes the composite form

$$\boldsymbol{y} = Q(\boldsymbol{x}) = D\big(E(\boldsymbol{x})\big).$$

[86] Bartle, R.G. The Elements of Real Analysis. Second Edition. Wiley. 1976. Page 59.
[87] Image was obtained from www.mqasem.net/vectorquantization/vq.html by means of a Google search.
[88] Gersho, A. and Gray, R. M. Vector Quantization And Signal Compression. Springer. 1992. Pages 317- 323.

The mapping of the random vector $\boldsymbol{X}$ onto an index i is achieved by the encoder $E: \mathbb{R}^m \rightarrow i$, while the mapping of the index i onto the reproduction set $R = \{\boldsymbol{y}_i\}_{i=1}^{k}$ is achieved by the decoder $D: i \rightarrow \mathbb{R}^m$. A brief summary is given below.

The vector encoder $E(\boldsymbol{x})$ can be defined as a composite function made up of a selector vector $\boldsymbol{S}$ $(k \times 1)$ and an index function A.

If an observed $\boldsymbol{x} \in \mathcal{C}_i$, then the selector vector

$\boldsymbol{S}(\boldsymbol{x}) = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$ has a 1 in the i^{th} position and 0's elsewhere and $A(\boldsymbol{S}(\boldsymbol{x})) = i$.

The i^{th} element of the selector vector $S_i(\boldsymbol{x})$ is determined by the indicator function

$$\phi(x) = \begin{cases} 1 \; if \; x \geq 0 \\ 0 \; if \; x < 0 \end{cases}.$$

The indicator function is applied to the scalar product $\boldsymbol{m}_v \cdot \boldsymbol{x} + b_v$, which will be positive if $\boldsymbol{x} \in \mathcal{C}_i$.

The selector vector, $S_i(\boldsymbol{x})$ is defined as

$$S_i(\boldsymbol{x}) = \prod_{v=1}^{L_i} \phi(\boldsymbol{m}_v \cdot \boldsymbol{x} + b_v)$$

where L_i is the number of $(m-1)$ dimensional intersecting half planes that constitute the cell C_i.

The decoder $D(i)$ is defined by the inverse function A^{-1} applied to the index i, where

$$D(i) = \sum_{l=1}^{k} \boldsymbol{y_l} \cdot [A^{-1}(i)]_l \, .$$

This assigns the index i to the reproduction point $\boldsymbol{y}_i$.

4.3 Vector Distortion

A vector distortion measure is the degree of dissimilarity or the approximation error between the input random vector $\boldsymbol{X}$ and output quantized random vector $\boldsymbol{Y}$.

The distortion or approximation error was described in the previous chapter, as a cost function that would assign a greater cost to greater deviation of the approximated value from the original value. The increase in dimensionality over that of scalar distortion measures provides both challenges and advantages in picking the correct distortion measure.

4.3.1 Five Distortion Measures [89]

Five distortion measures that are applicable for vector quantization will be considered in the next section.

4.3.1.1 L^2- Norm Distortion

$$d\big(\boldsymbol{x}, Q(\boldsymbol{x})\big) = ||\boldsymbol{x} - Q(\boldsymbol{x})||_2$$

where $||\cdot||_2 = ({x_1}^2 + {x_2}^2 + \cdots + {x_m}^2)^{\frac{1}{2}}$.

The L^2 - norm is a natural extension from the squared error distortion applied to scalars in $\mathbb{R}$ to the squared error distortion applied to vectors in $\mathbb{R}^m$.

4.3.1.2 L^P- Norm Distortion

The L^2- norm can be extended to the L^P- norm, where

$$d\big(\boldsymbol{x}, Q(\boldsymbol{x})\big) = ||\boldsymbol{x} - Q(\boldsymbol{x})||_P$$

where $||\cdot||_P = (|x_1|^P + |x_2|^P + \cdots + |x_m|^P)^{\frac{1}{P}}$.

4.3.1.3 Weighted Squared Error Distortion [90]

For a matrix of weights $\boldsymbol{W}$ that is symmetric and positive definite defines the weighted square distortion as

$$d\big(\boldsymbol{x}, Q(\boldsymbol{x})\big) = \big(\boldsymbol{x} - Q(\boldsymbol{x})\big)^T \boldsymbol{W} \big(\boldsymbol{x} - Q(\boldsymbol{x})\big).$$

One such matrix of weights is the inverse of the covariance matrix $\boldsymbol{S}$.

---

[89] See Deza, E.; Deza, M. (2006), "Dictionary of Distances", for more distance measures.
[90] Definitions obtained from the book Gersho, A. and Gray, R. M. Vector Quantization and Signal Compression. Springer. 1992. Pages 325-327.

If $\boldsymbol{S}$ is the covariance matrix of the random vector $\boldsymbol{X}$ where

$$\boldsymbol{S} = \mathbb{E}(\boldsymbol{X} - \mathbb{E}(\boldsymbol{X}))(\boldsymbol{X} - \mathbb{E}(\boldsymbol{X}))^{\boldsymbol{T}}.$$

The distortion measure $d\big(\boldsymbol{x}, Q(\boldsymbol{x})\big) = \big(\boldsymbol{x} - Q(\boldsymbol{x})\big)\boldsymbol{S}^{-1}\big(\boldsymbol{x} - Q(\boldsymbol{x})\big)^T$ is known as the Mahalanobis distance.

4.3.1.4 Noise Energy-to-Signal Energy Distortion

If $\boldsymbol{W} = (||\boldsymbol{x}||_2)^{-2}\mathbf{I}_m$ where $\mathbf{I}_m$ is the identity matrix of size m and $||\boldsymbol{x}||_2 > 0$, then

$$d\big(\boldsymbol{x}, Q(\boldsymbol{x})\big) = \frac{\big(||\boldsymbol{x}||_2 - Q(\boldsymbol{x})\big)^2}{(||\boldsymbol{x}||_2)^2}.$$

The Noise Energy-to-Signal Energy distortion is not a metric [91].

4.3.1.5 Log Likelihood Ratio Distortion

The Kullback-Leibler distance is given by

$$d\big(\boldsymbol{x}, Q(\boldsymbol{x})\big) = log_2\left(\frac{f_{\boldsymbol{X}}(\boldsymbol{x})}{f_{\boldsymbol{Y}}(\boldsymbol{y})}\right).$$

According to [92] Kullback-Leibler distance (or the expected log likelihood ratio) is the "coding penalty" for approximating the distribution $f_{\boldsymbol{X}}(\boldsymbol{x})$ with $f_{\boldsymbol{Y}}(\boldsymbol{y})$.

The Kullback-Leibler distance or one of its variations is a powerful "distance measure" that is often used in image compression. The Log Likelihood ratio is however not a metric.

For an interesting introduction to Kullback-Leibler distance see [93].

4.3.2 Combined Distortion of a Random Vector

Similarly to the previous chapter, the combined distortion is a function of the distortion measure computed over the support of the input random vector $\boldsymbol{X}$. The most commonly used combined distortion is the mean or expected distortion.

[91] The definition of a metric was given in section 3.5.1.
[92] Cover, T. M. and Thomas, J. A. Elements of Information Theory. Wiley Series in Telecommunications and Signal Processing. Second Edition. Page 19.
[93] Shlens, J. Notes on Kullback-Leibler Divergence and Likelihood Theory. 2007.

4.3.2.1 Mean Distortion

Let $\boldsymbol{X}$ be a continuous random vector with joint probability density function $f_{\boldsymbol{X}}(\boldsymbol{x})$.

The mean distortion is given by

$$D = \mathbb{E}\left(d\left(\boldsymbol{X}, Q(\boldsymbol{X})\right)\right) = \int_{\boldsymbol{X}} d\left(\boldsymbol{x}, Q(\boldsymbol{x})\right) dF_{\boldsymbol{X}}.$$

Now since the cells $\mathcal{C}_j$ are disjoint and $Q(\boldsymbol{x}) = \boldsymbol{y}_i \; \forall \; x \in \mathcal{C}_i$

$$D = \mathbb{E}\left(d\left(\boldsymbol{X}, Q(\boldsymbol{X})\right)\right) = \sum_{i=1}^{k} \int_{\mathcal{C}_i} d(\boldsymbol{x}, \boldsymbol{y}_i) dF_{\boldsymbol{X}}.$$

4.3.2.2 Sample Mean Distortion

To calculate the sample mean distortion for a set of observations $\{\boldsymbol{x}_i\}_{i=1}^{N}$ quantized according to the reproduction set $R = \{\boldsymbol{y}_i\}_{i=1}^{k}$ we have

$$\overline{D} = \sum_{i=1}^{k} \left(P_i \cdot \sum_{j=1}^{N} d\left(\boldsymbol{x}_j, \boldsymbol{y}_i\right) \cdot S_i\left(\boldsymbol{x}_j\right) \right)$$

where $P_i = \frac{1}{N} \sum_{j=1}^{N} s_i\left(\boldsymbol{x}_j\right)$.

4.4 Rate and Distortion Minimization

According to the web page *Data-Compression.com* [94], the main aim of data compression is to represent data obtained from an information source as accurately as possible using the fewest number of bits possible. Achieving this requires that both the distortion and code rate need to be minimized simultaneously.

In order to decrease the distortion of a quantized random vector or a quantized signal below a pre-specified level, a reproduction set $R = \{\boldsymbol{y}_i\}_{i=1}^{k}$ of size k, a quantizer rule Q, a distortion measure d and the set of cells $\{C_i\}_{i=0}^{k}$ must be chosen with a large enough k.

According to section 2.5.3, for the quantized random vector $\boldsymbol{Y}$, the expected code rate as a function of the probability mass function of the quantized random vector is bounded below by $H(\boldsymbol{Y}; \boldsymbol{\theta}, m)$.

According to the author [95], an increase in the size k of the reproduction set R will decrease (or leave unchanged) the combined distortion of a quantized random vector or quantized signal, if its support is bounded on $\mathbb{R}^m$. In most situations, an increase in the size of the reproduction point set will result in a decrease in overall distortion, however an increase will result in a larger code rate.

The constrained problem in rate distortion theory aims to find a lower bound for the code rate given for a particular distortion. Rate distortion theory is used for comparison purposes, it may however assist in choosing the reproduction point set in a few specialized cases.

4.4.1 Rate-distortion Theory

In section 2.11.2 it was shown that blocks of random variables of larger dimension m could be encoded at a lower code rate than their scalar counter-parts. In section 4.5 it will be shown that by quantizing blocks of random variables more efficient encoding and quantization could be performed. The increase in dimension, however leads to a greater distortion as the need for more reproduction points increases due to the increase in space volume.

Rate-distortion theory gives the theoretical bounds on what code rate can be achieved for a given loss of fidelity or what the minimum distortion is for a given code rate for encoded quantized vectors from a particular probability distribution [96].

---

[94] www.data-compression.com is a suggested site on the Stanford University Electrical Engineering site for Vector Quantization.

[95] According to the book by Gersho , A. and Gray, R. M. Vector Quantization and Signal Compression. Springer. 1992. Page 153. If the sample space is unbounded while the probability density in the overloaded sub-intervals is "small" and the probability density function is "smooth", then an increase in reproduction point set size k, will decrease the combined distortion (or leave it unchanged) that is if the combined distortion is finite.

[96] Wiegand, T. and Schwarz, H. Source Coding: Part I of Fundamentals of Source and Video Coding. Page 10.

The rate distortion function is the minimum code rate for all achievable code rates for a given distortion [97]. It is determined as a function of the increasing block size of independently identically distributed random variables constituting the random vector $\boldsymbol{X}$.

4.4.2 The Rate-Distortion Function

The distortion d between a random vector $\boldsymbol{X}$ of dimension m of independently identically distributed random variables and the respective quantized random vector $\boldsymbol{Y}$ obtained by means of a particular quantizer Q is calculated as the element-wise distortion as given by

$$d_m(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{m}\sum_{i=1}^{m} d(x_i, y_i).$$

The mean distortion is

$$D_m = \mathbb{E}_{\boldsymbol{X}}\big(d_m(\boldsymbol{X},\boldsymbol{Y})\big).$$

If the code rate of encoding the k reproduction points $\{\boldsymbol{y}_i\}_{i=1}^{k}$ is R, then the minimum of the rates for a given distortion D is given by the rate distortion function which is given by,

$$R(D) = \min_{\forall Q: D(Q)\leq D} I(\boldsymbol{X},\boldsymbol{Y}),$$

where $I(\boldsymbol{X},\boldsymbol{Y})$ is the mutual information between the input random vector $\boldsymbol{X}$ and the quantized random vector $\boldsymbol{Y} = Q(\boldsymbol{X})$, and the mutual information as defined in chapter two is given by

$$I(\boldsymbol{X},\boldsymbol{Y}) = \sum_{\boldsymbol{x}}\sum_{\boldsymbol{y}} f_{\boldsymbol{X},\boldsymbol{Y}}(\boldsymbol{x},\boldsymbol{y}) \cdot log_2\left(\frac{f_{\boldsymbol{X},\boldsymbol{Y}}(\boldsymbol{x},\boldsymbol{y})}{f_{\boldsymbol{X}}(\boldsymbol{x}) \cdot f_{\boldsymbol{Y}}(\boldsymbol{y})}\right).$$

The minimum is taken over every possible quantization Q.

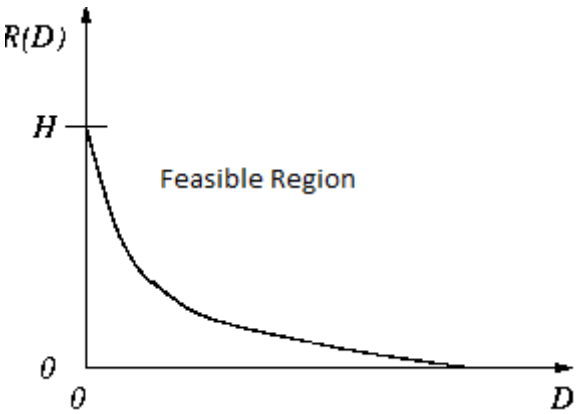


Figure 4.3 Rate-Distortion Function The minimum distortion D for a given rate R is determined over all possible quantization schemes $Q(\boldsymbol{X})$ for a particular code rate.

[97] Cover, T. M. and Thomas, J. A. Elements of Information Theory. Wiley Series in Telecommunications and Signal Processing. Second Edition. Page 306.

4.4.3 Performance of Scalar and Vector Quantizers and Encoders

According to the article [98], uniform scalar quantization of an independent source combined with an entropy encoder can achieve rate distortion values near the rate-distortion curve independent of the shape of the distribution.

When dependence structure is present, then vector quantizers have substantial gains over scalar quantizers and vector quantizers are able to achieve the rate distortion lower bound without entropy encoding [99].

4.4.4 Some Properties of $R(D)$

According to the power point presentation [100], the rate distortion function $R(D)$ is a continuous, monotone decreasing function of D and it is convex.

If $\boldsymbol{X}$ and $\left(\boldsymbol{X} - Q(\boldsymbol{X})\right)$ are statistically independent, then the lower bound can be achieved.

In section 3.6.4.4, it was shown that the Lloyd Max conditions with squared error distortion does not achieve independence between the input random vector $\boldsymbol{X}$ and the error $\left(\boldsymbol{X} - Q(\boldsymbol{X})\right)$. Therefore according to the author, although the Lloyd Max conditions are necessary and sufficient for an optimal quantizer in terms of minimizing the combined distortion, they are inadequate for the minimization rate distortion function.

4.4.5 The Unconstrained Problem

The problem of minimizing the code rate for a particular distortion can also be achieved by means of the unconstrained problem of minimizing the expected distortion, where the distortion is now a function of both the distortion and the uncertainty of $\boldsymbol{Y}$ with the penalty parameter λ. This was defined in the previous chapter as entropy constrained distortion, and was given by

$$d^*(\boldsymbol{x}, \boldsymbol{y}) = d(\boldsymbol{x}, \boldsymbol{y}) + \lambda \cdot \left(-\log_2\left(f_{\boldsymbol{Y}}(\boldsymbol{y})\right)\right).$$

One approach to minimizing the un-constrained problem is the iterative procedure known as the ECVQ algorithm, the algorithm will be discussed in section 4.7.1.

For an introduction to the unconstrained problem see the PDF [101].

---

[98] Lookabaugh, T.D. and Gray, R. M. High-Resolution Quantization Theory and the Vector Quantizer Advantage. IEEE Transactions on Information Theory, Vol. 35, no. 5, September 1989. Pages 1 and 3.

[99] Makhoul, J. Roucos, S. and Gish, H. Vector Quantization in Speech Coding. From the proceedings of the IEEE, vol. 73, no. 11, November 1985 , Section E, entitled Vector Quantization Model. Page 1566.

[100] Wiegand, Thomas: Digital Image Communication RD Theory and Quantization. Page 13.

[101] http://www.stanford.edu/class/ee368b/Handouts/04-RateDistortionTheory.pdf

4.5 Efficacy of the Vector Quantizer

This section looks at the increased effectiveness of vector quantization over scalar quantization and refers to results from section 2.11 with regards to vector encoding and scalar encoding.

4.5.1 Encoding Random Vectors

In section 2.11.1 it was shown that by increasing the block size or dimension m of a random vector, a decrease in the joint entropy of the random vector would result whenever there was a dependency relationship amongst the constituent elements of the vector.

It section 2.11.2 was also shown that a decrease in code rate would result from encoding a random vector as opposed to encoding its constituent elements and that this decrease would result even in the case where the constituent elements were statistically independent.

Therefore, as shown in section 2.11, encoding random vectors as opposed to the constituent random elements making up the random vector, will achieve a decrease in the code rate.

4.5.2 Quantizing Random Vectors as Opposed to Random Variables

According to the paper [[102]], the quantization of vectors, as opposed to scalars would take advantage of four properties that could lead to optimal quantization performance.

The four properties are linear dependencies, non-linear dependencies, shape of the distribution and dimensionality. Each will be discussed briefly and references will be made to various papers containing detailed discussions.

4.5.3 Linear Statistical Dependencies

The authors give a graphical illustration of a two dimensional correlated rectangular joint uniform probability density function is given.

It is argued that if scalar quantization is applied to each of the correlated random variables independently based on the appropriate marginal probability density functions and these quantized random variables are then jointly encoded, that reproduction points will be encoded that represent regions that lie outside the sample space of the joint probability density function.

The set of reproduction points and their related cells form a grid of size $I_1 \times I_2$ which cover a rectangular area, greater than the actual sample space.

[102] Makhoul, J. Roucos, S. and Gish, H. Vector Quantization in Speech Coding. From the proceedings of the IEEE, vol. 73, no. 11, November 1985, Section E, entitled Vector Quantization Model.

The authors argue that if vector quantization is applied to such a situation, that areas containing zero probability will not necessarily be assigned to reproduction points, thereby reducing the number of necessary reproduction points and the related code rate. This of course assumes that the same distortion will be incurred in both situations.

The increase in code rate due to assigning reproduction points to regions of zero probability (or very low probabilities) when scalar quantization is applied as opposed to vector quantization holds true for any joint probability distribution and is often exacerbated when correlation is present amongst the random variables.

4.5.4 Non-Linear Statistical Dependencies

When vector quantization is applied to random vectors, non-linear statistical dependencies within the source can be exploited. The authors use a similar illustration as in section 4.4.3, however, the rectangle is no longer slanted therefore showing that no correlation is present between the random variables.

The rectangle now contains a rectangular “hole” within it and a different spread in the probability mass function, which is used to represent some joint probability mass function whereby non-linear dependencies exist between the constituent random variables.

Once again, if scalar quantization is applied to each of the random variables independently based on the appropriate marginal probability density functions and these quantized random variables are then jointly encoded, then the reproduction points will be encoded that represent regions that lie on the “hole” in the joint probability density function which has zero probability of occurring.

The authors argue, once again that if vector quantization is applied to such a situation, that areas containing zero probability will not necessarily be assigned to reproduction points, which will reduce the related code rate.

Both the arguments in section 4.4.3 and 4.4.4 can be extended to higher dimensions, for different probability density functions.

4.5.5 Distribution Function Shape

Vector quantization is able to take advantage of the shape of the source probability density function, using cells of different shapes and sizes.

For example, if a particular probability density function has several modes, applying uniform scalar quantization to each vector element of the sample space will result in a rectangular grid of cells, however modal regions would require additional reproduction point vectors in order to reduce distortion for high probability outcomes, while allowing increased distortion in low probability areas, a rectangular grid would not achieve this goal.

In the paper [103] it was shown that though a high rate uniform scalar quantizer is optimal, it remains above the rate distortion function whereas a vector quantizer is able to achieve the rate distortion function, this is due to the inability of the scalar quantizer interval shape to cover the sample spaces with a high dimension without overlapping. In the case of entropy constrained optimization, however, the shape of the distribution function does not provide an advantage in efficiency of vector quantization over scalar quantizers [104].

4.5.6 Dimensionality

Vector quantization is more apt at dealing with quantization problems of dimension $(m > 1)$ than scalar quantization, because of the greater generality of the regions used for quantization and due to its space-filling ability.

In the paper [105], it is shown that by increasing the dimension from one to two, a rectangular grid can be replaced by a hexagonal lattice (which is an optimal polytope for $m = 2$ using the Euclidean distortion measure [106]), resulting in incremental efficiency due to the space filling capabilities of a hexagon over that of the rectangle.

In the paper [107], high rate vector quantizers are looked at theoretically in order to determine what increases in efficiency are attainable for increased dimension. These incremental increases in efficiency are considered for increasing dimensions where varying quantization methods, probability density function shapes and dependency relationships are used. Low rate vector and scalar quantizers and scalar and vector transformations are also examined.

Finally, according to the book [108], "Vector quantization is the "ultimate" solution to the quantization of a signal vector…it can at least match the performance of any arbitrary given coding system that operates on a vector of signal samples".

This all seems to show that vector quantization is far superior to scalar quantization – which it is, however in the next chapter it will be shown that scalar quantization combined with orthogonal transformations and entropy encoding achieves "good" compression at far less computational cost.

---

[103] Gish ,H. and Pierce, J. N. Asymptotically efficient quantizing. IEEE Trans. Inform. Theory, vol. IT-14, no. 5. Pages 676-683.
[104] Lookabaugh, T. D. and Gray, R. M. High-Resolution Quantization Theory and the Vector Quantizer Advantage. IEEE Transactions on Information Theory, vol. 35, no. 5, September 1989. Page 1027.
[105] Makhoul, J. Roucos, S. and Gish, H. Vector Quantization in Speech Coding. From the proceedings of the IEEE, vol. 73, no. 11, November 1985 , Section E, entitled Vector Quantization Model. Page 1560.
[106] Gersho, A. Asymptotically Optimal Block Quantization. IEEE Transactions on Information Theory, vol.IT-25, no.4, July 1979. Page 3.
[107] Lookabaugh, T. D. and Gray, R. M. High-Resolution Quantization Theory and the Vector Quantizer Advantage. IEEE Transactions on Information Theory, Vol. 35, no. 5, September 1989.
[108] Gersho, A. and Gray, R. M. Vector Quantization and Signal Compression. Springer. 1992. Page 313.

4.6 Generalized Lloyd Vector Quantization: The Optimal Problem

The vector optimal problem is minimizing the combined distortion given by

$$D = \min_{\{\mathcal{C}_i\},\{\boldsymbol{y}_i\}} \sum_{i=1}^{k} \int_{\mathcal{C}_i} d(\boldsymbol{x}, \boldsymbol{y}_i) \cdot f_{\boldsymbol{X}}(\boldsymbol{x}) d\boldsymbol{x}.$$

An increase in complexity should be considered when considering the vector valued form of the optimal problem discussed in the previous chapter. One example of this increase in complexity is that the intervals containing the reproduction points have now been extended to polytopal cells $\mathcal{C}_i$ with multiple boundary planes containing the reproduction vectors $\boldsymbol{y}_i$.

In general, there is no closed form for finding an optimal vector quantizer.

Solving the partial problems can be done in the same way as the previous chapter, a centroid and nearest neighbour condition can be obtained by either conditioning on the reproduction vectors or by conditioning on the i^{th} cell $\mathcal{C}_i$ [109].

The vector centroid condition is given by

$$\boldsymbol{y}_i = \mathbb{E}(\boldsymbol{X}|\boldsymbol{X} \in \mathcal{C}_i) \; \forall \; i = 1, 2, \dots, k.$$

This can calculated as

$$\boldsymbol{y}_i = \frac{\frac{1}{M}\sum_{j=1}^{M} \boldsymbol{x}_j \cdot S_i(\boldsymbol{x}_j)}{\frac{1}{M}\sum_{j=1}^{M} S_i(\boldsymbol{x}_j)}$$

with

$$S_i(\boldsymbol{x}_j) = \begin{cases} 1 & if \; \boldsymbol{x}_j \in \mathcal{C}_i \\ 0 & if \; \boldsymbol{x}_j \notin \mathcal{C}_i \end{cases}.$$

The vector nearest neighbour condition is given by

$$\mathcal{C}_i \subseteq \{\boldsymbol{x} : d(\boldsymbol{x}, \boldsymbol{y}_i) \leq d(\boldsymbol{x}, \boldsymbol{y}_q) \; for \; all \; q\}_{i=1}^{k}.$$

An allocation rule is necessary to assign the point uniquely to one cell in the case of equal distances.

Section 4.2.1.3, showed that polytopals cells are regular, in the next section 4.6.1, it will be shown that the nearest neighbour condition implies cell regularity.

[109] Gersho, A. and Gray, R. M. Vector Quantization and Signal Compression. Springer. 1992. Pages 350-354.

4.6.1 The Nearest Neighbour Condition implies Cell Regularity.

Let $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ be the reproduction points of the cells $\mathcal{C}_i$ and $\mathcal{C}_j$ respectively where

$$\mathcal{C}_i \subseteq \left\{ \boldsymbol{x} : \ d(\boldsymbol{x}, \boldsymbol{y}_i) \leq \ d\left(\boldsymbol{x}, \boldsymbol{y}_q\right), q = 1,2, \dots k \right\}_{i=1}^{k}.$$

The set of vectors $\left\{ \boldsymbol{x} : \ d(\boldsymbol{x}, \boldsymbol{y}_i) = d\left(\boldsymbol{x}, \boldsymbol{y}_j\right) \right\}$ are uniquely assigned to one of the cells $\mathcal{C}_i$ and $\mathcal{C}_j$ based on a prescribed allocation rule. If the distortion measure is the squared error distortion measure then the points will satisfy the linear equation

$$b = (\boldsymbol{m}^T) \cdot \boldsymbol{x}.$$

The set of points will constitute a hyper-plane in $\mathbb{R}^m$ that (possibly) [110] forms one of the boundaries of the cell C_i implying cell regularity by section 4.2.1.3.

Since we are given that

$$d(\boldsymbol{x}, \boldsymbol{y}_i) = \ d\left(\boldsymbol{x}, \boldsymbol{y}_j\right)$$

where the error squared distortion measure is used. This implies that

$$\| \boldsymbol{y}_i - \boldsymbol{x} \|^2 = \| \boldsymbol{y}_j - \boldsymbol{x} \|^2$$

$$\| \boldsymbol{y}_i \|^2 - 2\boldsymbol{y}_i^T \cdot \boldsymbol{x} + \| \boldsymbol{x} \|^2 = \| \boldsymbol{y}_j \|^2 - 2\boldsymbol{y}_j^T \cdot \boldsymbol{x} + \| \boldsymbol{x} \|^2$$

$$\| \boldsymbol{y}_i \|^2 - 2\boldsymbol{y}_i^T \cdot \boldsymbol{x} = \| \boldsymbol{y}_j \|^2 - 2\boldsymbol{y}_j^T \cdot \boldsymbol{x}$$

$$\| \boldsymbol{y}_i \|^2 - \| \boldsymbol{y}_j \|^2 = -2 \cdot \left(\boldsymbol{y}_j^T - \boldsymbol{y}_i^T\right) \cdot \boldsymbol{x}.$$

If we define the gradient vector $\boldsymbol{m}^T = -2 \cdot \left(\boldsymbol{y}_j^T - \boldsymbol{y}_i^T\right)$ and the intercept vector as $b = \| \boldsymbol{y}_i \|^2 - \| \boldsymbol{y}_j \|^2$, then we have the hyper-plane $b = (\boldsymbol{m}^T) \cdot \boldsymbol{x}$ as one of the boundaries of the cell $\mathcal{C}_i$. Since all the boundaries of each cell $\mathcal{C}_i$are determined by the nearest neighbour condition, then the boundaries are hyperplanes, which by the result in section 4.2.1.3 implies that the cells are convex and regular in the case the reproduction point $\boldsymbol{y}_i \in \mathcal{C}_i$.

[110] Not all centroids are necessarily used to determine the boundaries of a cell R_i reference Voronoi tessellation.

4.6.2 The Generalized Lloyd Algorithm (K-means Algorithm)

The Generalized Lloyd Algorithm is an extension of the Linde-Buzo-Gray Algorithm of scalar quantization. The Generalized Lloyd Algorithm is used to design a set of polytopal cells with a representative reproduction point set for data that is observed from a particular random vector source and is applicable for future vector quantization purposes.

For a continuous random vector $\boldsymbol{X}$ of dimension m with probability density function $f_{\boldsymbol{X}}$, let $\{\boldsymbol{x}_i\}_{i=1}^{M}$ be a set of M observed random vectors and $R = \{\boldsymbol{y}_i\}_{i=1}^{k}$ be the reproduction vector set.

4.6.2.1 Computing with the K-means Algorithm

1. Starting with initial reproduction points $\{\boldsymbol{y}_{t,i}\}_{i=1}^{k}$ for iteration $t = 1$.

2. Apply nearest neighbour partitioning using the set $\{\boldsymbol{y}_{t,i}\}_{i=1}^{k}$ using the nearest neighbour condition - Assign all observations $\{\boldsymbol{x}_j\}$ with the nearest reproduction point $\boldsymbol{y}_i$.

3. Apply the centroid condition to obtain the set of reproduction points $\{\boldsymbol{y}_{t+1,i}\}_{i=1}^{k}$, by finding the mean of all the observations $\{\boldsymbol{x}_j\}$ that have been assigned to the reproduction point $\boldsymbol{y}_i$.
   Determine the estimated combined distortion as
   $$D_{t+1} = \sum_{i=1}^{k}\left(P_{t+1,i} \cdot \sum_{j=1}^{M} d(\boldsymbol{x}_j, \boldsymbol{y}_{t+1,i}) \cdot S_{t+1,i}(\boldsymbol{x}_j)\right)$$
   where $P_{t+1,i} = \frac{1}{M}\sum_{j=1}^{M} S_{t+1,i}(\boldsymbol{x}_j)$.

4. Stop if a threshold is reached, else return to 2.

4.6.2.2 Problems with the Generalized Lloyd algorithm

The optimization process of the Generalized Lloyd algorithm is a descent algorithm that is very dependent on its initial conditions, which often causes suboptimal convergence.

- Sensitivity to Initial Conditions

Since the Generalized Lloyd algorithm is a descent algorithm, it is fast and converges to local minima in general, however, since the descent algorithm does not allow for an increase in average distortion the average distortion will not escape local minima, resulting in sub-optimum results. This is due to the shape and dimensionality of the distortion function, which is multimodal function in a high dimension, making it difficult to optimize.

- Dependence on Training Data

The model's dependence on training data is a general problem in data modelling, if the training data is poor, the model will be poor. This is even more applicable, if the model is unsupervised and the model is very general as is true for most vector quantization problems.

- The Number of Reproduction Vectors k

Deciding on the number of reproduction points k is often a problem in clustering, however, in vector quantization, the size of the codebook will be limited to the number of bits allocated to encode the quantized data set. The number of reproduction points k will often be predefined by the storage capacity or transmission capacity available.

4.7 Two Variations on the Generalized Lloyd Algorithm

Two variations of the Generalized Lloyd algorithm will be briefly described, their relationship with the Generalized Lloyd algorithm will be established and references will be given.

4.7.1 Entropy Constrained Vector Quantization: The Chou, Lookabaugh and Gray Algorithm

In section 4.5.5, the unconstrained rate distortion problem was stated. The paper [111] introduces a variation on the Generalized Lloyd algorithm named ECVQ or Entropy Constrained Vector Quantization.

The ECVQ algorithm is a descent algorithm that includes a weighted self-information factor that is added to the distortion measure, the new distortion is then minimized by means of an iterative minimization scheme (a descent algorithm) similar to that of the Generalized Lloyd algorithm. The minimization is locally optimal and can be used in conjunction with an entropy encoder like a Huffman encoder.

[111] See the paper by Chou, P. A. Lookabaugh, T. and Gray, R. M. Entropy-Constrained Vector Quantization. IEEE Trans. on Acoustics, Speech, and Signal Processing, Vol. 31, No.I. January 1989.

4.7.1.1 The Entropy Constrained Vector Quantization Algorithm

Given a random vector $\boldsymbol{X}$ with density function f_X generally unknown, let $\{\boldsymbol{x}_i\}_{i=1}^{M}$ be a set of M observed random vectors and $R = \{\boldsymbol{y}_i\}_{i=1}^{k}$ be the reproduction vector set.

1. Start with initial reproduction points $\{\boldsymbol{y}_{t,i}\}_{i=1}^{k}$ for iteration $t = 1$.

2. Apply biased [112] nearest neighbour partitioning using the set $\{\boldsymbol{y}_{t,i}\}_{i=1}^{k}$ by assigning $\boldsymbol{x}$ to the reproduction vector $\boldsymbol{y}_{t,i}$ by the rule, assign $\boldsymbol{x}$ to the reproduction vector $\boldsymbol{y}_{t,i}$ for a given λ, if
$$d(\boldsymbol{x}, \boldsymbol{y}_{t,i}) + \lambda \cdot \left(-\log_2\left(f_{\boldsymbol{Y}_t}(\boldsymbol{y}_{t,i})\right)\right) \leq d(\boldsymbol{x}, \boldsymbol{y}_{t,j}) + \lambda \cdot \left(-\log_2\left(f_{\boldsymbol{Y}_t}(\boldsymbol{y}_{t,j})\right)\right)$$
where $f_{\boldsymbol{Y}_t}(\boldsymbol{y}_{t,i})$ is estimated by $\frac{1}{M}\sum_{j=1}^{M} S_{t,i}(\boldsymbol{x}_j)$.

3. Apply the centroid condition to obtain the set of reproduction points $\{\boldsymbol{y}_{t+1,i}\}_{i=1}^{k}$ by finding the mean of all the observations $\{\boldsymbol{x}_j\}$ that have been assigned to the reproduction point $\boldsymbol{y}_i$.

4. Determine combined distortion as
$$\sum_{i=1}^{k}\left(P_{t+1,i} \cdot \sum_{j=1}^{M}\left(d(\boldsymbol{x}_j, \boldsymbol{y}_{t+1,i}) + \lambda \cdot \left(-\log_2(P_{t+1,i})\right)\right) \cdot S_{t+1,i}(\boldsymbol{x}_j)\right)$$
where $P_{t+1,i} = \frac{1}{M}\sum_{j=1}^{M} S_{t+1,i}(\boldsymbol{x}_j)$.

5. Stop if a threshold is reached, else return to 2.

If the parameter $\lambda = 0$, then ECVQ is just the K-means algorithm.

The initial values for $f_{\boldsymbol{Y}}(\boldsymbol{y}_i)$ are often determined from prior information or by applying K-means to identify the initial reproduction points $\{\boldsymbol{y}_i\}_{i=1}^{k}$and the respective estimated probabilities of $f_{\boldsymbol{Y}}(\boldsymbol{y}_i) = f_{\boldsymbol{X}}(\boldsymbol{x}|\boldsymbol{x} \in \mathcal{C}_i)$.

[112] de Garrido, D. P. and Pearlman, W. A. Conditional Entropy-Constrained Vector Quantization: High-Rate Theory and Design Algorithms. 2001. Page 4.

4.7.2 The Expectation Maximization Algorithm

The Expectation Maximization algorithm is described in the paper [113]. The EM algorithm is derived by maximizing the incomplete log likelihood of a mixture Gaussian distribution or the hidden Markov model. EM uses gradient decent iteration to find a locally optimal solution to the maximization problem.

Although the EM algorithm will not be discussed in detail here, it will be shown in section 4.7.3 that the EM algorithm can be considered as a more general form of the K-means algorithm, for a more general data structure. Please notice the notation $\{\boldsymbol{y}_{t,i}\}_{i=1}^{k}$ representing the reproduction point set before the t^{th} iteration is replaced by $\{\boldsymbol{\mu}_{t,i}\}_{i=1}^{k}$ due to the well-known $(\boldsymbol{\mu}, \Sigma)$ Gaussian notation for the mean and covariance matrix.

4.7.2.1 The Maximum Likelihood Problem

Let $\boldsymbol{X}$ be a random vector of dimension m with density function $f_{\boldsymbol{X}}$ where $f_{\boldsymbol{X}}$ is modelled on a mixture of multivariate Gaussian densities with k mixtures and let $\{\boldsymbol{x}_j\}_{j=1}^{M}$ be a set of observations. The conditional density function is given by

$$f_{\boldsymbol{X}}(\boldsymbol{x}_j;\ \boldsymbol{\mu}_i, \Sigma_i | \boldsymbol{x}_j \in \mathcal{C}_i) = \frac{exp\left(-\frac{1}{2}(\boldsymbol{x}_j - \boldsymbol{\mu}_i)^T\ \Sigma_i^{-1}(\boldsymbol{x}_j - \boldsymbol{\mu}_i)\right)}{(2\pi)^{\frac{m}{2}} \cdot |\Sigma_i|^{\frac{1}{2}}}.$$

The set of prior probabilities $\{P_i\}_{i=1}^{k}$ where

$$f_{\boldsymbol{X}}(\boldsymbol{x}_j \in \mathcal{C}_i) = P_i.$$

The maximum log-likelihood problem is given by

$$\max_{\boldsymbol{\mu},\Sigma}\left(\sum_{j=1}^{k}\sum_{i=1}^{M} P_{t,i} \cdot \left(-\frac{1}{2} \cdot ln(|\Sigma_i|) - \frac{1}{2}(\boldsymbol{x}_j - \boldsymbol{\mu}_i)^T\ \Sigma_i^{-1}(\boldsymbol{x}_j - \boldsymbol{\mu}_i)\right)\right)$$

with the constraint that $\sum_{i=1}^{k} P_i = 1$.

The maximum likelihood is determined over the set of mean vectors $\{\boldsymbol{\mu}_i\}_{i=1}^{k}$ and covariance matrices $\{\Sigma_i\}_{i=1}^{k}$ where the mean vectors represent the reproduction points in terms of the vector quantization problem.

1. Given the parameter set $R_t = \{\boldsymbol{\mu}_{t,i}, \Sigma_{t,i}, P_{t,i}\}_{i=1}^{k}$ for iteration $t = 1$.
2. Determine the parameter set $R_{t+1} = \{\boldsymbol{\mu}_{t+1,i}, \Sigma_{t+1,i}, P_{t+1,i}\}_{i=1}^{k}$ by computing the following

[113] Bilmes, J. A. A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models. 1998.

E-Step

$$M_{t,i,j} = \frac{P_{t,i} \cdot f_{\boldsymbol{X}}(\boldsymbol{x}_j \,;\, \boldsymbol{\mu}_{t,i}, \Sigma_{t,i})}{\sum_{i=1}^{k} P_{t,i} \cdot f_{\boldsymbol{X}}(\boldsymbol{x}_j \,;\, \boldsymbol{\mu}_{t,i}, \Sigma_{t,i})}.$$

$$P_{t+1,i} = \frac{1}{M} \sum_{j=1}^{M} M_{t,i,j}.$$

M-Step

$$\boldsymbol{\mu}_{t+1,i} = \sum_{j=1}^{M} \left(\frac{M_{t,i,j} \cdot \boldsymbol{x}_j}{\sum_{j=1}^{M} M_{t,i,j}} \right).$$

$$\Sigma_{t+1,i} = \sum_{j=1}^{M} \left(\frac{M_{t,i,j} \cdot (\boldsymbol{x}_j - \boldsymbol{\mu}_{t+1,i}) \cdot (\boldsymbol{x}_j - \boldsymbol{\mu}_{t+1,i})^T}{\sum_{j=1}^{M} M_{t,i,j}} \right).$$

The initial values for $\boldsymbol{\mu}, \Sigma$ and P are either determined due to prior information or by applying K-means to identify the set of initial means (reproduction points) $\{\boldsymbol{\mu}_i\}_{i=1}^{k}$ with $P = \frac{1}{k}$ and $\Sigma = I_m$ where I_m is the $m \times m$ identity matrix or by using other methods.

The use of K-means as an initialization strategy will become apparent in the next section.

4.7.3 The K-Means: Special Case of Expectation Maximization

The K-means algorithm is a specific case of the expectation maximization and can be used in the context of vector quantization when data is modelled using a mixture Gaussian distribution.

4.7.3.1 The Relationship between the EM Algorithm and K-Means [114]

By introducing the following constraints to the mixture Gaussian distribution

- Quantization (also known as hard-clustering)

$$f_X(\boldsymbol{x}_j) \propto \begin{cases} f_X(\boldsymbol{x}_j;\ \boldsymbol{\mu}_i, \Sigma_i | \boldsymbol{x}_j \in \mathcal{C}_i)\ if\ \boldsymbol{x}_j \in \mathcal{C}_i \\ 0 \qquad\qquad if\ \boldsymbol{x}_j \notin \mathcal{C}_i \end{cases}.$$

- Spherical density functions with equal variances for each cell $\mathcal{C}_i$

$$\Sigma_i = \sigma^2 \cdot I_m\ \forall i = 1, 2, \dots, k.$$

The conditional probability density function for C_i is then given by

$$f_X(\boldsymbol{x}_j;\ \boldsymbol{\mu}_i, \sigma^2 | \boldsymbol{x}_j \in C_i) = (2\pi)^{-\frac{m}{2}} \cdot exp\left(-\frac{||\boldsymbol{x}_j - \boldsymbol{\mu}_i||^2}{2\sigma^2}\right).$$

The unconditional density function can be calculated by

$$f_X(\boldsymbol{x}_j;\ \boldsymbol{\mu}_i, \sigma^2) \propto \sum_{i=1}^{k} f_X(\boldsymbol{x}_j;\ \boldsymbol{\mu}_i, \sigma^2 | \boldsymbol{x}_j \in C_i) \cdot f_X(\boldsymbol{x}_j \in C_i).$$

The likelihood function is then given by

$$L(\boldsymbol{\mu}, \Sigma | \{\boldsymbol{x}_j\}_{j=1}^{M}) \propto \prod_{j=1}^{M} \sum_{i=1}^{k} f_X(\boldsymbol{x}_j;\ \boldsymbol{\mu}_i, \sigma^2 | \boldsymbol{x}_j \in C_i) \cdot f_X(\boldsymbol{x}_j \in C_i).$$

Using quantization we have

$$f_X(\boldsymbol{x}_j | \boldsymbol{x}_j \notin C_i) = 0.$$

Leading to the reduction of the sum from

$$\sum_{i=1}^{k} f_X(\boldsymbol{x}_j;\ \boldsymbol{\mu}_i, \sigma^2 | \boldsymbol{x}_j \in C_i) \cdot f_X(\boldsymbol{x}_j \in C_i)$$

[114] Based on the helpful power point presentation by Guestrin, C. Machine Learning. Carnegie Mellon University. 2007.

to

$$f_X(\boldsymbol{x}_j;\ \boldsymbol{\mu}_i, \sigma^2 | \boldsymbol{x}_j \in C_i) \cdot f_X(\boldsymbol{x}_j \in C_i).$$

Substituting the above expressions into the likelihood function we get

$$L(\boldsymbol{\mu}, \Sigma | \{\boldsymbol{x}_j\}_{j=1}^{M}) \propto \prod_{j=1}^{M} P_i \cdot (2\pi)^{-\frac{m}{2}} \cdot exp\left(-\frac{||\boldsymbol{x}_j - \boldsymbol{\mu}_i||^2}{2\sigma^2}\right).$$

By taking the natural logarithm we get the log-likelihood function

$$ln(L(\boldsymbol{\mu}, \Sigma | \{\boldsymbol{x}_j\}_{j=1}^{N})) \propto \sum_{j=1}^{M} ln(P_i) - \frac{m}{2} \cdot ln(2\pi)\ - \frac{||\boldsymbol{x}_j - \boldsymbol{\mu}_i||^2}{2\sigma^2}.$$

The maximum likelihood in terms of $\boldsymbol{\mu}_i$ can be determined by fixing the set of prior probabilities P_i and maximizing the above expression by minimizing the sum

$$\max_{\boldsymbol{\mu}_i} ln(L(\boldsymbol{\mu}, \Sigma | \{\boldsymbol{x}_j\}_{j=1}^{M})) = \min_{\boldsymbol{\mu}} \sum_{j=1}^{M} ||\boldsymbol{x}_j - \boldsymbol{\mu}_i||^2$$

If $\boldsymbol{\mu}_i$ is replaced by the reproduction point $\boldsymbol{y}_i$, then the expectation maximization applied to find the reproduction point set $\{\boldsymbol{y}_i\}_{i=1}^{k}$ is equivalent to the Generalized Lloyd algorithm.

4.8 Conclusions of Chapter Four

- Vector quantization is the natural extension of scalar quantization to $\mathbb{R}^m$. This extension results in an increase in power and complexity.

- Vector quantization is applied to a random vector $\boldsymbol{X} \in \mathbb{R}^m$ where the vector $\boldsymbol{X}$ is quantized as a single unit. The quantization is determined by a set of non-overlapping, regular polytopes $\{\mathcal{C}_i\}_{i=1}^{k}$ known as cells.

- In the case where a vector quantizer has already been designed, then the process of vector quantization can be decomposed into a set of linear function combined with logical operators.

- The concept of distortion can be extended to vector quantizers.

- Rate-distortion theory attempts to find the theoretical bounds on the amount of compression achievable for a given combined distortion.

- The quantization of vectors as opposed to scalars takes advantage of four properties, namely linear dependencies, non-linear dependencies, the density function shape and dimensionality. Vector quantization is shown to be substantially more powerful than scalar quantization, in particular when structural dependency is present.

- The Generalized Lloyd Algorithm or K-Means algorithm is the result of the minimization of the vector optimal problem of minimizing the combined distortion for the Euclidian distortion measure.

- The sub-optimality of the Generalized Lloyd is considered with respect to the weakness of the greedy algorithm, sensitivity to initial values, the determination of the reproduction set size and it's susceptibility to poor training data.

- The ECVQ algorithm is a descent algorithm, similar to that of the Generalized Lloyd algorithm, with a distortion measure, that includes a penalty parameter that punishes increased uncertainty of the quantized random variable. The ECVQ algorithm reduces to the Generalized Lloyd algorithm when the penalty parameter is equal to zero.

- The EM algorithm is descent algorithm that maximizes the incomplete log-likelihood function in two steps, namely the expectation and maximization steps. The Generalized Lloyd algorithm is a special case of the EM algorithm.

Chapter Five: Scalar Transformations and Picture Study

In the previous chapter, it has been argued that vector quantization is substantially more powerful than scalar quantization in quantizing data, in that it is able to reduce the combined distortion for a given code rate below that of scalar quantization. In practice (image, sound and video encoding) vector quantization is seldom used.

Three reasons will be given for this, namely computational complexity, alternative bit allocation strategies and the availability of techniques that reduce gains in of vector quantization over that of scalar quantization due to distribution shape and linear dependence.

Bit allocation provides a scalar quantization alternative to quantizing and encoding vectors, while linear transformations combined with entropy encoding reduce linear dependence, achieve more effective bit allocations and encode higher dimensional data more efficiently.

5.1 Complexity

In general vector quantization is far more computationally complex and time consuming than scalar quantization. Vector Quantization is often too slow for use “on-the-run” compression for image, audio and video data compression.

In the case where uniform scalar quantization is combined with procedures like image space transforms [115] and unitary linear transforms in order to compress data efficiently with as little perceived distortion as required, these procedures combined with uniform scalar quantization are still substantially less time consuming and complex than even simple vector quantization techniques.

5.2 Bit Allocation

The bit allocation problem is discussed in the book [116] it will be considered briefly in this section.

Consider the random vector $\boldsymbol{X} = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_m \end{pmatrix}$ of dimension m with constituent random variable elements X_i each with a known marginal density function f_{X_i} with $\mathbb{E}(X_i) = 0$ and $var(X_i) = \sigma_i$.

[115] A particular image space transform is discussed in section 5.2. For a detailed discussion on image space transforms see Pratt , William K. Digital Image Processing. Fourth Edition. Chapters 3-4.
[116] Gersho, A. and Gray, R. M. Vector Quantization and Signal Compression. Springer. 1992. Pages 226 -234.

If scalar quantization is applied to each random variable X_i, how then can bits be assigned to each quantized random variable $Y_i = Q(X_i)$ in order to minimize the combined distortion if the overall code rate is constrained by the value R^*.

Since the overall code rate $\sum_{i=1}^{m} r_i$ incurred by encoding each random variable X_i is dependent on the encoding scheme and on the number of reproduction points k_i available for encoding X_i, then the combined distortion incurred by quantizing X_i with Y_i is a function of the code rate r_i.

This relationship is given by $D_i = W(r_i)$ where D_i is the combined distortion incurred by quantizing X_i with Y_i.

The bit allocation problem can then be worded as,

find the set of code rates $\{r_i\}_{i=1}^{m}$ to satisfy the inequality, where

$$\sum_{i=1}^{m} r_i \leq R^*$$

with the code rate r_i incurred by encoding the i^{th} element Y_i. In order to minimize the combined distortion $W(r_i)$ incurred by quantizing X_i with Y_i summed over each element of the random vector $\boldsymbol{X}$.

That is find a set $\{r_i\}_{i=1}^{k}$ of code rates in order to minimize $D = \sum_{i=1}^{m} W(r_i)$ [117].

The bit allocation problem will not be assessed in this mini dissertation, it does however present another template to compress data, that is, to reduce the code rate below some threshold by quantizing data in a selectively "smart" way [118].

In the book, [119] it is shown that under certain assumptions, the individual codebook sizes should be directly proportional to the variance of that random variable.

Since the distortion incurred by quantizing individual random variables is additive, and since the amount of distortion incurred for each random variable is a function of its variance and reproduction sets size, random variables with greater variance would require larger reproduction sets, whilst random variables with smaller variance would then require smaller reproduction sets in order to achieve similar overall distortion levels.

[117] An optimal scalar quantizer $Q_i(X_i)$ has a reproduction point set of size k_i that minimizes the overall distortion this quantizer will satisfy the Lloyd Max conditions, however will not necessarily be determined by the Lloyd Max algorithm (or LBG algorithm).
[118] Gersho, A. and Gray, R. M. Vector Quantization And Signal Compression. Springer. 1992. Page 229.
[119] Gersho, A. and Gray, R. M. Vector Quantization And Signal Compression. Springer. 1992. Pages 229 and 231.

In chapter one, the notion of selectively assigning longer codewords to less probable outcomes and shorter codewords to more probable outcomes was introduced, similarly ordering random variables in an order of importance would allow for a greater number of bits to be assigned to random variables containing more information and fewer bits to those containing less information.

5.3 Techniques that Increase Scalar Quantization Gains

- Entropy Encoding (like Huffman encoding, Arithmetic encoding [120], etc.)

In the paper [121] the space filling, dependence and shape advantages of using vector quantization compared to scalar quantization are calculated using asymptotic assumptions and sphere packing bounds.

The question of using a high rate scalar quantizer with entropy encoding as opposed to vector quantization is addressed. The following is concluded.

In the case where the random vector is made up of independently identically distributed random variables, if the uncertainty is constrained, then it is shown that due to the independence assumption that the dependence advantage and shape gains are absent and the gains in space filling achieved by vector quantization over scalar quantization are bounded above.

However, in the case where dependency structure is present amongst the random variables, then entropy encoding combined with scalar quantization does not achieve the efficacy of vector quantization.

- Decorrelation

If the correlation amongst the random variables constituting the random vector could be reduced or eliminated and “selective” scalar quantizers could be applied to the de-correlated transformed variables, then the gains of vector quantization and encoding over that of scalar quantization and encoding will be reduced.

As discussed in section 4.4.5 although vector quantization can take advantage of the shape of the distribution, in the case of entropy constrained optimization, the shape of the distribution does not provide an advantage in efficiency for vector quantization over scalar quantization[122].

[120] Only Huffman encoding is discussed in this mini dissertation, but many other entropy encoders exist, an excellent introduction can be found in the book Solomon, D. and Motta, G. Handbook of Data Compression, Springer. 2010.
[121] Lookabaugh,T. D. and Gray, R. M. High-Resolution Quantization Theory and the Vector Quantizer Advantage. IEEE Transactions on Information Theory, Vol. 35, no. 5, September 1989. Page 1025.
[122] Lookabaugh, T. D. and Gray, R. M. High-Resolution Quantization Theory and the Vector Quantizer Advantage. IEEE Transactions on Information Theory, Vol. 35, no. 5, September 1989. Page 1027.

If linear transformations are applied to the data in order to remove most of the linear dependence and allow for selective bit allocation, then scalar quantization and entropy encoding can be applied to achieve compression results that are possibly near that of vector quantization, with a smaller combined distortion.

5.4 Linear Transformations

Linear transforms will be discussed as a method to de-correlate data and allow for efficient bit allocation.

5.4.1 Data Processing Inequality, Linear Transformations and Quantization

When applying linear transforms to a random vector, does the amount of information change and what about quantization?

According [123] if $\boldsymbol{X}$ and $\boldsymbol{Z}$ are any two discrete random vectors with mutual information $I(\boldsymbol{X}, \boldsymbol{Z})$, if $\boldsymbol{Y}$ is a random vector obtained by an operation on $\boldsymbol{Z}$ alone, then

$$I(\boldsymbol{X}, \boldsymbol{Y}) \leq I(\boldsymbol{X}, \boldsymbol{Z}).$$

And if $I(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{Y}) = 0$ then $\boldsymbol{Y}$, which is a function of the raw data $\boldsymbol{Z}$, is a sufficient statistic of $\boldsymbol{X}$**.** In other words, a reduction in mutual information content will occur unless a sufficient statistic is used to represent the raw data $\boldsymbol{Z}$**.**

According to the textbook [124], if $t(\theta)$ is a sufficient statistic for θ, then any one-to-one function of $t(\theta)$ is also sufficient.

A linear transform $T(\boldsymbol{Z})$ which is a one-to-one function of the raw data, will not result in a decrease in mutual information.

A non-linear transform, such as quantization will lead to a reduction in mutual information content.

5.4.2 Decomposing a Random Vector into its General Spectrum and Applying Bit Allocation

One effective and very common approach is to transform a random vector into a set of transformed coefficients by decomposing it into its generalized spectrum.

The resulting spectrum will be composed of the transform coefficients and their related orthogonal bases functions or vectors.

Each spectral component in the transform domain (transform coefficients) represents the amount of energy of the specific spectral function in the original image [125].

---

[123] Golomb, S. W., Peile, R. E. and Sholtz, R. A. Basic Concepts in Information Theory: The Adventures of Secret Agent 00111. Springer. 1994. The Data Processing Theorem (Theorem 6.1). Page 313.
[124] Biswas, S. and Srivastav, G.L. Mathematical Statistics, A Textbook. Alpha Science International. 2011. Theorem 13.4. Page 13.10.

The transformed coefficients are much less correlated (at best orthogonal) and often a smaller subset of the coefficients can account for most of the variability of process. There is almost as much information in the smaller subset of transformed coefficients as there is in the original coefficients, so the information can be compacted.

A greater number of reproduction points may be assigned to the random variable with larger transform coefficients since these coefficients represent a greater amount of energy or variation within the random variable. This bit allocation strategy allows a substantial reduction in code rate for a small increase in distortion.

5.4.3 The Unitary Linear Transformation

In images, it is more efficient to represent a picture by uncorrelated data, than in a correlated form. If the data can then be ranked, from most information content and picture quality, to the least, then less important data can be removed (or quantized at a lower rate).

The unitary linear transform is a linear transformation in the form of a unitary transform matrix applied to a vector of random variables.

5.4.4 Unitary Transformation Matrix

According to the book, [[126]] a square matrix $\boldsymbol{\mathcal{T}}$ of dimension m with complex entries is unitary if its column vectors are orthonormal, i.e. $\boldsymbol{\mathcal{T}}^{*}\boldsymbol{\mathcal{T}} = \boldsymbol{I}$ where $\boldsymbol{\mathcal{T}}^{*}$ is the conjugate transpose of $\boldsymbol{\mathcal{T}}$.

Several important properties of the unitary matrix are listed below

1. The unitary transform matrix is invertible $\boldsymbol{\mathcal{T}}^{-1} = (\boldsymbol{\mathcal{T}}^{*})^{T}$
2. The unitary transform matrix $\boldsymbol{\mathcal{T}}$ preserves measure, $||\boldsymbol{\mathcal{T}}\boldsymbol{X}|| = ||\boldsymbol{X}||$
3. The quantized transformed coefficients will have the same distortion as their un-transformed counterparts.

When considering the mean squared error distortion measure, statement two implies statement three. This will be shown next.

Proof

Consider the random vector $\boldsymbol{X}$, the quantized random vector $\boldsymbol{Y}$ and the unitary transformation matrix $\boldsymbol{\mathcal{T}}$. The combined distortion D is calculated as

$$D = \mathbb{E}(||\boldsymbol{X} - \boldsymbol{Y}||^{2})$$

[125] Pratt, W. K. Digital Image Processing. Fourth Edition. Wiley. 2007. Page 192.
[126] Beauregard, F. Linear Algebra. Third Edition. Addison Wesley Longman. 1995. Page 470.

Let $\widetilde{\boldsymbol{X}} = \boldsymbol{\mathcal{T}}\boldsymbol{X}$ be the transformed random vector and $\widetilde{\boldsymbol{Y}} = \boldsymbol{\mathcal{T}}\big(Q(\boldsymbol{X})\big)$ be the transformed, quantized random vector, then consider the difference

$$\begin{aligned} &||\widetilde{\boldsymbol{X}} - \widetilde{\boldsymbol{Y}}||^2 \\ &= ||\boldsymbol{\mathcal{T}}\boldsymbol{X} - \boldsymbol{\mathcal{T}}\boldsymbol{Y}||^2 \\ &= ||\boldsymbol{\mathcal{T}}\,||^2 \cdot ||\boldsymbol{X} - \boldsymbol{Y}||^2 \end{aligned}$$

But $\boldsymbol{\mathcal{T}}^t \cdot \boldsymbol{\mathcal{T}} = \boldsymbol{I}$.

$$= \ ||\boldsymbol{X} - \boldsymbol{Y}||^2.$$

Therefore,

$$\mathbb{E}\big(||\widetilde{\boldsymbol{X}} - \widetilde{\boldsymbol{Y}}||^2\big) = \ \mathbb{E}(||\boldsymbol{X} - \boldsymbol{Y}||^2).$$

The overall distortion of the transformed coefficients is equal to the overall distortion of their untransformed counterparts as stated in property three.

The transformation that removes (most) of the correlation from the data and does not increase the overall distortion is a rotation. This can be achieved with an orthogonal transformation, that is a real unitary transformation.

Two such orthogonal transformations are the Discrete Karhunen-Loeve Transform, and the Discrete Cosine Transform.

5.4.5 The Discrete Karhunen-Loeve Transform [127]

The discrete Karhunen-Loeve transform (better known as principal component analysis) is an orthogonal transform applied to the statistical nature of the data. It is data dependent and based on the minimization of the mean squared error overall distortion..

The transformation is achieved as follows,

If $\boldsymbol{X}$ is an $m \times 1$ random vector $\boldsymbol{X} = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_m \end{pmatrix}$ with a mean vector $E(\boldsymbol{X}) = \boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_m \end{pmatrix}$

and $m \times m$ covariance matrix $\Sigma = \mathbb{E}[(\boldsymbol{X} - \boldsymbol{\mu}) \cdot (\boldsymbol{X} - \boldsymbol{\mu})^T]$ with eigenvector matrix $\boldsymbol{\mathcal{E}} = \begin{pmatrix} -\boldsymbol{e}_1^T- \\ -\boldsymbol{e}_2^T- \\ \vdots \\ -\boldsymbol{e}_m^T- \end{pmatrix}$ of the covariance matrix Σ and the eigenvalue and normalized eigenvector pairs $(\lambda_1, \boldsymbol{e}_1), (\lambda_2, \boldsymbol{e}_2), \dots, (\lambda_m, \boldsymbol{e}_m)$ with $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m \geq 0$.

[127] Johnson, R. and Wichern, D. Applied Multivariate Statistical Analysis. Pearson Prentice Hall. 2007. Pages 430-433.

The Karhunen-Loeve Transform of the random vector $\boldsymbol{X}$ is given by

$$\boldsymbol{Y} = \boldsymbol{\mathcal{E}}^T \cdot \boldsymbol{X}.$$

And the inverse transformation is given by

$$\boldsymbol{X} = \boldsymbol{\mathcal{E}} \cdot \boldsymbol{Y}.$$

The transformed coefficient is given by

$$\tilde{X}_i = \boldsymbol{e}_i \cdot \boldsymbol{X}.$$

With covariance matrix given by

$$cov(\tilde{X}_a, \tilde{X}_b) = \boldsymbol{e}_a^t \cdot \Sigma \cdot \boldsymbol{e}_b = \begin{cases} \lambda_j \,;\, a = b \\ 0 \,;\, a \neq b \end{cases}.$$

It can be shown that the Karhunen-Loeve Transform is the best possible transform for minimizing overall distortion, for a given bit allocation [128].

The Karhunen-Loeve Transform is the optimal orthogonal transform with minimal mean square error for a given bit allocation, but due to having to obtain the eigenvalues and eigenvectors of the covariance matrices that may be arbitrarily large, it is computationally expensive to implement [129].

An alternative is the Discrete Fourier Transform and in particular, the Discrete Cosine Transform, which will be discussed in the next three sections.

5.4.6 The Discrete Two Dimensional Fourier Transform

In certain situations, a random matrix $\mathcal{X}$ of size $(m \times m)$ will be considered, with the u^{th} row and v^{th} column entry $\mathcal{X}_{u,v}$ a random variable with density function $f_{\mathcal{X}_{u,v}}$[130].

The discrete two dimensional Fourier transform of an indexed random variable $X_{u,v}$ is

$$\mathcal{Y}_{a,b} = \frac{1}{m} \cdot \sum_{u=0}^{m-1} \sum_{v=0}^{m-1} \mathcal{X}_{u,v} \cdot e^{-\frac{(a \cdot u + b \cdot v) \cdot 2 \cdot \pi \cdot i}{m}} ; i = \sqrt{-1}.$$

The inverse transform is given by

[128] Gersho, A. and Gray, R. M. Vector Quantization And Signal Compression. Springer. 1992. Page 241.
[129] Solomon, D. and Motta, G. Handbook of Data Compression. Springer. 2010. Page 475.
[130] The use of a random matrix will become apparent later on in the chapter when looking at image data, where the statistical dependence structure is contained in the row –column matrix structure.

$$\mathcal{X}_{u,v} = \frac{1}{m} \cdot \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} \mathcal{Y}_{a,b} \cdot e^{\frac{(a \cdot u + b \cdot v) \cdot 2 \cdot \pi \cdot i}{m}}; i = \sqrt{-1}.$$

Alternatively, the transform matrix can be used, where

$$\boldsymbol{\mathcal{T}} = \frac{1}{\sqrt{k}} \cdot \begin{bmatrix} W^0 & W^0 & W^0 & \cdots & W^0 \\ W^0 & W^1 & W^2 & \cdots & W^{m-1} \\ W^0 & W^2 & W^4 & \vdots & W^{2 \cdot (m-1)} \\ \vdots & \vdots & \cdots & \ddots & \vdots \\ W^0 & W^{m-1} & W^{2 \cdot (m-1)} & \cdots & W^{(m-1)^2} \end{bmatrix}$$

with $W = e^{-\frac{2 \cdot \pi \cdot i}{m}}$ where $e^{x \cdot 2 \cdot \pi \cdot i} = \cos(2 \cdot \pi \cdot x) + i \cdot \sin(2 \cdot \pi \cdot x)$.

The transformation of the random matrix $\mathcal{X}$ of size $(m \times m)$ can be achieved by means of matrix multiplication

$$\mathcal{Y} = \boldsymbol{\mathcal{T}} \mathcal{X} \boldsymbol{\mathcal{T}}.$$

The inverse transformation of the random matrix $\boldsymbol{Y}$ of size $(m \times m)$ can be achieved by means of matrix multiplication by the complex conjugate

$$\mathcal{X} = \boldsymbol{\mathcal{T}}^* \mathcal{Y} \boldsymbol{\mathcal{T}}^*.$$

The discrete Fourier transform is independent of the distribution of the random matrix, it is however dependent only on the size $(m \times m)$ of the random matrix.

The discrete cosine transformation is a special case of the discrete Fourier transform.

5.4.7 The Discrete Cosine Transform from a Discrete Fourier Transform

The Fourier transform of a real, symmetric matrix is real [131]. In the case of image pixels, the matrix $\mathcal{X}$ is real, but not symmetric.

According to the book [132] by applying a reflection of the real matrix values in $\mathcal{X}$ over each of its edges see Figure 5.1, we obtain a symmetric $(2 \cdot m) \times (2 \cdot m)$ matrix $\mathcal{X}^{s}_{u,v}$.

This reflection can be achieved by defining the $(2 \cdot m) \times (2 \cdot m)$ matrix

$$\mathcal{X}^{s}_{u,v} = \begin{cases} \mathcal{X}_{u,v} & u \geq 0, v \geq 0 \\ \mathcal{X}_{-1-u,v} & u < 0, v \geq 0 \\ \mathcal{X}_{u,-1-v} & u \geq 0, v < 0 \\ \mathcal{X}_{-1-u,-1-v} & u < 0, v < 0 \end{cases}$$

[131] Pratt, W. K. Digital Image Processing. Fourth Edition. Wiley. 2007, Pages 195-196.
[132] Pratt, W. K. Digital Image Processing. Fourth Edition. Wiley. 2007. Page 196.

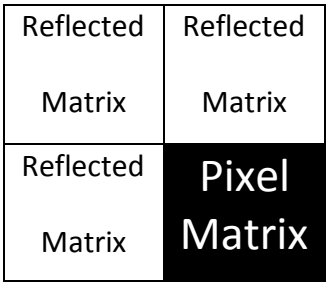


Figure 5.1 Reflection of Image over Each of its Edges The pixel matrix is made symmetric by reflecting it over the x-axis, y-axis and origin.

The Fourier transform of this real, symmetric $(2 \cdot m) \times (2 \cdot m)$ matrix is

$$\mathcal{Y}^{s}_{a,b} = \frac{1}{2 \cdot m} \cdot \sum_{u=-m}^{m-1} \sum_{v=-m}^{m-1} \mathcal{X}^{s}_{u,v} \cdot e^{-\frac{\left(a \cdot \left(u+\frac{1}{2}\right)+b \cdot \left(v+\frac{1}{2}\right)\right) \cdot 2 \cdot \pi \cdot i}{2 \cdot m}}.$$

This however, can be simplified, the redundancy is discarded and the summation is normalized into the regular discrete cosine transform, which is represented by the discrete cosine transform matrix $\boldsymbol{\mathcal{T}}$.

5.4.8 The Discrete Cosine Transform Matrix

The DCT (also known as DCT-2) matrix $\boldsymbol{\mathcal{T}}$ of size $m \times m$ where the i^{th} row and j^{th} column entry is given by

$$\boldsymbol{\mathcal{T}}_{i,j} = \begin{cases} \sqrt{\frac{2}{m}} \cdot \cos\left(\frac{\pi}{m} \cdot i \cdot \left(j+\frac{1}{2}\right)\right) & \text{for } i = 1,2,\dots,m-1 \text{ and } j = 0,1,2,\dots,m-1 \\ \sqrt{\frac{1}{m}} \cdot \cos\left(\frac{\pi}{m} \cdot i \cdot \left(j+\frac{1}{2}\right)\right) & \text{for } i = 0 \text{ and } j = 0,1,2,\dots,m-1 \end{cases}$$

The application of the $m \times m$ transform matrix to the matrix $\mathcal{X}$ is shown next.

5.4.9 DCT Transformation of $\mathcal{X}$ [133]

The transformation of the random matrix $\mathcal{X}$ of size $(m \times m)$ with $\mathcal{Y} = \boldsymbol{\mathcal{T}} \cdot \mathcal{X} \cdot \boldsymbol{\mathcal{T}}^{T}$ is given by

$$\mathcal{Y}_{i,j} = \frac{2}{m} \cdot c(i) \cdot c(j) \cdot \sum_{u=0}^{m-1} \sum_{v=0}^{m-1} \mathcal{X}_{u,v} \cdot \cos\left(\frac{(2 \cdot u+1) \cdot i \cdot \pi}{2 \cdot m}\right) \cdot \cos\left(\frac{(2 \cdot v+1) \cdot j \cdot \pi}{2 \cdot m}\right)$$

with $c(a) = \begin{cases} \frac{1}{\sqrt{2}} & for\ a = 0 \\ 1 & for\ a = 1,2,\dots,m-1. \end{cases}$

And the inverse transform $\boldsymbol{\mathcal{T}}^{-1}$ of size $m \times m$, then $\mathcal{X} = \boldsymbol{\mathcal{T}}^{T} \cdot \mathcal{Y} \cdot \boldsymbol{\mathcal{T}}$ is given by

[133] http://www.mathworks.com/help/toolbox/images/f21-16366.html.

$$\mathcal{X}_{i,j} = \frac{2}{m} \cdot \sum_{u=0}^{m-1} \sum_{v=0}^{m-1} c(u) \cdot c(v) \cdot Y_{u,v} \cdot \cos\left(\frac{(2 \cdot u + 1) \cdot i \cdot \pi}{2 \cdot m}\right) \cdot \cos\left(\frac{(2 \cdot v + 1) \cdot j \cdot \pi}{2 \cdot m}\right).$$

The discrete cosine transform is more computationally efficient than Karhunen-Loeve transform in terms of computational time and ease of use and since discrete cosine transform matrix is determined independently of the random vector.

If the random variables constituting the random matrix $\mathcal{X}$ are highly correlated, then the discrete cosine transform coefficients are "near" to the optimal coefficients of the Karhunen-Loeve transform. The relationship between the discrete cosine transform and the Karhunen-Loeve transform as applied to a Markov chain of order one will be discussed next.

5.4.10 Discrete Karhunen-Loeve Transform to the Discrete Cosine Transform

In the paper [134] the relationship between discrete Karhunen-Loeve transform and discrete cosine transform is derived.

Consider the random variable ε_i with $E(\varepsilon_i) = 0$ and $var(\varepsilon_i) = \sigma_\varepsilon^2$.

Given X_i a stationary random process of order 1 where $X_i = constant + p \cdot X_{i-1} + \varepsilon_i$ and $|p| < 1$.

The correlation matrix $Corr(X_i, X_j)$ is given by the Toeplitz matrix [135]

$$Corr(X_i, X_j) = \begin{bmatrix} 1 & p & p^2 & \dots & p^{m-1} \\ p & 1 & p & \ddots & \vdots \\ p^2 & p & 1 & \ddots & p^2 \\ \vdots & \ddots & \ddots & \ddots & p \\ p^{m-1} & \dots & p^2 & p & 1 \end{bmatrix}.$$

The eigenvalues λ_i where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_k \geq 0$ are given by

$$\lambda_i = \frac{(1 - p^2)}{1 - 2 \cdot p \cdot \cos(\omega_i) + p^2} \text{ for } i = 1, 2, \dots, m.$$

where ω_i are the real positive roots of the equation:

$$\tan(m \cdot \omega) = \frac{-(1 - p^2) \cdot \sin(\omega)}{\cos(\omega) - 2 \cdot p + p^2 \cdot \cos(\omega)}.$$

[134] Ray, W.D. and Driver, R.M. Futher decomposition of the Karhunen-Loeve series representation of a stationary random process. IEEE Trans. 1970, IT-16. Pages 845-850.
[135] Wang, R. Introduction to Orthogonal Transforms with Applications in Data Processing and Analysis. Cambridge. 2012. Page 554.

The Karhunen-Loeve transform $\boldsymbol{\mathcal{E}}$ matrix is given by

$$\boldsymbol{\mathcal{E}}_{i,j} = \sqrt{\frac{2}{m+\lambda_i}} \cdot \sin\left(\omega_i \cdot \left(j - \frac{(m+1)\cdot\pi}{2}\right) + \frac{i\cdot\pi}{2}\right) \quad i,j = 1,2,\dots,m.$$

It has be shown that as $p \to 1$ then $\lambda_0 \to m$ and $\lambda_i \to 0\ \forall\ i \neq 0$ then $\boldsymbol{\mathcal{E}}_{i,j}$ becomes

$$\boldsymbol{\mathcal{E}}_{i,j} = \begin{cases} \sqrt{\frac{2}{m}} \cdot \cos\left(\frac{\pi}{m} \cdot i \cdot \left(j + \frac{1}{2}\right)\right) \text{ for } i = 1,2,\dots,m-1 \text{ and } j = 0,1,2,\dots,m-1 \\ \sqrt{\frac{1}{m}} \cdot \cos\left(\frac{\pi}{m} \cdot i \cdot \left(j + \frac{1}{2}\right)\right) \text{ for } i = 0 \text{ and } j = 0,1,2,\dots,m-1 \end{cases}$$

The transform matrix $\boldsymbol{\mathcal{E}}_{i,j}$ is just the discrete cosine transform matrix described in the previous section.

5.5 Picture Study

In this section, the compression of a digital image is studied in the context of the rest of the mini dissertation.

A small amount of detail is included about RGB images, linear transformations applied to RGB images and the distribution of a transformed image's coefficients.

Firstly, a brief introduction to the analog and digital image is necessary.

5.5.1 Analog Image

An (analog) image is a spatial energy function f applied to a 2-dimensional spatial distribution of Cartesian coordinates (x, y) and the wavelength λ [136].

$$f(x, y, \lambda) \in \mathbb{R}^+ / \{0\}$$

Any point of Cartesian coordinates (x, y), the amounts of red, green and blue light to match the colour response of an observer, can be obtained by integration

$$R(x, y) = \int_0^\infty f(x, y, \lambda) \cdot R(\lambda) d\lambda$$
$$G(x, y) = \int_0^\infty f(x, y, \lambda) \cdot G(\lambda) d\lambda$$
$$B(x, y) = \int_0^\infty f(x, y, \lambda) \cdot B(\lambda) d\lambda$$

with $R(\lambda)$, $G(\lambda)$, $B(\lambda)$ known as the spectral tristimulus values.

The domain (x, y) of the image $f(x, y)$ is generally bounded on a $(N \times M)$ rectangle and can be sampled and quantized to obtain a digital image.

5.5.2 Digital Red-Green-Blue (RGB) Image

The RGB pixel values are a result of the sampling and quantization of an analog image, the method of sampling is done as follows.

The energy function values for the $(N \times M)$ image $f(x, y)$ are sampled at regular spatial intervals .

$$(x, y) = (i \cdot \Delta x, j \cdot \Delta y)$$

with $i = 1, 2, 3, \dots n$ and $j = 1, 2, 3, \dots, m$.
The values of Δx and Δy defined as

$$\Delta x = \frac{N}{n}, \Delta y = \frac{M}{m}$$

[136] Nixon, M. S. and Aguado, A. S. Feature Extraction and Image Processing. Amsterdam: Academic, 2008. Page 62.

where $N, M, n, m \in \mathbb{N}$ with n, m as the number of sampling points.

The colour values are measured at each point $(i \cdot \Delta x, j \cdot \Delta y)$, this value is then quantized (rounded) for digital use, this point is known as a pixel.

Each pixel has a colour that can be matched by specific proportions of the primary colours, red, green and blue, this is known as additive colour matching [137].

The result is a digital image $f_D(x, y)$ with x and y the integer co-ordinates for the $(N \times M)$ pixels.

If each pixel is encoded with 24 bits then each element of the RGB vector will be quantized onto one of $2^{24} = 16777216$ binary values. A colour RGB image with $(N \times M)$ pixels will then require $3 \times N \times M \times 24$ bits to store or transmit.

5.5.3 RGB Pixel as a Random Vector

Each pixel $\boldsymbol{X}_{i,j}$ is a random RGB vector, where the index (i, j) identifies to which row and to which column the pixel belongs. The random vector $\boldsymbol{X}_{i,j}$ is a 3 dimensional vector containing the RGB values of each pixel.

$$\boldsymbol{X}_{i,j} = \begin{pmatrix} X_R \\ X_G \\ X_B \end{pmatrix}.$$

The random variables X_R, X_G, X_B that make up the vector are discrete values, where

$$\{X \in \mathbb{Z}^+ : 0 \leq X \leq 255\}.$$

The digital image can then be defined by the $\big((N \times M) \times 3\big)$ matrix f_D.

$$f_D = \begin{pmatrix} \boldsymbol{X}^T{}_{1,1} \\ \boldsymbol{X}^T{}_{1,2} \\ \vdots \\ \boldsymbol{X}^T{}_{1,M} \\ \boldsymbol{X}^T{}_{2,1} \\ \vdots \\ \boldsymbol{X}^T{}_{N,M} \end{pmatrix}.$$

[137] Pratt, W. K. Digital Image Processing. Fourth Edition. Wiley. 2007. Pages 49-50.

Figure 5.2 A 24 Bit RGB Format Photo Digital Image of taken at Second Beach, Port St. Johns, Eastern Cape. The image is in 24 bit RGB format with a length $N = 800$ pixels and a height of $M = 1200$ pixels. The space required to store the image is 2813 kilobytes.

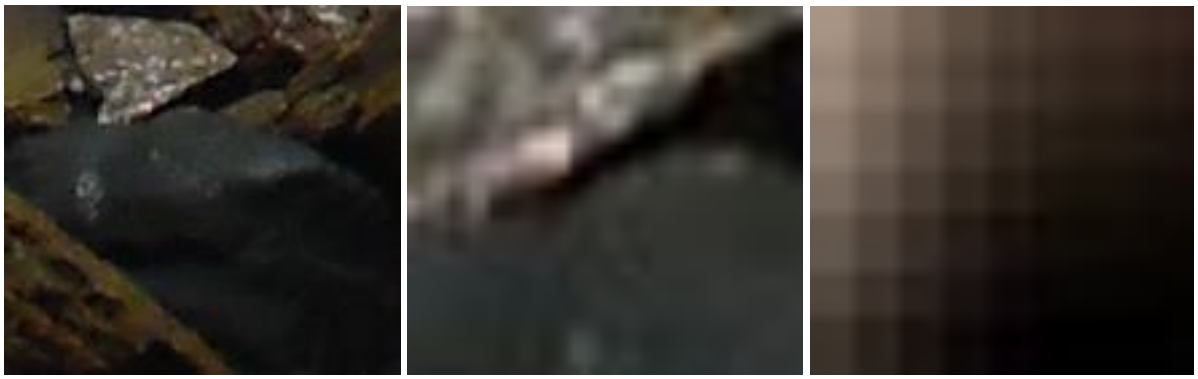


Figure 5.3 Zooming into the Photo By zooming into the image (zooming increases from leftmost to rightmost image) over a particular area, the pixels making up the image become visible. Each of these pixels is obtained my colour matching of the three primary colours, red, green and blue. Each pixel is given by the vector $\boldsymbol{X}_{i,j}$.

5.5.4 The Image Compression Process

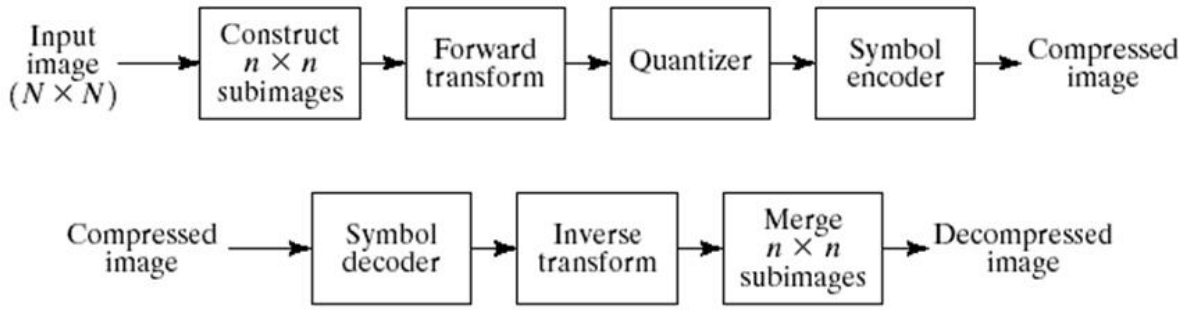


Figure 5.4 Process of Image Compression The $(N \times N)$ and $(n \times n)$ have been replaced with $(N \times M)$ and $(k \times k)$ respectively for notational purposes within this document [138].

This short image study is similar to the process of compressing a colour RGB image into a JPEG image format, based on the procedure given in [139] and several other texts.

Compressing an Image

1. The image was loaded onto SAS 9.2 using the code included at the end of the chapter. Code 5.1 SAS 9.2 Code: Input and Output Image.

2. The RGB vectors are converted to YC_bC_r by means of a linear transformation of the 960000 row by 3 column RGB matrix [140].

$$RGB \Rightarrow YC_bC_r$$

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.500 \\ 0.500 & -0.419 & -0.081 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix}$$

The colour space transformation from RGB to YC_bC_r is used since, human observers are more sensitive to changes in brightness than colour [141]. YC_bC_r allows for manipulations of the luma (Y), chroma blue $(C_b)$and chroma red (C_r) vectors as opposed to the red, green and blue RGB colour vectors.

[138] Image obtained in a Google search.
[139] Solomon, D. and Motta, G. Handbook of Data Compression, Springer. 2010. Pages 520-534.
[140] Pratt, W. K. Digital Image Processing. Fourth Edition. Wiley. 2007. Page 83.
[141] Pratt, W. K. Digital Image Processing. Fourth Edition. Wiley. 2007. Pages 39-42.

Code 5.1 SAS 9.2 Code: RGB Transform

```
data bitmap_raw1;
set bitmap_raw1;
t=_n_;
Ty=(0.299*Red+0.587*Green+0.114*Blue);
TCb=-0.169*Red-0.331*Green+0.5*Blue+128;
TCr=0.5*Red-0.419*Green-0.081*Blue+128;
run;
```

3. The image is split into blocks of 8×8 pixels, for each block each of the Y, C_b and C_r data vectors undergo a discrete cosine transform as discussed earlier in this chapter.

The transform coefficients or weights of the projections of each 8×8 block onto the set of (orthogonal) basis images correspond to the amount of energy of the specific basis function within the original image. Since these basis images are orthogonal, then the transform coefficients will be uncorrelated.

Code 5.2 SAS 9.2 Code: Discrete Cosine Transform

```
dct=j(8,8,0);
r=nrow(dct);
c=ncol(dct);
pi=3.14159265358979323846264338327795;
      do i=2 to r;
            do j=1 to c;
                  dct[i,j]=sqrt(2/r)*COS((2*(j-1)+1)*(i-1)*pi/(2*r));
            end;
      end;

do j=1 to c;
      dct[1,j]=sqrt(1/r);
end;
y_red=dct*red_8_8*dct`;
y_green=dct*green_8_8*dct`;
y_blue=dct*blue_8_8*dct`;
```

Table 5.1 Eight by Eight Discrete Cosine Transform Matrix

Eight by Eight Discrete Cosine Transform Matrix							
0.3536	0.3536	0.3536	0.3536	0.3536	0.3536	0.3536	0.3536
0.4904	0.4157	0.2778	0.0975	-0.0975	-0.2778	-0.4157	-0.4904
0.4619	0.1913	-0.1913	-0.4619	-0.4619	-0.1913	0.1913	0.4619
0.4157	-0.0975	-0.4904	-0.2778	0.2778	0.4904	0.0975	-0.4157
0.3536	-0.3536	-0.3536	0.3536	0.3536	-0.3536	-0.3536	0.3536
0.2778	-0.4904	0.0975	0.4157	-0.4157	-0.0975	0.4904	-0.2778
0.1913	-0.4619	0.4619	-0.1913	-0.1913	0.4619	-0.4619	0.1913
0.0975	-0.2778	0.4157	-0.4904	0.4904	-0.4157	0.2778	-0.0975

After the discrete cosine transform has been applied to the 8×8 block matrix, the result is a coefficient matrix or transformed matrix $\mathcal{Y}$.

The top left hand corner of $\mathcal{Y}$ contains the coefficients which are the most important in terms of variation and therefore information content, these are generally larger than zero in absolute value, while the coefficients decrease in absolute value as one moves towards the bottom right-hand corner.

4. Scalar quantization is applied to the coefficient matrix $\mathcal{Y}$, however, each transformed matrix $\mathcal{Y}_Y, \mathcal{Y}_{C_b}, \mathcal{Y}_{C_r}$ is firstly divided by a scaled quantization matrix.

The $\mathcal{Y}_{C_b}, \mathcal{Y}_{C_r}$ matrixes are divided by the chrominance quantization matrix, while $\mathcal{Y}_Y$ is divided by the luminance quantization matrix. These matrices are scaled by an integer factor ρ that determines the amount of compression.

The scaled coefficients are then scalar quantized by rounding down to the nearest integer. The scaling reduces the already small spread of lower order transform coefficients to very near zero, thereby increasing the probability that they will be quantized onto zero by rounding.

Code 5.3 SAS 9.2 Code: Quantization Matrix with Scaling Factor $\rho = 5$

```
qr=round(y_red_1/(5*C_quantization_matrix));
qg=round(y_green_1/(5*L_quantization_matrix));
qb=round(y_blue_1/(5*C_quantization_matrix));
```

Table 5.2 Luminance Quantization Matrix [142]

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

[142] Solomon, D. and Motta, G. Handbook of Data Compression. Springer. 2010. Page 529.

Table 5.3 Chrominance Quantization Matrix

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

The scalar quantization of the transform coefficients, results in the majority of the matrix being zero, in particular the bottom-right half of the matrix, while the top-left corner is generally greater than zero in absolute terms.

5. The quantized coefficients are then encoded using an entropy encoding algorithm using run length encoding and variations of either Huffman encoding or an arithmetic encoder.

The top left hand coefficient is commonly known as a DC coefficient and is encoded separately from the other 63 AC coefficients, the reason for this will become apparent in section 5.5.5.3 [[143]].

The DC coefficients are first differenced and then encoded using a table based on Huffman encoding. The use of a table is possible due to the fact that the DC coefficients are very similar, in general have a particular distribution shape and are the average of all the other 63 pixels in their particular 8×8 pixel matrix.

The AC coefficients on the other hand are encoded using a combination run length encoding, as illustrated in example 1.1 in chapter one, and either Huffman encoding or arithmetic encoding. The AC coefficients, often have long runs of zero's and are encoded in a zig-zag fashion.

[143] For a more complete discussion see Solomon, D. and Motta, G. Handbook of Data Compression. Springer. 2010. Pages 530-535.

Reconstructing an Image

1. Decode quantized coefficients, which will not be shown in this document.
2. Multiply by quantization table.

Code 5.4 SAS 9.2 Code: Reconstruction with Scaling Factor $\rho = 5$

```
rr=(qr#(5*L_quant_matrix));
rg=(qg#(5*C_quant_matrix));
rb=(qb#(5*C_quant_matrix));
```

3. Apply inverse discrete cosine transform.

This is achieved by transposing the DCT matrix in table 5.1 and applying the inverse transformation in section 5.4.9.

4. Convert image back to $RGB, YC_bC_r \Rightarrow RGB$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.000 & -0.001 & 1.402 \\ 1.000 & -0.344 & -0.714 \\ 1.000 & 1.772 & 0.001 \end{bmatrix} \cdot \begin{bmatrix} Y \\ C_b - 128 \\ C_r - 128 \end{bmatrix}.$$

The results of the process will be given in section 5.6.

5.5.5 Sub-images, AC coefficients and Linear Transform Similarities

Two interesting facets of the transform coefficients will be discussed in this section, namely the distribution of the AC coefficients due to the discrete cosine transform and the similarity between the Karhunen Loeve and discrete cosine coefficients.

Both facets are due to the strong correlation structure within the 8×8 pixel matrix or sub-image.

5.5.5.1 Sub-images and the Correlation Structure

Pixels can be grouped together into $(k \times k)$ blocks (or sub-images) based on their spatial location as seen in table 5.5. The random vectors in each sub-image are often very correlated due to their proximity within an image.

Table 5.4 (3×3) Block Sub-image Pixel Matrix

$\boldsymbol{X}_{i-1,j-1}$	$\boldsymbol{X}_{i-1,j}$	$\boldsymbol{X}_{i-1,j+1}$
$\boldsymbol{X}_{i,j-1}$	$\boldsymbol{X}_{i,j}$	$\boldsymbol{X}_{i,j+1}$
$\boldsymbol{X}_{i+1,j-1}$	$\boldsymbol{X}_{i+1,j}$	$\boldsymbol{X}_{i+1,j+1}$

The block consists of 8-connected neighbours of the pixel $\boldsymbol{X}_{i,j}$. These blocks can be made arbitrarily large and need not be square.

The sample correlation matrix of the sub-image can be determined for each colour.

For the set of green elements of the (3×3) sub-image, the matrix can be reshaped into a (9×1) vector and the sample correlations maybe determined. This will be shown next.

The green, reshaped (9×1) vector is given by

$$\boldsymbol{X}_G = \begin{pmatrix} X_{G,1,1} \\ X_{G,2,1} \\ X_{G,3,1} \\ X_{G,1,2} \\ \vdots \\ X_{G,3,3} \end{pmatrix}.$$

The green sample mean vector $\overline{\boldsymbol{X}}_G$ is calculated over the entire $(N \times M)$ image by obtaining the average of the corresponding values $X_{G,i,j}$ in each sub-image.

If the sub-images are $(k \times k)$ in size, then there are $\frac{N \times M}{k \times k}$ values and the element $\overline{X}_{G,i,j}$ is given by,

$$\overline{X}_{G,i,j} = \frac{k \times k}{N \times M} \sum X_{G,i,j}.$$

The sample variances ${S^2}_{G,i,j}$ are calculated for each element $X_{G,i,j}$ over the set of $\frac{N \times M}{k \times k}$ values, where ${S^2}_{G,i,j}$ is given by,

$${S^2}_{G,i,j} = \frac{k \times k}{(N \times M - k \times k)} \sum \left(X_{G,i,j} - \overline{X}_{G,i,j}\right)^2.$$

For the image in figure 5.2, for illustrative purposes, the value assigned for $k = 4$, so each sub-image element is added over the entire set of $\frac{1200 \times 800}{4 \times 4} = 60000$ sub-images.

The sample correlation between the elements is determined over the entire image, by the formula

$$Correlation\ Matrix = \frac{\left((\boldsymbol{X}_G - \overline{\boldsymbol{X}}_G)(\boldsymbol{X}_G - \overline{\boldsymbol{X}}_G)^T\right)}{Trace\left((\boldsymbol{X}_G - \overline{\boldsymbol{X}}_G)(\boldsymbol{X}_G - \overline{\boldsymbol{X}}_G)^T\right)}.$$

The sample correlation matrix for figure 5.2 with $k = 4$ is given below.

Table 5.5 Sample Correlation Matrix

	$X_{G,1,1}$	$X_{G,1,2}$	$X_{G,1,3}$	$X_{G,1,4}$	$X_{G,2,1}$	$X_{G,2,2}$	$X_{G,2,3}$	$X_{G,2,4}$	$X_{G,3,1}$	$X_{G,3,2}$	$X_{G,3,3}$	$X_{G,3,4}$	$X_{G,4,1}$	$X_{G,4,2}$	$X_{G,4,3}$	$X_{G,4,4}$
$X_{G,1,1}$	1	0.978	0.955	0.93	0.992	0.97	0.948	0.924	0.984	0.964	0.942	0.916	0.893	0.876	0.857	0.833
$X_{G,1,2}$	0.978	1	0.977	0.951	0.97	0.992	0.969	0.945	0.963	0.984	0.962	0.937	0.874	0.894	0.875	0.852
$X_{G,1,3}$	0.955	0.977	1	0.974	0.948	0.97	0.992	0.967	0.942	0.963	0.985	0.959	0.855	0.875	0.895	0.872
$X_{G,1,4}$	0.93	0.951	0.974	1	0.922	0.944	0.966	0.992	0.917	0.938	0.96	0.985	0.833	0.852	0.873	0.895
$X_{G,2,1}$	0.992	0.97	0.948	0.922	1	0.978	0.955	0.93	0.992	0.971	0.949	0.923	0.901	0.882	0.863	0.839
$X_{G,2,2}$	0.97	0.992	0.97	0.944	0.978	1	0.977	0.952	0.971	0.993	0.97	0.944	0.881	0.902	0.883	0.858
$X_{G,2,3}$	0.948	0.969	0.992	0.966	0.955	0.977	1	0.974	0.949	0.971	0.993	0.966	0.861	0.882	0.903	0.878
$X_{G,2,4}$	0.924	0.945	0.967	0.992	0.93	0.952	0.974	1	0.924	0.946	0.968	0.993	0.839	0.859	0.88	0.902
$X_{G,3,1}$	0.984	0.963	0.942	0.917	0.992	0.971	0.949	0.924	1	0.978	0.956	0.93	0.908	0.889	0.87	0.846
$X_{G,3,2}$	0.964	0.984	0.963	0.938	0.971	0.993	0.971	0.946	0.978	1	0.977	0.951	0.888	0.909	0.889	0.865
$X_{G,3,3}$	0.942	0.962	0.985	0.96	0.949	0.97	0.993	0.968	0.956	0.977	1	0.974	0.867	0.888	0.909	0.885
$X_{G,3,4}$	0.916	0.937	0.959	0.985	0.923	0.944	0.966	0.993	0.93	0.951	0.974	1	0.844	0.864	0.886	0.909
$X_{G,4,1}$	0.893	0.874	0.855	0.833	0.901	0.881	0.861	0.839	0.908	0.888	0.867	0.844	1	0.98	0.959	0.935
$X_{G,4,2}$	0.876	0.894	0.875	0.852	0.882	0.902	0.882	0.859	0.889	0.909	0.888	0.864	0.98	1	0.979	0.955
$X_{G,4,3}$	0.857	0.875	0.895	0.873	0.863	0.883	0.903	0.88	0.87	0.889	0.909	0.886	0.959	0.979	1	0.976
$X_{G,4,4}$	0.833	0.852	0.872	0.895	0.839	0.858	0.878	0.902	0.846	0.865	0.885	0.909	0.935	0.955	0.976	1

There is a high correlation between the elements contained within each vector, in particular adjacent pixels, the correlation seems to decrease slowly as the distance between pixels increases, however all the elements within the data vector still remain highly correlated.

In the previous three chapters it was shown that,

- The strong dependence relationship between pixels means that encoding vectors (as opposed to scalars) would lead to larger decreases in code rate.

- By applying an orthogonal transform to the data, an increase in efficiency of a scalar quantizer applied to the transformed image data, was to be expected.

5.5.5.2 Comparison of Transform Coefficients due to KLT and DCT

The discrete cosine transform and Karhunen-Loeve transform were discussed in sections 5.4.5 and 5.4.8 respectively, in section 5.4.10 it was shown that under certain circumstances, namely a strong correlation structure, that the coefficients of both transformations should be very similar. This was tested on figure 5.2 and the results are contained below in table 5.6.

Table 5.6 The DCT and KLT Transformed Coefficients

Transformed Coefficient Variances for DCT and KLT (Three by Three Pixel Vector)					
KLT – Red Variances	KLT – Green Variances	KLT – Blue Variances	DCT – Red Variances	DCT – Green Variances	DCT – Blue Variances
11570.81243	11064.012	6786.8834	11569.957	11063.375	6786.0709
212.7326117	190.10672	165.3958	170.04413	153.35281	134.3281
83.59251066	87.074358	83.916816	83.684224	87.162095	83.961279
27.79516128	28.992503	27.932139	56.682701	51.112232	44.783035
13.27699135	13.820358	12.951452	27.839153	29.001855	27.933561
0.975048529	0.9735916	0.9904367	0.9689963	0.9675922	0.9844558
0.329662865	0.3283919	0.3382222	0.3229978	0.3225307	0.3281518
0.310876418	0.3112064	0.3125115	0.3229945	0.3225245	0.3281459
0.104781369	0.1046898	0.1062306	0.1076625	0.1075051	0.1093784

The variation of the discrete cosine transform coefficients is very close to that of the Karhunen-Loeve transform coefficients, the second, fourth transform coefficient of the discrete cosine transform show marked differences. The Karhunen-Loeve transform variances decrease quicker in the early-middle coefficients than the discrete cosine transform variances. The lower variances, however are pretty much the same for both transforms.

5.5.5.3 Histograms of the AC and DC Transform Coefficients

The discrete cosine transform coefficients were plotted for all 60000 (4×4) sub-images, they were almost identical in form to the Karhunen-Loeve coefficients for the RGB colours.

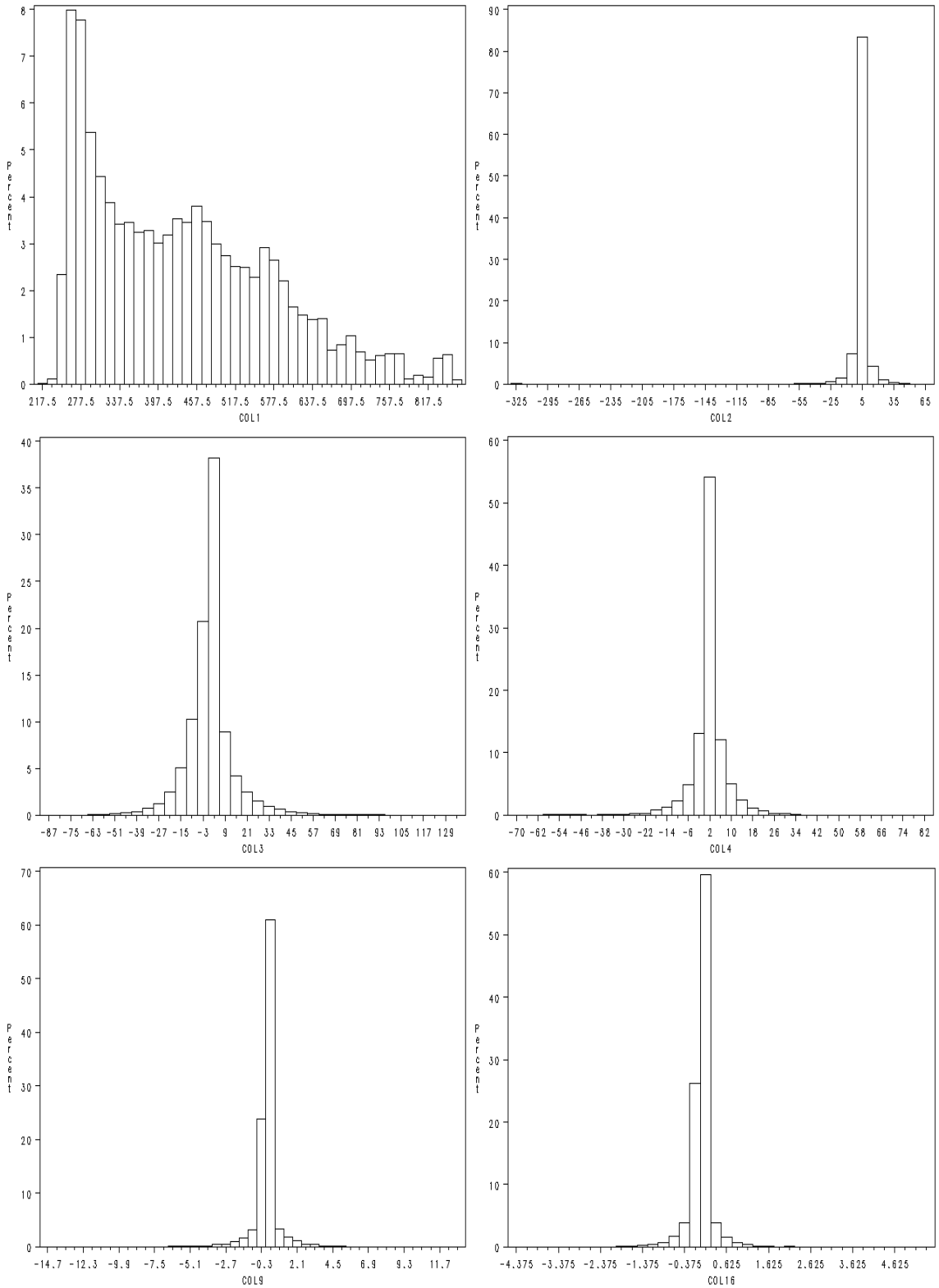


Figure 5.5 Histograms of the AC and DC Coefficients The top left hand corner image is the histogram of the DC coefficients, while the rest of the coefficients are the remaining 15 AC coefficients.

In the case of both the discrete cosine transform and the Karhunen-Loeve transform, the transform was applied on each of the 60000 (4×4) sub-images of a RGB image matrix of size 800×1200.

The histograms for all AC coefficients seemed to follow a similar distribution for both transformations. The possible distribution of the coefficients was confirmed by several sources, including the paper [144].

A summary of the discussion presented will follow.

The (4×4) sub-image contains 60000 values of the un-transformed pixel values $X_{i,j}$ for red, green and blue.

According to the authors, since the pixel values are assumed to be identically distributed, then the weighted sum of these pixels is approximately normally distributed, due to the central limit theorem, while the spatial correlation between pixels determines the width of the distribution.

Now since both the discrete cosine transform and Karhunen-Loeve transform are unitary transformations, the mean of the transform coefficients is zero, while the variance $\sigma_{i,j}$ differs for each element $\tilde{X}_{i,j}$ of the sub-image.

Therefore each of the 16 transformed coefficients $\tilde{X}_{i,j}$ constituting the (4×4) sub-image matrix follows a normal distribution with a variance that differs $\sigma_{i,j}^2$ where $i = 1, 2, \dots, 4$ and $j = 1, 2, \dots, 4$. That is

$$f_X\left(x_{i,j} | \sigma_{i,j}^2 = \sigma^2\right) = \frac{e^{-\frac{(x_{i,j})^2}{2 \cdot \sigma^2}}}{\sqrt[2]{2 \cdot \pi \cdot \sigma^2}}.$$

According to the author, empirical data showed that the variances $\sigma_{i,j}^2$ which where estimated by the sample variances $s_{i,j}^2$ were approximately exponentially or half Gaussian distributed.

The conditional distribution of $X_{i,j}$ is

$$f_X\left(X_{i,j}\right) = \int_0^\infty f_X\left(x_{i,j} | \sigma_{i,j}^2 = \sigma^2\right) \cdot f_\sigma(\sigma^2) \cdot d(\sigma^2).$$

Assuming an exponential probability density function for the variance σ

$$f_\sigma(\sigma) = \begin{cases} \lambda \cdot e^{-\lambda \cdot \sigma} & for\ \sigma \geq 0 \\ 0 & for\ \sigma < 0 \end{cases}.$$

[144] Lam, E. Y. and Goodman, J. W. A Mathematical Analysis of the DCT Coefficient Distributions for Images. IEEE Trans.on Image Proc., vol. 9, no. 10, Oct. 2000. Pages 1-5.

$$f_X(X_{i,j}) = \int_0^{\infty} \frac{e^{-\frac{(X_{i,j})^2}{2\cdot\sigma^2}}}{\sqrt[2]{2\cdot\pi\cdot\sigma^2}} \cdot \lambda \cdot e^{-\lambda\cdot\sigma^2} d(\sigma^2).$$

Then by substituting $t = \sigma^2$ and $dt = (2\cdot\sigma)d\sigma$ into the expression, we get

$$= \frac{2\cdot\lambda}{\sqrt[2]{2\cdot\pi}} \cdot \int_0^{\infty} e^{-\frac{(X_{i,j})^2}{2\cdot t^2}} \cdot e^{-\lambda\cdot t^2} dt.$$

$$= \frac{2\cdot\lambda}{\sqrt[2]{2\cdot\pi}} \cdot \int_0^{\infty} e^{-\frac{(X_{i,j})^2}{2\cdot t^2} - \lambda\cdot t^2} dt.$$

$$= \frac{2\cdot\lambda}{\sqrt[2]{2\cdot\pi}} \cdot \frac{1}{2} \cdot \sqrt[2]{\frac{\pi}{\lambda}} \cdot e^{-2\cdot\sqrt{\lambda\cdot\frac{(X_{i,j})^2}{2}}}$$

$$= \frac{\sqrt{2\cdot\lambda}}{2} \cdot e^{-\sqrt{2\cdot\lambda}\cdot|X_{i,j}|}$$

$$f_X(X_{i,j}) \sim Laplace\left(\sqrt[2]{2\cdot\lambda}\right).$$

The authors argue similarly for a σ that follows the half Gaussian distribution, the result is also a Laplace distributed $X_{i,j}$

$$f_X(X_{i,j}) \sim laplace\left(\sqrt[2]{\frac{2}{s}}\right).$$

The width of the Laplace distributions become smaller as the coefficients $X_{i,j}$ move towards the right-hand bottom corner of the coefficient matrix, in other words as (i,j) get larger, the coefficients $X_{i,j}$ which are centred at zero have increasingly smaller variance.

The width of the transform coefficient distribution, it's steady monotone decrease and it's zero mean due to the nature of the orthogonal transform provide a template for data compression.

The Laplace distribution of the discrete cosine and Karhunen-Loeve transform coefficients is not agreed upon by everybody see [145] for arguments on why no particular distribution will fit the DCT statistics exactly over a wide range of applications.

[145] Yovanof, G. S. and Liu, S. Statistical Analysis of the DCT Coefficients and their Quantization Error. In Conf. Rec. 30th Asilomar Conf. Signals, Systems, Computers, vol. 1, 1997. Pages 1-5.

In the book [146], the author provides several look up tables of Huffman encodings for both the AC and differenced DC coefficients - independently of the image encoded [147]! The author, however comments that adaptive arithmetic encoders are sometimes used, and can achieve slightly better (5%-10%) compression ratios.

[146] Solomon, D. and Motta, G. Handbook of Data Compression. Springer. 2010. Pages 532-535.

[147] This seems to agree with the article by Lam, E. Y. and Goodman, J. W. that the coefficients do in fact follow a specific distribution, namely the Laplace distribution, with deviation occurring at parameter as opposed to distribution level.

5.5.6 Image Compression Results

Figure 5.6 Comparison of the Compressed Image with its Uncompressed Counterpart The top image is the original 24 bit 800 by 1200 RGB bitmap image, while below it is the reconstructed image for increasing (left-to-right) compression ratios obtained by the increasing the quantization matrix scaling factor ρ from 1 through to 10.

Figure 5.7 The Error Image The difference between the original 24 bit 800 by 1200 RGB bitmap image the reconstructed image for increasing (left-to-right) compression ratios obtained by the increasing the quantization matrix scaling factor ρ from 1 through to 10.

The error image was determined for each RGB vector by calculating the absolute difference,

$$E_R = (|O_R - R_R|)$$
$$E_G = (|O_G - R_G|)$$
$$E_B = (|O_B - R_B|)$$

where E_R is the error image for the vector R, O_R is the original image vector R and R_R is the reconstructed image vector R.

Table 5.7 Comparison of Compression Ratios

ρ	Original Bitmap Storage Capacity (kilobytes)	Compressed Storage Capacity (kilobytes)	% Original Storage Capacity over the Compressed Storage Capacity
1	2813	310	907%
2	2813	241	1167%
3	2813	201	1400%
4	2813	167	1684%
5	2813	145	1940%
6	2813	121	2325%
7	2813	105	2679%
8	2813	92	3058%
9	2813	86	3271%
10	2813	71	3962%

5.5.6.1 Comments on Image Compression Results

The two images in figure 5.6 are compared and the error image in figure 5.7 is examined to highlight the difference between the two images.

Four differences are briefly discussed, namely, the blocking and pixilation, ringing around several edges, the error image's colouration and the compression achieved.

- Blocking Artefacts and pixilation

For smaller values of ρ pixilation becomes perceptible, even for $\rho = 2$ there is small criss-cross artefacts at the edges of the rocks in the foreground. As ρ gets larger, particularly around $\rho = 4$, blocking artefacts are very noticeable and affect the image quality.

For $\rho > 4$ the image quality is severely distorted with discolouration and loss of specific elements (sticks, edges and water foam) of the image.

- Ringing

Due to the discrete cosine transform approximating jumps in colour at the edges of the rocks, there is a ring around them which is more apparent for higher values of ρ.

- Error Image

The error image shows the increase in discolouration with the increase in the parameter ρ, it also shows the loss of clearly defined edges in particular with the water and foam.

- Compression

The amount of compression achievable is high, but the loss of vital image information is very apparent for increasing values of ρ.

5.6 Conclusions of Chapter Five

- Vector quantization is far more computationally complex and time consuming than scalar quantization, so alternative strategies are considered in order to make up for using scalar quantization as opposed to vector quantization. These include scalar quantization combined with bit allocation approaches, orthogonal transforms and entropy encoding.

- The bit allocation problem attempts to assign bits optimally to the random variables that constitute the random vector. The variance of the random variable provides a template, that is, to achieve an optimal bit allocation more reproduction points need to be assigned to random variables with greater variance.

- The computational complexity, bit allocation problem and trouble with dependence structure is partially solved by means of scalar quantization combined with entropy encoding and orthogonal transforms, the latter is discussed in greater detail.

- The discrete Karhunen-Loeve transform is an orthogonal transform applied to the statistical nature of the data. It is the optimal orthogonal transformation for de-correlation and bit allocation. The discrete Karhunen-Loeve is however data dependent and therefore impractical due to time economy.

- The discrete cosine transform is a special case of the discrete Fourier transform and is substantially more computationally efficient than Karhunen-Loeve transform in terms of computational economy, due to being independent of the statistical nature of the data.

- The discrete cosine transform, is a limiting case of the discrete Karhunen-Loeve transform where the correlation between random variables that constitute the random vector approach a unity.

- The discrete cosine transform coefficients and the Karhunen-Loeve transform coefficients appear to follow a Laplace distribution centred around zero with decreasing variance, which allows for bit allocation and uniform scalar quantization of the in-significant coefficients onto zero.

Final Conclusion and Summary

This dissertation looked at how data compression works, how quantization is used in compression, how vector quantization and scalar quantization compare in reducing error and compressing data and how linear transformations assist in the data compression process.

These are the conclusions.

Data compression is divided into two overlapping compression methods, namely lossy and lossless compression, where lossless compression forms an integral part of lossy compression.

Lossless compression of discrete data has lower bound on the bit rate per data element (or code rate). This lower bound is known as the entropy which is calculated as a function of the probability distribution determined over the set of outcomes obtained at the data source (a random variable) and is measured in bits.

Several important relationships between the probability distribution, the entropy and the code rate were identified.

> Firstly, increased heterogeneity results in lower entropy. Zero entropy in the case of maximum heterogeneity, or determinism and maximum entropy for discrete uniform data, which exhibits maximum homogeneity.
>
> Secondly, grouping or blocking data elements reduces or leaves the (joint) entropy unchanged. In the case of blocks comprised of independent random variables, the joint entropy is equal to the entropy of the ungrouped data, while in the case of a dependence structure between random variables, the joint entropy is reduced below that of the entropy of the ungrouped random variables.
>
> Lastly, the effect of blocking data elements reduces the code rate, this is true even in the case where the blocks are comprised of independent random variables.

Lossy compression is achieved by combining lossless compression with scalar or vector quantization.

Scalar Quantization

Scalar quantization is a technique of (selectively) introducing errors into scalar data by assigning observations to a predefined set of codewords, or a codebook. Scalar quantization decreases the range of the random source of the data thereby increasing the compression.

Three types of scalar quantization were discussed, namely uniform scalar quantization, companded uniform scalar quantization and Lloyd Max scalar quantization. The difference between the three methods is found in the design phase of the quantizer that is how the codebook and intervals are determined.

The implementation is the same for all three, that is, assign the observation to the codeword which represents the interval where the observation is taken.

Since scalar quantization introduces error into the data, distortion measures are required to measure the error in order to determine the performance for a given constraint on codebook size.

Some scalar quantization designs rely on performance measures that minimize the mean error for a given codebook size, alternative designs minimize the mean error against the entropy of the quantized random source- inverse requirements that are described by the rate – distortion theory.

The LBG algorithm is a greedy iterative procedure for designing a codebook and boundary points by clustering and assigning temporary means to simulated or real data. The LBG algorithm is the result of iteratively applying Lloyd Max conditions which result from the minimization of the mean squared distortion measure.

For low code rates, several alternative scalar quantization techniques are available, however for high code rates uniform scalar quantization is (near) optimal in that it minimizes the code rate for any bounded, well behaved random source.

Vector Quantization

Vector quantization is a generalization of scalar quantization in its application to random vectors as opposed to scalars.

Vector quantization allows for greater flexibility over scalar quantization in the design phase which results in smaller errors and greater compression. This is due to the diversity of available shapes, the volume increase due to increased dimensionality and the ability to incorporate dependence structure into the design.

The rate distortion lower bound is an optimum bound on the code rate for a given distortion. The improvements in the design of a vector quantizer imply that for a large dimension, a vector quantizer is able to achieve the rate distortion lower bound (even) in the presence of statistical dependency.

The generalized Lloyd algorithm extends the LBG algorithm into two or more dimensions. The ECVQ algorithm and the EM algorithm can be viewed as extensions of the generalized Lloyd algorithm framework by using different distortion measures, resulting in more parameters and extensions to the optimization process.

Linear Scalar Transformations and Picture Study

Vector quantization is very effective at quantizing data, however it is often too time consuming in both design and implementation phases, especially when the probability structure is unknown or changing. An alternative to vector quantization is scalar quantization combined with techniques that improve scalar quantization performance.

High code rate scalar quantizers are very effective at quantizing data, however when dependence structure is present, vector quantizers are still far better.

A bit allocation strategy that assigns more codewords to variables with greater variance combined with a linear transform that removes linear dependence before scalar quantization can be achieved with a single linear operation, the Karhunen Loeve transform (also known as the Principle Component transform).

The Karhunen Loeve transform, which is data dependent, can be substituted with the Discrete Cosine transform, which is data independent, with very similar results when the data is highly correlated, as in image data.

The Discrete Cosine transform of image data results in $n-1$ predictable probability distributions, with zero means and decreasing variance, of the n transform coefficients.

These coefficients are quantized by means of a uniform scalar quantizer and then encoded by an entropy encoder.

All the processes are reversible except for quantization, so the data can be decoded easily with only the codebook and the compressed transform coefficients.

Appendix

Code 5.1 SAS 9.2 Code: Input and Output Image

```
SAS/4GL macro for reading BMP files (DIB3, no compression, 24 bits
only)Copyright 2010 Bogdan Taranta
%LET BLIBNAME=WORK;
/*destition libname, output will be stored in BITMAP_RAW dataset */
%LET BSOURCE=g:\Vector Quantization\image
segmentation\Test_pics\test_pic7.bmp;
/* path to BMP picture */
/* reading raw HEX data */
data BITMAP_RAW;
  infile "&BSOURCE" recfm=n;
  length byte $1;
  input byte $char1.;
  offset = _n_-1;
  /*byte possition*/
  value = rank(byte);
  /*byte value*/
  format offset HEX4.;
  keep offset value;
run;
/*data _null_;*/
/*  set bitmap_raw;*/
/*  file "c:\temp\mytest.bmp" recfm=n ;*/
/**/
/*  out_char = byte(value);*/
/*  put out_char $char1. @;*/
/*run;*/
/* making DIB pattern header, reffering to
http://en.wikipedia.org/wiki/BMP_file_format */
data BITMAP_HEADER;
  format soffset HEX4.;
    input @1 soffset HEX4. /*field position*/
        @6 bytes /*field lenght*/
        @9 desc $; /*field description*/
  do i=1 to bytes; /*repeating field */
    add = (i-1)*256; /* this will be used to convert values into integers*/
    if add = 0 then add = 1;
    offset = soffset+i-1; /*ordinary counter*/
    output;
  end;
  drop soffset bytes i;
  format offset HEX4.;
  datalines;
0000  2 MAGIC
0002  4 F_SIZE
0006  2 RES1
0008  2 RES2
000A  4 B_OFFSET
000E  4 H_SIZE
0012  4 B_WIDTH
0016  4 B_HEIGHT
001A  2 COL_PLAN
001C  2 COL_DEP
001E  4 COMPRESS
0022  4 I_SIZE
```

```
0026   4 HRES
002A   4 VRES
002E   4 PALETE
0032   4 IMPCOL
run;
proc sort data=BITMAP_HEADER  out=BITMAP_HEADER ;
  by offset;
run;
proc sort data=BITMAP_RAW  out=BITMAP_RAW ;
  by offset;
run;
/* comparing DIB header from file to previosly generated pattern */
data BITMAP_HEADER;
  merge BITMAP_HEADER (in=l)  BITMAP_RAW;
  by offset;
  if l = 0 then stop;/* stop when pattern is finished */
run;
/* reading data from DIB header into macro variables */
proc sort;
  by desc;
run;
data BITMAP_HEADER;
  set BITMAP_HEADER;
  by desc;
  retain cvalue 0;
  cvalue = cvalue + (value*add);
  if last.desc then do;
    call symput(desc,compress(put(cvalue,best12.)));
    output;
    cvalue = 0;
  end;
  keep desc cvalue;
run;

%put &b_offset;
%put &b_width;
%put &b_height;
data  null ;
  pixel_row_total=ceil(&b_width*24/32)*4;
  call symput("pixel row total",compress(put(pixel_row_total,best12.)));
  pixel_row = (24*&b_width/32)*4;
  call symput("pixel_row",compress(put(pixel_row,best12.)));
run;
%put &pixel_row_total;
%put &pixel row;
data BITMAP_RAW1;
  set BITMAP RAW (where=(offset>=&B_OFFSET));
  retain X 0;
  retain Y &B_HEIGHT;
  retain BLUE;
  retain GREEN;
  retain RED;
  retain BYTE3 0;
  retain byte_cnt 1;

  if (&pixel row=&pixel_row_total and byte_cnt>&pixel_row) then do;
    byte_cnt = 1;
    Byte3 = 0;end;

  if (byte cnt<=&pixel row) then do;
    if ( BYTE3 < 3 ) then BYTE3 = BYTE3 + 1;
```

```
    else BYTE3 = 1;
    if BYTE3 = 1 then BLUE=value;
    if BYTE3 = 2 then GREEN=value;
    if BYTE3 = 3 then RED=value;
    if BYTE3 = 1 then X = X + 1;
    if ( BYTE3 = 3 ) then do;
      if X>&B WIDTH then do;
        X = 1;
        Y = Y - 1;
      end;
    end;
    if BYTE3 = 3 and X<=&B_WIDTH then do;
      output BITMAP_RAW1;
      *myout = 1;
    end;

    byte_cnt + 1;
  end;
  else if (byte_cnt<&pixel_row_total) then do;
    /*nothing - these are padding to make up 32 bytes*/
    byte_cnt + 1;
  end;
  else do;
    Byte3 = 0;
    X = X + 1;
/*    if X>&B_WIDTH then do;*/
/*      X = 1;*/
/*      Y = Y - 1;*/
/*    end;*/
    byte cnt = 0;
    byte_cnt + 1;
  end;
  keep X Y BLUE GREEN RED;
run;

%macro print_2_file(in,out);
%LET BTARGET=g:\Vector Quantization\image
segmentation\Test_pics\&out._result.bmp;
proc sort data=&in. out=bitmap_raw2;
  by descending y x;
run;
data bitmap_raw_header;
  set bitmap_raw;
  where (offset<&B_OFFSET);
run;
proc transpose data=bitmap_raw2 out=bitmap_raw_tps(rename=(col1=value));
  by descending y x;
  var blue green red;
run;
data _null_;
  set bitmap_raw_header bitmap_raw_tps;
  file "&BTARGET" recfm=n;
  out_char = byte(value);
  put out char $char1. @;
run;%mend;
```

Bibliography

1. Bartle, R.G. The Elements of Real Analysis. Second Edition. Wiley. 1976.

2. Beauregard, F. Linear Algebra. Third Edition. Addison Wesley Longman. 1995.

3. Bilmes, J. A. A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models. 1998. (PDF).

4. Biswas, S. and Srivastav, G.L. Mathematical Statistics, A Textbook. Alpha Science International. 2011.

5. Bradley , P. S. Microsoft Research and Fayyad, U. M. Microsoft Research. Refining Initial Points for K-Means Clustering. 1998. (PDF).

6. Brokish, C. W. Lewis, M. A-Law and mu-Law Companding Implementations using the TMS320C54x, MTSA. Texas Instruments. 1997. (PDF).

7. Cambel, A.B. Applied Chaos Theory: A Paradigm for Complexity. Academic press. 1993.

8. Chou, P. A. Lookabaugh, T. and Gray, R. M. Entropy-Constrained Vector Quantization. IEEE Trans. on Acoustics, Speech, and Signal Processing, Vol. 31, No.I. January 1989. (PDF).

9. Cleveland State University CIS 658 Fall 2005. (PDF).

10. Cover, T. M. and Thomas, J. A. Elements of Information Theory. Wiley Series in Telecommunications and Signal Processing. Second Edition. 2002

11. de Garrido, D. P. and Pearlman, W. A. Conditional Entropy-Constrained Vector Quantization: High-Rate Theory and Design Algorithms. 2001.

12. Deza, E. Deza, M. Dictionary of Distances. Elsevier Science & Technology Books. 2006.

13. Gersho , A. and Gray, R. M. Vector Quantization and Signal Compression. Springer. 1992.

14. Gersho, A. Asymptotically Optimal Block Quantization. IEEE Transactions on Information Theory, vol.IT-25, no.4, July 1979. (PDF).

15. Gish ,H. and Pierce, J. N. Asymptotically efficient quantizing. IEEE Trans. Inform. Theory, vol. IT-14, no. 5, pp. 676-683. (PDF).

16. Golomb, S. W., Peile, R. E. and Sholtz, R. A. Basic Concepts in Information Theory: The Adventures of Secret Agent 00111. Springer. 1994.

17. Gray, R.M and Davisson, L.D. An Introduction to Statistical Signal Processing. Cambridge University Press. 2010.

18. Guestrin, C. Machine Learning. Carnegie Mellon University. 2007. (PDF).

19. Johnson, R. and Wichern, D. Applied Multivariate Statistical Analysis. Pearson Prentice Hall. 2007.

20. Kolesnikov, A. Image Compression, Lecture 8, Scalar Quantization. Department of Computer Science, University of Joensuu, Joensuu, Finland. (PDF).

21. Lam, E. Y. and Goodman, J. W. A Mathematical Analysis of the DCT Coefficient Distributions for Images. IEEE Trans.on Image Proc., vol. 9, no. 10, Oct. 2000, pp. 1661-1666. (PDF).

22. Li, Y. Sharan, L. Adelson, E. H. Compressing and Companding High Dynamic Range Images with Subband Arcitectures. 2005. (PDF).

23. Lookabaugh, T.D. and Gray, R. M. High-Resolution Quantization Theory and the Vector Quantizer Advantage. IEEE Transactions on Information Theory, Vol. 35, no. 5, September 1989. (PDF).

24. Makhoul, J. Roucos, S. and Gish, H. Vector Quantization in Speech Coding. From the proceedings of the IEEE, vol. 73, no. 11, November 1985 , Section E, entitled Vector Quantization Model. (PDF).

25. Mathai, A.M. and Rathie, P.N. Basic Concepts in Information Theory and Statistics. Wiley. 1975.

26. Meila, M. and Heckerman, D. An experimental comparison of several clustering methods. 1998. (PDF).

27. Mitra, S. K. Digital Signal Processing a Computer-Based Approach. Mcgraw Hill Higher Education. 2002.

28. Nixon, M. S. and Aguado, A. S. Feature Extraction and Image Processing. Amsterdam: Academic, 2008. (PDF).

29. Pratt, W. K. Digital Image Processing. Fourth Edition. Wiley. 2007.

30. Ray, W.D. and Driver, R.M. Futher decomposition of the Karhunen-Loeve series representation of a stationary random process. IEEE Trans. 1970, IT-16, pp. 845-850. (PDF).

31. Roman, S. Coding and Information Theory. Springer-Verlag. 1997.

32. Rosenfeld, A. and Kak, A. C. Digital Picture Processing. Academic Press. 1976.

33. Rozanov,Y.A. Probability Theory: A Concise Course. Dover. 1977.

34. Shlens, J. Notes on Kullback-Leibler Divergence and Likelihood Theory. 2007. (PDF).

35. Solomon, D. and Motta, G. Handbook of Data Compression, Springer. 2010.

36. Wang, R. Introduction to Orthogonal Transforms with Applications in Data Processing and Analysis. Cambridge. 2012.

37. Widrow, B. Statistical Theory of Quantization. IEEE Transactions on Instrumentation and Measurement, Vol. 45, No 2, April 1996. (PDF).

38. Wiegand, T. Rate Distortion Theory & Quantization. Digital Image Communication. (PDF).

39. Wiegand, T. and Schwarz, H. Source Coding: Part I of Fundamentals of Source and Video Coding. (PDF).

40. Yovanof, G. S. and Liu, S. Statistical Analysis of the DCT Coefficients and their Quantization Error. In Conf. Rec. 30th Asilomar Conf. Signals, Systems, Computers, vol. 1, 1997, pp. 601–605. (PDF).