Empirical Study of the Connection Between Quantizer Distortion and Training Set Size

by

Jay Shah

A project submitted to the

Department of Mathematics and Statistics

in conformity with the requirements

for the degree of

Master of Science (of Eng.)

Queen's University

Kingston, Ontario, Canada

January 2004

Acknowledgements

I would first like to thank my supervisor, Dr. Tamás Linder, for his guidance and his helpful advice throughout the duration of this project. I would also like to thank Daniel Nagy for his invaluable help with regards to the problem of fast scalar quantizer design. Further thanks go to Firouz Behnamfar and Libo Zhong for their friendly assistance in the lab.

Special thanks go to Sally Dam, my family and friends for their caring concern and support.

Finally, I would like to thank Dr. Tamás Linder, the Department of Mathematics and Statistics, the School of Graduate Studies, and the Provincial Government of Ontario for their financial support over the last year.

Abstract

Optimal quantizer design is an interesting problem from both a theoretical and a practical point of view. Unfortunately, the design of optimal quantizers for the true information source is not always possible. In such cases, we resort to the design of empirically optimal quantizers. These quantizers are optimal for the empirical distribution of a given training set, but are not necessarily optimal for the distribution of the true source. It is known, however, that the distortion of these quantizers approaches optimal levels as the training set size is allowed to increase without bound [10]. It is thus of great interest to determine exactly how quickly the distortion of these quantizers converges to the optimal distortion as a function of training set size.

In this project, we first review the basics of quantization. We then examine the literature on the consistency of empirical design, and on the bounds on the performance of empirically optimal quantizers. Both upper and lower bounds are presented, the latter of which having been derived in a minimax sense. Due to the minimax nature of these lower bounds, they do not necessarily hold for specific source distributions and are thus of little use in practice. This leads us to investigate these lower bounds in more depth via simulation. We consider quantizers designed for four different source distributions: uniform, Gaussian, Laplacian, and beta. For each source distribution, we seek to find lower bounds that are tighter than the minimax bounds from the literature.

In our simulations, we consider the effect of increasing training set size on the test and training distortions of empirically optimal scalar quantizers. For each source distribution, we consider quantizers that operate at rates 1, 2, and 3 bits/sample. The minimax lower bounds mentioned above are of the form αN^{β}, where N is the cardinality of the training set and $\beta = -0.5$. In our work, we seek to find values for α and β via simulation. In particular, we are interested in the value of the parameter β. We observe that the convergence rates for certain commonly used distributions are faster than those predicted by the minimax lower bounds. That is, $\beta < -0.5$.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this work we are concerned with empirically designed quantizers. These are quantizers that are designed by teaching them about the source distribution via a training set. Unfortunately, training data may be expensive to obtain, and as the size of the training set increases, so does the complexity of the design algorithm. On the other hand, if the training set is too small, the resulting quantizer will perform rather poorly. This is because a small training set may not be representative of the source. Thus, it is of great interest to be able to determine exactly how much training data is required for a given application. The goal of this project is to determine the rate at which the distortion of an empirically optimal quantizer converges to the distortion of the theoretical optimum quantizer for the source as a function of training set size.

1.1 Contributions of this Project

In this project, we consider the test and training distortion of empirically optimal quantizers as a function of training set size for four different source distributions. These distributions are uniform, Gaussian, Laplacian, and beta. More precisely, we attempt to find lower bounds of the form αN^{β} on the rate at which these distortions converge to the optimal distortion for each source. By doing this, we hope to determine whether or not the minimax lower bounds from the literature can be tightened. We consider the performance of quantizers on these sources for rates of 1, 2, and 3 bits/sample.

For each of the twelve combinations of distribution and rate, we fit power curves to our distortion data in order to determine the sought relationship. Our results are presented in Chapter 4.

1.2 Overview of this Project

In Chapter 2, the topic of data compression is introduced. The notions of rate and distortion are then discussed. Following this discussion, the problem of quantization is addressed. The focus of this discussion is on existence, achievability, and performance bounds. Finally, several algorithms for optimal scalar quantizer design are described.

In Chapter 3, issues relevant to the implementation of this project are discussed. The primary focus of this chapter is the generation of training and test sets. In Chapter 4, the results of our simulations are presented. For each distribution, the performance of empirically optimal quantizers under three different rates is considered. Finally, concluding remarks are offered in Chapter 5.

Chapter 2

Background

The purpose of this chapter is to introduce the problem of quantization and to give motivation for the work undertaken in this project. We will begin by introducing the notion of data compression and discussing some basic quantities characterizing data compression systems, namely rate and distortion. With those concepts firmly in our grasp, we will then move on to a review of quantization, where we will focus on fixed-rate quantizers. Once this groundwork has been laid, the consistency of empirically designed quantizers will be introduced. We will then review both upper and lower bounds on the performance of empirically designed quantizers, the latter of which will be stated in a minimax sense. Due to the minimax nature of these lower bounds, we are motivated to investigate the actual performance of empirically optimal quantizers for specific distributions.

2.1 Data Compression, Rate and Distortion

We will begin our discussion of data compression with a look at the classical point-to-point communication model. As shown in Figure 2.1, the input data passes through

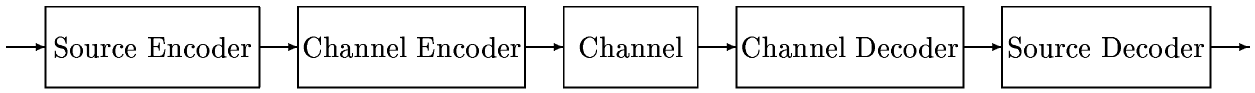

Figure 2.1: Model of a communications system.

multiple encoding steps prior to transmission and multiple decoding steps after it has been received. The first encoding step is performed at the Source Encoder. The purpose of the source encoder is to remove any redundancy from the input data, and hence compress the data to be transmitted. Once this compression step has taken place, the data is passed to the Channel Encoder, where its resiliency to channel noise is increased. The data is then transmitted across a potentially noisy channel. At the receiver, channel and source decoding are performed in series in order to reconstruct the original data.

In this work, our focus is on source encoding. As stated above, the role of the source encoder is to remove any redundancy in the data, thereby compressing it. More explicitly, the source encoder is where data compression takes place.

Data compression is the process by which a data set is replaced by a smaller representation of itself. There are two broad categories into which data compression schemes may fall: lossless and lossy. The type of scheme used depends entirely on the application being considered. In lossless compression, the original data is completely recoverable from the compressed data. Thus, lossless data compression is ideal for the compression of text, for example. In contrast to this, lossy compression involves an irreversible process, where the original data is not completely recoverable. Lossy compression finds applications in multimedia, for example, as well as in other areas where complete recovery of the original data is unnecessary. Our focus in this work is on lossy compression. Before proceeding, we introduce some basic notation.

We let the data set to be compressed be represented by the N-tuple $Z_1^N = (Z_1, Z_2, ..., Z_N)$, where N is the cardinality, or size, of the data set. The elements in the data set are assumed to be independent and identically distributed (i.i.d.). Furthermore, for generality, we assume that each element in the data set is an n-dimensional vector, that is $Z_i \in \mathbb{R}^n$, for $i = 1, \ldots, N$. We will now consider the performance measures used to assess the worth of a lossy compression scheme.

There are two ways to assess the performance of lossy compression schemes. The first performance measure is called *rate*. The rate of a compression scheme is defined as the expected number of bits required to represent an element of the original data set. Thus, rate is a measure of the degree to which the data is compressed. Since we are dealing with fixed-rate compression schemes, the expected number of bits required is actually the exact number of bits required. That is, each Z_i, when compressed, is represented by the same number of bits. As intuition suggests, our goal is to minimize the rate, thereby minimizing the number of bits required to represent each Z_i. Since the compressed data is generally stored digitally or in a computer's memory, the length of each representative vector is measured in bits, and so we will measure rate in units of *bits per sample.*

The second performance measure we must consider is called *distortion.* By definition, any lossy compression scheme will result in reconstruction values that differ from the original values. The distortion of a compression scheme is a quantitative measure of the difference between these two sets of values. A *distortion measure*, denoted $d(\cdot, \cdot)$, is a function that quantifies the distortion incurred when a sample is quantized. This function, which takes the sample to be compressed, Z_i, as its first argument and the representation of that sample, $\hat{Z}_i$, as its second argument, always returns a non-negative value. For a given compression scheme, the distortion, D, is defined as follows

$$D = E\left[d\left(Z_i, \hat{Z}_i\right)\right]. \tag{2.1}$$

One of the simplest and most ubiquitous distortion measures is the mean squared error (MSE), which is defined as the average of the sum of the squares of the Euclidean distance from each original data point to its reconstruction point. Formally, if we let $\{\hat{Z}_i\}_{i=1}^{N}$ be the reconstruction values for each of the elements in that data set, then

$$MSE = E\left[\left(Z_i - \hat{Z}_i\right)^2\right]. \tag{2.2}$$

In practice, we calculate the empirical MSE, denoted MSE_N, via

$$MSE_N = \frac{1}{N}\sum_{i=1}^{N}\left(Z_i - \hat{Z}_i\right)^2. \tag{2.3}$$

Equation (2.3) was used throughout our simulations to determine the training and test distortions of the empirical quantizers we designed.

As was the case with rate, intuition leads us to conclude that minimal distortion is desirable. It should be noted that different types of data have different acceptable levels of distortion. For example, a very low level of distortion is required when compressing medical images, while a much higher level of distortion is acceptable when compressing images for use on the Web.

It is crucial to consider both of the performance measures discussed above when designing a data compression scheme. Unfortunately, minimum rate and minimum distortion are contradicting requirements. If the rate is too small, there will be insufficient bits with which to represent each data point, and so the distortion will be high. Conversely, in order to make the distortion very low, we must be able to represent a lot of information about each sample, which requires a larger number of bits, and hence a high rate. It is important to find a balance between distortion and rate for which both have acceptable values for the application under consideration. Typically, we impose either a rate or distortion constraint, depending on the application, and then attempt to minimize the other measure, given the constraint [16]. In our work, we attempt to minimize distortion given prescribed values for the rate.

Since the problem of optimal quantizer design depends on the distribution of the input data, where each input sample is drawn from the same distribution, we need not consider all Z_i. Instead, we introduce a new entity called a *source* [10]. A source is defined to be a random vector, drawn from the same distribution as the input data. We will denote a source by X.

As stated above, the goal of any quantizer design algorithm is to minimize the expected distortion when it is used to quantize the true source, while keeping the rate low. In some cases, the source distribution is known, and so it is possible to design an optimal quantizer via optimization procedures. Unfortunately, in most cases of practical significance, the source distribution is either not available or is so complex that it makes the optimization problem prohibitively difficult. In such instances, it is common practice to provide a set of random samples drawn from the source distribution as input to the algorithm. Such a set is called a *training set.* The idea behind this is that a sufficiently large training set, if drawn truly at random, will provide a good representation of the actual source. If the training set is too small, the resulting quantizer will perform extremely well when used on the same training set, but will perform quite poorly when used to quantize an independent test set.

With this comes a new set of concerns, however. Firstly, how well can a quantizer designed in this way perform? More specifically, as the number of training samples increases, will an optimal quantizer be designed? Secondly, what size of training set is required to produce a sub-optimal quantizer with a prescribed level of performance? In the following sections, we explore some theoretical aspects of these problems.

2.2 The Problem of Quantization

In the previous section, we addressed the ideas of data compression, rate and distortion. In particular, we addressed the concept of quantization and introduced the topic of optimal quantizer design. In the current section, we elaborate on the topic of quantization and review sufficient theory to justify the existence of optimal quantizers. The discussion presented in this section is based primarily on [10].

As mentioned above, we employ a lossy data compression scheme. More specifically, we are interested in fixed-rate quantization. Quantization is informally defined as the process of taking values drawn from a large set of numbers and representing them with values drawn from a smaller set of numbers. In *vector quantization*, each sample is an n-dimensional vector drawn from the source distribution, where $n \geq 1$. Hence, the source is in fact an n-dimensional random vector. For the special case where $n = 1$, the process is called *scalar quantization.* In fixed-rate vector quantization, each representative vector, or *codevector*, is represented by the same number of bits, and hence has the same rate.

Before proceeding, we introduce some new terms to facilitate our discussion. The collection of codevectors to which all samples can be mapped is called the *codebook.* It is denoted $\mathcal{C}$ and is given by $\mathcal{C}= \{y_1, \dots, y_K\}$, where each y_i is a point in $\mathbb{R}^n$ called a codevector. The i^{th} decision region, denoted S_i, is given by $S_i = \{x|q(x) = y_i\}$, $i = 1, \dots, K$. Hence, a K-level quantizer has K decision regions. Furthermore, the decision regions for a given quantizer do not intersect and their union is all of $\mathbb{R}^n$. Hence, they form a partition of $\mathbb{R}^n$.

In order to describe a K-level quantizer, q, it is sufficient to specify its codebook and its decision regions. A simple rule exists which illustrates this point: $q(x) = y_i$ iff $x \in S_i$, where y_i is the i^{th} codevector and S_i is its corresponding decision region [10].

It follows directly from the definition of fixed-rate vector quantizers that each codevector is represented by the same number of bits. Futhermore, if we assume the cardinality of the codebook to be K, that is we are interested in K-level quantizers, then we have the following expression for the rate of our quantizer

$$R = \log_2 K \tag{2.4}$$

where we take the base of the logarithm to be two because we are using bits. It should be noted that it is convention to define the rate of a quantizer in this way, even when K is not a power of 2 [10].

The other performance measure, namely distortion, will now be addressed. Recall, the distortion of a compression scheme is given by (2.1). Substituting the source, X, into this equation gives

$$D(\mu, q) = E\left[d\left(X, q(X)\right)\right] \tag{2.5}$$

where the dependence of D on the quantizer in question, q, and on the distribution of the source, μ, is made explicit.

Using the definition of expected value, (2.5) can be rewritten as

$$D(\mu, q) = \int_{\mathbb{R}^n} d\left(x, q(x)\right) \mu(dx). \tag{2.6}$$

This, in turn, can be simplified further by substitution of the mean squared error distortion measure for $d\left(x, q(x)\right)$

$$D(\mu, q) = \int_{\mathbb{R}^n} \|x - q(x)\|^2 \mu(dx). \tag{2.7}$$

We now define the *optimal distortion* of a K-level quantizer to be the distortion of the optimal quantizer for the source. Since our goal is to minimize distortion, the quantizer with optimal distortion is the quantizer with minimum distortion. Such a quantizer is referred to as an *optimal quantizer* and is denoted q^*. We now formalize this notion via the following relationship

$$D_K^*(\mu) = \inf_{q \in Q_K} D(\mu, q) \tag{2.8}$$

where $D_K^*(\mu)$ denotes the optimal distortion of K-level quantizers for the source, μ, and Q_K denotes the set of all K-level quantizers on μ. It follows immediately that any quantizer is optimal if its distortion is equal to the optimal distortion for the source. That is, q^* is an optimal K-level quantizer on μ if

$$D(\mu, q^*) = D_K^*(\mu) . \tag{2.9}$$

2.2.1 Existence of Optimal Quantizers

We have now developed sufficient notation to address the first key point of this chapter. That point is the following: it is always possible, in theory, to design a quantizer whose distortion is optimal for a given source. Before stating this result more formally as a theorem, we will consider a pair of conditions that will simplify our discussion.

The first condition presented here is the Nearest Neighbour Condition (NNC). The NNC states that, given two K-level quantizers, $q^{(1)}$ and $q^{(2)}$, with the same codebook, $\mathcal{C}$, where $q^{(1)}$ is taken to be arbitrary and $q^{(2)}$ is defined by the following rule

$$q^{(2)}(x) = \arg\min_{y_i \in C} \|x - y_i\|^2 \tag{2.10}$$

then the distortions of $q^{(1)}$ and $q^{(2)}$ obey the following relationship

$$D\left(\mu, q^{(2)}\right) \leq D\left(\mu, q^{(1)}\right). \tag{2.11}$$

This condition says that any quantizer whose decision regions are based on (2.10) must have a distortion that is less than or equal to the distortion of any other quantizer for that source.

The second condition that we will consider is called the Centroid Condition (CC). This condition states that if we let $q^{(1)}$ and $q^{(2)}$ be two quantizers with the same decision regions, $\{S_i\}_{i=1}^{K}$, where $q^{(1)}$ has arbitrary codevectors and $q^{(2)}$ has codevectors $y_i^{(2)}$ given by

$$y_i^{(2)} = E\left[X | X \in S_i\right], \quad i = 1, \ldots, K \tag{2.12}$$

then the distortions of the two quantizers obey the following relationship

$$D\left(\mu, q^{(2)}\right) \leq D\left(\mu, q^{(1)}\right).$$

Thus, the second condition says that any quantizer whose codevectors are based on (2.12) must have a distortion that is less than or equal to the distortion of any other quantizer for that source.

Quantizers that satisfy both the NNC and the CC are called *Lloyd-Max quantizers.* An algorithm for finding such quantizers was developed by Lloyd and Max, using a steepest descent approach. Unfortunately, Lloyd-Max quantizers are locally optimal, but are not necessarily globally optimal [10].

Using the NNC, we now define a special type of quantizer called the *nearest neighbour quantizer.* A K-level quantizer, q, is a nearest neighbour quantizer if it satisfies the NNC.

Since it satisfies the NNC, (2.11) tells us that, if an optimal quantizer exists, then an optimal nearest neighbour quantizer exists as well. This brings us to our first theorem.

Theorem 1 *There exists a nearest neighbour quantizer, $q^* \in Q_K$, such that $D(\mu, q^*) = D_K^*(\mu)$ [10].*

This theorem implies that an optimal K-level nearest neighbour quantizer always exists. Furthermore, this tells us that an optimal K-level quantizer always exists.

2.2.2 Empirical Design

In the last section, we showed that optimal K-level quantizers exist. In the current section, we will see that we can, in theory, design these optimal quantizers.

As stated above, it is often impractical, or even impossible, to design a quantizer using the true source. In such cases, a training set consisting of random samples drawn from the source distribution is used. We begin this section by defining some helpful new notation, as well as some new quantities.

We represent a training set by $Z_1^N = Z_1, \ldots, Z_N$, where the Z_i are i.i.d. We denote the quantizer constructed using Z_1^N by $q_N = q_N(\cdot, Z_1, \ldots, Z_N)$. Such a quantizer is called an *empirically designed quantizer.*

There are multiple distortion measures associated with an empirically designed quantizer. The first one we will consider is called the *test distortion.* Test distortion is defined as the distortion incurred when an empirically designed quantizer is used to quantize a large test set that is assumed to be representative of the true source. Formally, it is denoted $D(\mu, q_N)$ and is given by

$$D(\mu, q_N) = E\left[\|X - q_N(X)\|^2 | Z_1^N\right]. \tag{2.13}$$

If we let the test set be represented by $Y_1^{N_{test}} = \{Y_1, \ldots, Y_{N_{test}}\}$, where N_{test} is the size of the test set, then we can calculate the test distortion according to the following relationship

$$D(\mu, q_N) = \frac{1}{N_{test}} \sum_{k=1}^{N_{test}} \|Y_k - q_N(Y_k)\|^2 \tag{2.14}$$

The next type of distortion we will consider is called *training distortion.* This quantity is a measure of the distortion incurred when the data being quantized is the same data from which the quantizer was designed. It is given by

$$D(\mu_N, q_N) = \frac{1}{N} \sum_{k=1}^{N} \|Z_k - q_N(Z_k)\|^2 \tag{2.15}$$

where μ_N is the empirical distribution of the training set. Both test and training distortion are random variables since they depend on the quantizer, q_N, which in turn depends on the training set, Z_1^N.

Using this, we can now define the *empirically optimal quantizer*, q_N^* as follows

$$q_N^* = \arg\min_{q \in Q_K} \frac{1}{N} \sum_{k=1}^{N} \|Z_k - q(Z_k)\|^2. \tag{2.16}$$

Thus, the empirically optimal quantizer satisfies $D(\mu_N, q_N^*) = \inf_{q \in Q_K} D(\mu_N, q) = D_K^*(\mu_N)$. Clearly, given a training set, an empirically optimal quantizer always exists.

Theorem 2 *For any $K \geq 1$, the sequence of empirically optimal K-level quantizers q_N^*, $N = 1, 2, \ldots$ satisfies*

$$\lim_{N \to \infty} D(\mu, q_N^*) = D_K^*(\mu). \tag{2.17}$$

Theorem 2 is due to [15].

This theorem implies that, for large N, it is reasonable to expect that the distribution of the training set, μ_N, provides a reasonable approximation for the actual distribution of the source, μ. This result implies that as N becomes very large, it is possible to design a quantizer that is optimal for the source based on an empirical design. The question now becomes, how large is large?

Before moving on to consider some performance bounds, it should be noted that the training distortion also provides a very good estimate of the optimal distortion as N becomes large. However, this estimate is optimistic, since the quantizer is designed to fit the training set, which is not be perfectly representative of the source. This bias can be seen by considering the expected training distortion

$$\begin{aligned} E\left[D\left(\mu_N, q_N^*\right)\right] &\leq E\left[D\left(\mu_N, q^*\right)\right] \\ &= E\left\{\frac{1}{N}\sum_{k=1}^{N}\|Z_k - q^*(Z_k)\|^2\right\} \\ &= D\left(\mu, q^*\right) \\ &= D_K^*\left(\mu\right). \end{aligned} \tag{2.18}$$

This inequality is strict unless the source distribution is discrete and is concentrated on K points or less [9].

In this section, we saw that both the training distortion and the test distortion of empirically optimal quantizers provide accurate estimates for the distortion of a quantizer that is optimal for the true source as N becomes arbitrarily large. However, we have yet to see how these quantizers perform when finite training sets are used. The next section outlines several bounds that help to identify the type of performance we should expect for all values of N.

2.2.3 Bounds on Quantizer Performance

By Theorem 2, as the size of the training set approaches infinity, the test distortion of the empirically optimal quantizer approaches the optimal value. Similarly, the training distortion also approaches the optimal value as N approaches infinity. However, training data can be expensive. Also, the computation time required to design a quantizer using empirical techniques can become prohibitively large when N is allowed to increase without bound. For these reasons, it is imperative that we determine the rate at which the distortion of an empirically optimal quantizer converges to the distortion of the optimal quantizer for the source as a function of training set size. Knowing this will enable us to determine how many training samples are actually required for a given application.

To simplify the problem addressed in this section, only source distributions with bounded support are considered. That is, $P\left(\|X\| \leq M\right) = 1$ for some bound M. We also introduce some new notation here.

Let $b(M)$ be a ball of radius M in $\mathbb{R}^n$ centered at the origin. Let $\mathcal{M}\ (M)$ be the set consisting of distributions supported only on $b(M)$. Let $Q_K(M)$ be the set consisting of all K-level nearest neighbour quantizers whose codevectors all lie inside $b(M)$. With these definitions in hand, we now present the following theorem.

Theorem 3 *There exists a positive constant, $C = C(n, K, M)$, such that for all $N \geq 1$ and all $\mu \in \mathcal{M}(M)$ the following inequality holds*

$$E\left\{\sup_{q \in Q_K(M)} \left|D(\mu_N, q) - D(\mu, q)\right|\right\} \leq \frac{C}{\sqrt{N}}\ . \tag{2.19}$$

Theorem 3 is due to [10].

This result means that the expected largest difference between the distortions of an arbitrary quantizer in $Q_K(M)$ when used to quantize a set of size N and the true source is bounded from above by $\frac{1}{\sqrt{N}}$ times a constant. Futhermore, the constant depends only on the dimension of the codevectors, the number of codevectors, and the bound described above. This important result leads us to our first performance bound.

Theorem 4 *For all K and M, there exists a real-valued quantity $C_1 = C_1(K, M)$ such that for all training set sizes $N \geq 1$ and all source distributions $\mu \in \mathcal{M}(M)$, the following bound holds*

$$E\left[D\left(\mu, q_N^*\right)\right] - D_K^*(\mu) \leq \frac{C_1}{\sqrt{N}} \ . \tag{2.20}$$

This bound can be restated as follows

$$D_K^*(\mu) \leq E\left[D\left(\mu, q_N^*\right)\right] \leq D_K^*(\mu) + \frac{C_1}{\sqrt{N}} \tag{2.21}$$

where we make explicit the fact that the expected test distortion is bounded from below by the optimal distortion for the source. Theorem 4 is due to [9].

This theorem says that the difference between the expected test distortion of an empirically optimal quantizer and the actual optimal distortion is bounded from above by $\frac{1}{\sqrt{N}}$ times a constant. Thus, we now have an upper bound on the test distortion of empirically optimal quantizers that tightens with increasing N.

As was addressed earlier, there is another important type of distortion to be considered when designing empirically optimal quantizers, namely the training distortion. This distortion also provides an estimate of the optimal distortion for large N, and so a bound on it is sought as well. Fortunately, such a bound exists, and is stated in the following theorem.

Theorem 5 *There exists a real-valued quantity, $C = C(K, M)$, such that for all $N \geq 1$ and all $\mu \in \mathcal{M}(M)$ the following inequality holds*

$$D_K^*(\mu) - E\left[D\left(\mu_N, q_N^*\right)\right] \leq \frac{C}{\sqrt{N}}\ . \tag{2.22}$$

This bound can be restated as follows

$$D_K^*(\mu) - \frac{C}{\sqrt{N}} \leq E\left[D\left(\mu_N, q_N^*\right)\right] \leq D_K^*(\mu) \tag{2.23}$$

where we make explicit the fact that the expected training distortion is bounded from above by the optimal distortion for the source. Theorem 5 is due to [10].

This theorem states that the difference between the expected training distortion of the empirically optimal quantizer and the optimal distortion for the source is bounded from above by $\frac{1}{\sqrt{N}}$ times a constant. Two important observations should be made regarding the statement presented in this theorem. Firstly, the difference between the expected training distortion and the optimal distortion is bounded from above by a decreasing function of N for all positive training set sizes. Secondly, the expected training distortion of an empirically optimal quantizer is always smaller than the optimal distortion, giving us evidence of the bias discussed earlier.

Recently, work presented in [6] has shown that the upper bound on the test distortion can indeed by tightened for certain classes of source distributions, asymptotically speaking at least. Their relevant results are summarized in the following theorem.

Theorem 6 *Let X be a scalar random variable with a strictly log-concave density, f, supported on [-B,B]. Then,*

$$E\left[D(\mu, q_N^*)\right] - D_K^*(\mu) = \mathcal{O}\left(\frac{\log N}{N}\right). \tag{2.24}$$

The first important condition to note is that this theorem applies only to scalar random variables. This is in contrast to the other bounds presented in this chapter. Also, the statement in this theorem applies only to random variables whose densities are log-concave and whose supports are bounded. Thus, it applies to the uniform distribution, but not to the other distributions studied in our work. Technically, though, the Gaussian and Laplacian densities studied here have bounded supports due to the finite precision with which computers store data, and so this theorem applies to them as well. For such distributions, this theorem implies a faster rate of convergence than that guaranteed by the more general upper bound presented above.

The quantity on the left hand side of (2.24) is explicitly related to N via the following inequality

$$E\left[D(\mu, q_N^*)\right] - D_K^*(\mu) \le \frac{C_1 \log N}{N} + \frac{C_2}{N}. \tag{2.25}$$

Unfortunately, the constants in (2.25) are defined non-explicitly and depend in a complicated way on the globally optimal quantizer. For this reason, the bound presented in Theorem 6 should be accepted with caution. If the constants, C_1 and C_2, are very large, then the bound may be of little use in practice.

So far, we have considered upper bounds on the performance of empirically optimal quantizers. These results showed that the difference between both the training and test distortions and the optimal distortion is bounded from above by a decreasing function of N. However, we wish to know even more about the performance of these quantizers. The purpose of the remainder of this section is to present lower bounds on these differences which illustrate that, in fact, unless the source distribution is known a priori, it is impossible to guarantee a convergence rate that is faster than that presented above, namely $\frac{C}{\sqrt{N}}$, for $K > 1$.

We begin this discussion by stating two interesting results. Consider an empirically optimal quantizer, q_N^*, where $K = 1$. That is, q_N^* is a 1-level quantizer. It is easy to see that the expected test distortion of this quantizer is given by

$$E\left[D\left(\mu, q_N^*\right)\right] = D_1^*(\mu) + \frac{D_1^*(\mu)}{N} \ . \tag{2.26}$$

This observation is interesting because it shows that, for a 1-level empirically optimal quantizer, the difference between the expected test distortion and the optimal distortion is bounded by $\frac{1}{N}$. This implies a much faster rate of convergence than that guaranteed by Theorem 4. We now turn our attention to the expected training distortion of q_N^*, which satisfies the following relationship

$$E\left[D\left(\mu_N, q_N^*\right)\right] = D_1^*(\mu) - \frac{D_1^*(\mu)}{N} \ . \tag{2.27}$$

This observation shows that a convergence rate proportional to $\frac{1}{N}$ is also attainable for the training distortion. Unfortunately, these results apply only to the case where $K = 1$ [10]. In fact, for $K > 2$, it is not possible to guarantee convergence rates faster than $\frac{1}{\sqrt{N}}$ for an arbitrary distribution. The following theorems, both of which are taken from [10], show that, for $K > 2$, a convergence rate proportional to $\frac{1}{\sqrt{N}}$ is the best we can guarantee.

Theorem 7 *For any empirically designed K-level ($K > 2$) quantizer, q_N, designed using a training set of size $N \geq N_0$ training vectors,*

$$\sup_{\mu \in \mathcal{M}(M)} E\left[D\left(\mu, q_N\right)\right] - D_K^*(\mu) \geq \frac{C_2}{\sqrt{N}} \tag{2.28}$$

The value of N_0 for which this result holds depend on the number of levels, K, and on the constant $C_2 = C_2(n, K, M) > 0$. This theorem states that, for each value of $N > N_0$, there exists at least one distribution $\mu \in \mathcal{M}(M)$ such that the difference between the test distortion of an empirically designed quantizer and the optimal distortion for that source is bounded from below by $\frac{1}{\sqrt{N}}$ times a constant. Interestingly, as N changes, so does the distribution for which this difference is a maximum [9]. That is, in a minimax sense, this is the fastest convergence rate. This in turn tells us that, for a source with an arbitrary distribution in $\mathcal{M}(M)$, we cannot ensure a convergence rate faster than $\frac{1}{\sqrt{N}}$. This is an important result because, unlike the upper bounds discussed earlier, this bound is derived without knowledge of the employed method of empirical design [10]. The bound presented in Theorem 7 is weak, however, in the sense that it does not tell us very much about the actual performance of empirical quantizers for a given source. A similar bound for the training distortion will now be presented.

Theorem 8 *For any empirically optimal K-level ($K > 2$) quantizer, q_N^*, designed using a training set of size $N \geq N_0$ training vectors,*

$$\sup_{\mu \in \mathcal{M}(M)} D_K^*(\mu) - E\left[D\left(\mu_N, q_N^*\right)\right] \geq \frac{C_3}{\sqrt{N}} \tag{2.29}$$

where, again, $N_0 = N_0(K)$ and $C_3 = C_3(n, K, M) > 0$.

This theorem is from [9].

This theorem states that we can always find a distribution in $\mathcal{M}(M)$ for which the difference between the optimal distortion for the source and the training distortion of an empirically optimal quantizer is bounded from below by $\frac{1}{\sqrt{N}}$ times a constant. Unlike the case of Theorem 7, here we can find a single distribution that performs poorly for all values of $N \geq N_0$ [10].

The results presented in this section demonstrate that the convergence rates of both the test and training distortions of an empirically optimal quantizer to the optimal distortion for the source are bounded from below by $\frac{1}{\sqrt{N}}$ times a constant. However, since the bounds are minimax in nature, they do not necessarily hold for all source distributions. It has been demonstrated in [6] that an upper bound proportional to $\frac{\log N}{N}$ holds for sources whose distributions meet certain criteria. We seek to investigate these bounds via simulation for several commonly used distributions. For our simulation results, see Chapter 4. In the following section, we consider several methods for the design of empirically optimal quantizers.

2.3 Optimal Quantizer Design

The goal of empirical quantizer design algorithms is to partition the sample space, $\mathbb{R}^n$, into K decision regions, each with its own representative vector, y_i, $i = 1, \ldots, K$. This is done by minimizing the expected distortion, given a set of training vectors. As stated earlier in this chapter, a method exists for finding locally optimal vector quantizers. Unfortunately, no method is known for the design of globally optimal vector quantizers. However, efficient algorithms for the design of globally optimal scalar quantizers are known. It is for this reason that we restrict ourselves to the design of empirically optimal quantizers for which $n = 1$. That is, we are concerned with the design of empirically optimal scalar quantizers.

Any scalar quantizer is characterized by its codebook, $C = \{y_1, \ldots, y_K\} \subset \mathbb{R}$. The distortion of a quantizer is related to the elements of its codebook via

$$\begin{aligned} D(\mu, q) &= E\left[\min_{1 \le i \le K} \|X - y_i\|^2\right] \qquad (2.30) \\ &= g(y_1, \ldots, y_K). \end{aligned}$$

Thus, the distortion is actually a function of the codepoints. This function can be specified more precisely as $g : \mathbb{R}^K \to [0, \infty)$.

In light of this new information, the problem of optimal quantizer design becomes a minimization problem. More precisely, we wish to find $\min_{\mathbf{y}} g\,(y_1, \ldots, y_K)$.

There are several different methods by which to design empirically optimal quantizers. Depending on the time and resources available, as well as the application under consideration, some methods may be more desirable than others. In the discussion that follows, we outline several techniques that are used in practice. It should be noted that all of the techniques discussed below rely on the minimization of training distortion in order to find an empirically optimal quantizer.

2.3.1 The Brute Force Approach

In the present section, we consider the brute force method of scalar quantizer design. As its name suggests, this method is not elegant. Given a training set of size N, it computes all possible $K : N$ quantizers and returns the one with minimum training distortion as the empirically optimal quantizer. We will now consider this algorithm in more detail.

Before applying the quantizer design algorithm, it is prudent to sort the training data. By presenting the algorithm with sorted input, its complexity can be reduced substantially, at a negligible cost.

Since there are K decision regions, each containing a fraction of the N training points, a rough upper bound is given by N^K. In fact, it is easy to show that there are actually $\mathcal{O}(N^K)$ possible quantizers.

Thus, the brute force algorithm has an exponential runtime complexity. In the following section, we consider a more elegant approach that is much faster than the method considered above.

2.3.2 Two-Phase Algorithm

This basic algorithm, presented in [18], allows empirically optimal scalar quantizers to be designed in $O(KN^2)$ time. This is much faster than the asymptotic runtime of the brute force algorithm.

This method relies on the fact that optimal scalar quantizers exhibit monotonicity properties. It exploits these properties and, by applying dynamic programming techniques, is able to achieve a runtime complexity substantially lower than those of previously existing algorithms.

The remainder of this section is divided into two parts. First, we consider the notion of monotonicity and see how it applies to optimal scalar quantizers. Then, we outline the details of the algorithm proposed in [20] and analyze its runtime complexity.

Monotonicity of Optimal Scalar Quantizers

Before considering the monotonicity results on which this algorithm is based, we first introduce some new notation.

Let $Q_N^K = \{\mathbf{q} | 1 \leq q_1 < q_2 < \ldots < q_{K-1} < N\} \subset \mathcal{N}^{K-1}$. Q_N^K is the set of all K-level quantizers, where N is the number of points on which the pmf under consideration is defined and q_i is the position of the i^{th} decision boundary. For our purposes, N is the cardinality of the training set.

Let $P(x_i)$ be the discretized pdf, or pmf, of the source under consideration. Since we employ a training set of size N, we take $P(x_i) = \frac{1}{N}$.

Let x_i be the i^{th} training point.

Let y_i be the i^{th} codepoint.

Let $W(x_i, x_i - y_i)$ be an error weighting function. For our purposes, we take this to be the mean squared error measure, and so we replace it by $(x_i - y_i)^2$.

Let

$$\begin{aligned}\mu(a,b] &= \arg\min_y \sum_{a<i\leq b} P(x_i)\, W(x_i, x_i - y) \\ &= \arg\min_y \frac{1}{N} \sum_{a<i\leq b} (x_i - y)^2. \qquad (2.31)\end{aligned}$$

Thus, $\mu(a,b]$ is the representative point for the interval $(x_a, x_b]$.

Let

$$\begin{aligned}\epsilon(a,b] &= \sum_{a<i\leq b} P(x_i)\, W(x_i, x_i - \mu(a,b]) \\ &= \frac{1}{N} \sum_{a<i\leq b} (x_i - \mu(a,b])^2. \qquad (2.32)\end{aligned}$$

Thus, $\epsilon(a,b]$ is the distortion incurred over the interval $(x_a, x_b]$ when a single representative point is used.

Let

$$E(q) = \sum_{1\leq j\leq K} \epsilon(q_{j-1}, q_j] \qquad (2.33)$$

where $q \in Q_N^K$, be the distortion of the quantizer q. By convention, we take $q_0 = 0$ and $q_K = N$.

The most important monotonicity property is best expressed as a lemma. This lemma is presented here in the same form as in [20] in order to ensure that no subtleties are lost.

Lemma 1 *Given a discrete density $P(x_i)$ and $1 < t < k < n \leq N$, the subvector $(q_1, q_2, ..., q_{t-1})$ of an optimal $k : n$ quantizer $\mathbf{q} \in Q_n^k$ is itself an optimal $t : q_t$ quantizer in $Q_{q_t}^t$.*

This lemma says that the first $t-1$ codepoints of an optimal $k : n$ quantizer form an optimal $t : q_t$ quantizer. We will now consider the details of the algorithm.

Dynamic Programming Algorithm

The algorithm presented in [20] uses a dynamic programming approach to achieve fast scalar quantizer design. It is divided into three main phases. The first phase is dedicated to preprocessing. In the second phase, the preprocessed data is used to compute the location of the last decision boundary, denoted $h(k,n)$, and the distortion for each empirical $k:n$ quantizer for $2 \leq k \leq K$ and $k < n \leq N$. Finally, in the third phase, the results from phase two are used to find the empirically optimal $K:N$ quantizer.

As stated above, the first phase of the algorithm is devoted to preprocessing. More specifically, the 0^{th}, 1^{st}, and 2^{nd} cumulative moments of the approximate pmf, or training set distribution, are computed. Since we place equal weight on each training point, the pmf used in our work is equal to $\frac{1}{N}$ at each of the N points over which it is supported. The desired moments are computed according to the following relationships

$$M_0(i) = \frac{i}{N}, \quad i = 1, \ldots, N$$

$$M_1(i) = M_1(i-1) + \frac{X(i)}{N}, \quad i = 1, \ldots, N \tag{2.34}$$

$$M_2(i) = M_2(i-1) + \frac{(X(i))^2}{N}, \quad i = 1, \ldots, N$$

where M_j is an array storing the j^{th} cumulative moments, $X(i)$ is the i^{th} training point, and $M_j(0) = 0$ for $j = 0, 1, 2$. Since we compute $3N$ moments, phase one has an asymptotic runtime of $\mathcal{O}(N)$. We will now examine phase two and see how the values found in phase one are used.

Before proceeding with the details of the second phase, we provide formal definitions for the quantities to be computed. The first quantity of interest is the location of the final decision boundary for an empirically optimal $k : n$ quantizer. This is denoted $h(k,n)$, and is given by

$$h(k,n) = \max\left\{\tau \,\middle|\, \tau = \operatorname*{arg\,min}_{k<i<n}\{E(k-1,i) + \epsilon(i,n]\}\right\} \tag{2.35}$$

Another important quantity is the distortion of an empirically optimal $k : n$ quantizer. This is denoted $E(k,n)$ and is given by

$$E(k,n) = E(k-1, h(k,n)) + \epsilon(h(k,n), n] \tag{2.36}$$

Equation (2.35) shows the dependence of $h(k,n)$ on $E(k-1,i)$, $i \in \{k+1, \dots, n-1\}$. Equation (2.36) shows the dependence of $E(k,n)$ on $h(k,n)$. Since these quantities depend on each other, it is natural that they be computed together.

The first step is to calculate the distortion of all empirically optimal 1-level quantizers. Recall that since the training set is sorted, there are exactly N such quantizers. These distortion values are found according to

$$E(1,i) = M_2(i) - M_2(0) - \frac{(M_1(i) - M_1(0))^2}{M_0(i) - M_0(0)}, \quad i = 1, \dots, N. \tag{2.37}$$

Since $M_j(0) = 0$ for all j, this expression can be simplified to

$$E(1,i) = M_2(i) - \frac{(M_1(i))^2}{M_0(i)}, \quad i = 1, \dots, N. \tag{2.38}$$

In terms of runtime complexity, finding these distortions takes $\mathcal{O}(N)$ time.

Next, we compute $h(k,n)$ and $E(k,n)$ for $k = 2, \dots, K$ and $n = k, \dots, K$. For a given k, we perform the following routine

```
for n from k to N do
    $h(k,n) \leftarrow k+1$
```

```
$mindist \leftarrow E(k-1,k+1) + M_2(n) - M_2(k+1) - \frac{(M_1(n)-M_1(k+1))^2}{M_0(n)-M_0(k+1)}$
$for\ i\ from\ k+2\ to\ n\ do$
    $if\ \left(E(k-1,i) + M_2(n) - M_2(i) - \frac{(M_1(n)-M_1(i))^2}{M_0(n)-M_0(i)} < mindist\right)\ then$
        $h(k,n) \leftarrow i$
        $mindist \leftarrow E(k-1,i) + M_2(n) - M_2(i) - \frac{(M_1(n)-M_1(i))^2}{M_0(n)-M_0(i)}$
```

So, initially, we set the location of the final decision boundary to be in the leftmost possible position and calculate the total distortion associated with this choice. We then move the final decision boundary to the right, one position at a time. For each move, the total distortion is calculated. The final decision boundary that leads to minimal distortion is the value given to $h(k,n)$.

Once $h(k,n)$ has been found, for a given k, $E(k,n)$ can be obtained for $k \leq n \leq N$ as follows.

```
$for\ n\ from\ k\ to\ N\ do$
    $E(k,n) \leftarrow E(k-1,h(k,n)) + M_2(n) - M_2(h(k,n)) - \frac{(M_1(n)-M_1(h(k,n)))^2}{M_0(n)-M_0(h(k,n))}$
```

Thus, once $E(1,1..N)$ is calculated, $h(2,2..N)$ can be obtained. These values are in turn used to find $E(2,2..N)$. Using $E(2,2..N)$, we can then find $h(3,3..N)$, which allows us to find $E(3,3..N)$, etc.

In terms of runtime complexity, calculating the initial $E(1,1..N)$ values takes $\mathcal{O}(N)$ time. Computing the $h(k,n)$ values for a given k takes $\mathcal{O}(N^2)$ time, and computing the $E(k,n)$ for a given k takes $\mathcal{O}(N)$ time. Since k ranges from 2 to K, the overall runtime complexity of phase two is $\mathcal{O}(KN^2)$.

In the third and final phase of the algorithm, the values for $h(k,n)$ found in phase two are used to recover the empirically optimal $K:N$ quantizer.

We first define $qKN(i)$ to be the position of the first training point belonging to the

$(i+1)^{st}$ decision region. There are three values to initialize before proceeding. We set $qKN(0) = 1$ because $X(1)$ is the first training point. We then set $qKN(K) = N + 1$ because the $(K+1)^{st}$ level does not exist. Finally, we set $qKN(K-1) = h(K, N)$ because this is the location of the final decision boundary for the empirically optimal $K : N$ quantizer. The remainder of the decision boundaries are found via the following routine

```
for i from (K − 2) downto 1 do
    $qKN(i) \leftarrow h(i+1, qKN(i+1))$
```

That is, given that the final decision boundary of the empirically optimal $K : N$ quantizer equals $h(K, N)$, the second last decision boundary is equal to the the final decision boundary of the empirically optimal $(K-1) : h(K, N)$ quantizer, and so on.

Once all decision boundaries have been found, the codepoints are computed as the centroids of the decision regions.

The third phase of the algorithm has an asymptotic runtime of $\mathcal{O}(N)$. Thus, the overall runtime complexity of this algorithm is $\mathcal{O}(KN^2)$.

2.3.3 Fast Scalar Quantizer Design Using Matrix Search

The algorithm discussed in this section enables us to design empirically optimal scalar quantizers in $\mathcal{O}(KN \log N)$ time. This is substantially faster than the runtime of the algorithm presented in the previous section. In fact, in practice, we found that for $N = 50000$, the matrix search algorithm had a runtime about 30% as long as that of the previous algorithm.

We will first address the matrix search technique and see why it works. We will then consider the actual implementation used in this project.

Matrix Search

We begin by considering an upper triangular $N \times N$ matrix, $\mathcal{A}_{NN}$. Each entry of $\mathcal{A}_{NN}$, denoted a_{ij}, is the distortion of a k-level quantizer. More specifically, a_{ij} is the distortion of a k-level quantizer on the first j training points, whose final decision boundary occurs at the i^{th} training point. Clearly, the minimum entry in the j^{th} column, denoted $a^*_{\downarrow j}$, gives the distortion of the empirically optimal k-level quantizer on the first j training points. The index of the row to which $a^*_{\downarrow j}$ belongs is the location of the final decision boundary of this quantizer. This, however, is not sufficient, as, for each value of k, all entries in $\mathcal{A}_{NN}$ must be computed. Since $\mathcal{A}_{NN}$ is an $N \times N$ matrix, this leads to a runtime complexity of $\mathcal{O}(KN^2)$.

Fortunately, due to the monotonicity properties exhibited by empirically optimal scalar quantizers, we are not required to compute every entry of each matrix. Knowing the location of the minimum element in column j' eliminates large blocks of $\mathcal{A}_{NN}$ from contention in our search for the other column minima. More precisely, if $a^*_{\downarrow j'}$ is located in row i', then for $j < j'$, $a^*_{\downarrow j}$ must be located in a row with index $\leq i'$. Similarly, for $j > j'$, $a^*_{\downarrow j}$ must be located in a row with index $\geq i'$. That is, for $j < j'$, $a^*_{\downarrow j}$ is in $\mathcal{A}_{i_1..i',\ j_1..j'-1}$, and for $j > j'$, $a^*_{\downarrow j}$ is in $\mathcal{A}_{i'..i_2,\ j'+1..j_2}$, where i_1, i_2, j_1, and j_2 are the topmost, bottommost, rightmost and leftmost locations in the portion of the matrix that is currently under consideration.[14]. Using the notion of monotonicity to speed up our search, the following algorithm was implemented.

Algorithm

It should be noted that our algorithm finds the empirically optimal quantizer by minimizing the empirical squared error distortion, rather than the empirical mean squared error distortion.

The first step in our algorithm is to preprocess the training data. We first define and compute the following values

$$U(c) = U(c-1) + X(c), \quad c = 1, \ldots, N, \quad U(0) = 0 \tag{2.39}$$

$$V(c) = V(c-1) + (X(c))^2, \quad c = 1, \ldots, N, \quad V(0) = 0 \tag{2.40}$$

$$Q_{opt}(1, c) = \frac{U(c)}{c}, \quad c = 1, \ldots, N \tag{2.41}$$

$$D_{opt}(1, c) = V(c) - c\,(Q_{opt}(1, c))^2, \quad c = 1, \ldots, N \tag{2.42}$$

where $U(c)$ is the sum of the first c training points, $V(c)$ is the sum of the squares of the first c training points, $Q_{opt}(1, c)$ is the codepoint of the empirically optimal 1-level quantizer over the first c training points, and $D_{opt}(1, c)$ is the distortion associated with $Q_{opt}(1, c)$. The computation of these quantities takes $\mathcal{O}(N)$ time. The calculation of $D_{opt}(1, c)$ is not obvious, and so is shown here

$$\begin{aligned} D_{opt}(1, c) &= \sum_{i=1}^{c} (Z_i - q(Z_i))^2 \\ &= \sum_{i=1}^{c} \left(Z_i^2 - 2Z_i q(Z_i) + q^2(Z_i)\right) \\ &= \sum_{i=1}^{c} Z_i^2 - 2Q_{opt}(1, c) \sum_{i=1}^{c} Z_i + Q_{opt}^2(1, c) \sum_{i=1}^{c} 1 \\ &= V(c) - 2Q_{opt}(1, c)\,(cQ_{opt}(1, c)) + cQ_{opt}^2(1, c) \\ &= V(c) - cQ_{opt}^2(1, c). \end{aligned} \tag{2.43}$$

Once the preprocessing phase is complete, the main phase begins. In the main phase, we do the following

for k from 2 to K do

$top = k - 1$

$bottom = N - 1$

$left = k$

$right = N$

$msearch(top,\ bottom,\ left,\ right,\ k)$

where $msearch()$ is a recursive function that finds the empirically optimal quantizer. The values top, $bottom$, $left$, and $right$ are the boundaries of the matrix to be searched. The details of this function will now be discussed.

The function begins by choosing the middle column of the current portion of the matrix under consideration. The middle column is chosen according to

$$CI \leftarrow \left\lceil \frac{left\ +\ right}{2} \right\rceil . \tag{2.44}$$

We then move down column CI, starting at row top and ending at row $CI - 1$. For row i, where $i \in \{top, \dots, CI-1\}$, we calculate the distortion associated with placing the final decision boundary at location i according to

$$D = D_{opt}(k-1,i) + V(CI) - V(i) - \frac{[(CI \times Q_{opt}(1,CI) - i \times Q_{opt}(1,i))]^2}{CI - i}. \tag{2.45}$$

Again, the derivation of this expression is not entirely obvious, and so it is offered here.

$$\begin{aligned} D &= \text{distortion of } k\text{-level quantizer on } Z_1, \dots, Z_{CI} \text{ with } Z_{i+1}, \dots, Z_{CI} \in S_k \\ &= D_{opt}(k-1,i) + \text{distortion for final decision region} \\ &= D_{opt}(k-1,i) + \sum_{n=i+1}^{CI} (Z_n - q(Z_n))^2 \end{aligned}$$

$$
\begin{aligned}
&= D_{opt}(k-1,i) + \sum_{n=i+1}^{CI} Z_n^2 - 2\sum_{n=i+1}^{CI} Z_n q(Z_n) + \sum_{n=i+1}^{CI} q^2(Z_n) \\
&= D_{opt}(k-1,i) + \sum_{n=i+1}^{CI} Z_n^2 - 2Q_{i+1\to CI}\sum_{n=i+1}^{CI} Z_n + Q_{i+1\to CI}^2 \sum_{n=i+1}^{CI} 1 \\
&= D_{opt}(k-1,i) + V(CI) - V(i) - 2Q_{i+1\to CI}\left((CI-i)Q_{i+1\to CI}\right) + (CI-i)Q_{i+1\to CI}^2 \\
&= D_{opt}(k-1,i) + V(CI) - V(i) - 2(CI-i)Q_{i+1\to CI}^2 + (CI-i)Q_{i+1\to CI}^2 \\
&= D_{opt}(k-1,i) + V(CI) - V(i) - (CI-i)Q_{i+1\to CI}^2 \\
&= D_{opt}(k-1,i) + V(CI) - V(i) - (CI-i)\left[\frac{CI}{CI-i}Q_{opt}(1,CI) - \frac{i}{CI-i}Q_{opt}(1,i)\right]^2 \\
&= D_{opt}(k-1,i) + V(CI) - V(i) - \frac{\left[CI \times Q_{opt}(1,CI) - i \times Q_{opt}(1,i)\right]^2}{CI-i}
\end{aligned}
$$

where $Q_{i+1\to CI}$ is the centroid of training points $(i+1)$ through CI.

The minimum distortion value is $D_{opt}(k, CI)$ and the location of the final decision boundary leading to this minimum distortion is $h(k, CI)$. These correspond to the distortion and final decision boundary for an empirically optimal $k : CI$ quantizer.

If there are still unsearched columns to the left of the column CI, then $msearch()$ is called recursively on the upper left submatrix with $top = top$, $bottom = h(k, CI)$, $left = left$, and $right = CI - 1$. If there are still unsearched columns to the right of column CI, then $msearch()$ is called recursively on the lower right submatrix with $top = h(k, CI)$, $bottom = bottom$, $left = CI + 1$, and $right = right$.

We will now consider the runtime complexity of this algorithm. As was the case for the algorithm presented in the previous section, the preprocessing phase takes $\mathcal{O}(N)$ time.

For the second phase, we examine the runtime complexity of the function $msearch()$. This function begins by selecting the "middle" column of the matrix. In order to find the minimum in that column, it must make approximately $\lceil \frac{N}{2} \rceil$ calculations. Once the minimum is found, $msearch()$ is called on the "middle" columns of the appropriate submatrices to the left and to the right of the original "middle" column. To find the minima in each of these columns, a total of approximately N calculations are made. We then call $msearch()$ on up to four more submatrices, where again, due to the monotonicity properties of the matrix, a total of approximately N calculations are done. Since the number of columns investigated doubles with every iteration, approximately $\log_2 N$ iterations are required. Since up to N calculations are performed at each iteration, it takes $\mathcal{O}(N \log N)$ time to find all column minima. So, $msearch()$ runs in $\mathcal{O}(N \log N)$ time. Now, $msearch()$ must be called from the main program once for each value of k, from 2 to K. Thus, the runtime complexity of the second phase is $\mathcal{O}(KN \log N)$.

The third phase has an asymptotic runtime of $\mathcal{O}(N)$, and so the overall runtime of our algorithm is $\mathcal{O}(KN \log N)$.

Chapter 3

Design and Implementation

The goal of this project is to study the rate at which the distortion of empirically optimal scalar quantizers converges to the distortion of the theoretical optimum quantizer for a given source. In order to accomplish this, several design and implementation issues must be considered. Firstly, the generation of training and test sets from the appropriate distributions must be dealt with. Secondly, factors such as the rates to be considered and the programming language to be used must considered. Finally, once the results are obtained, they must be properly interpreted so that accurate conclusions can be drawn. In this chapter, we will address these issues.

3.1 Training and Test Sets

Once the quantizer has been designed, it is important to test it to determine how well it performs. By the same argument presented above, it is often impractical, or even impossible, to test the quantizer using the true source. To circumvent this obstacle, a large set of random samples is drawn from the source distribution and used to determine the quantizer's performance. Such a set is called a *test set*. It is crucial that the training and test sets be statistically independent in order to avoid biasing the results. In this section, we consider the issues relevant to the design and implementation of these sets.

3.1.1 Pseudo-Random Number Generation

The first issue we will examine is that of pseudo-random number generation. Ideally, a training (or test) set is a sequence of truly random numbers drawn from the source distribution. However, computers cannot generate truly random numbers. In order to overcome this problem, a pseudo-random number generator is used.

Before moving on to discuss how such a generator works, it is important to understand the difference between truly and pseudo-random numbers. Pseudo-random numbers are generated in a sequence. Due to the way in which they are generated, once a predetermined number of initial values is known, the rest of the numbers in the sequence can be correctly predicted with probability 1. In the case of truly random numbers, each number is independent of each other number, and so there is no way of obtaining one number given another. It is for this reason that pseudo-random numbers are not truly random. However, the numbers in a pseudo-random sequence exhibit some of the most important empirical behaviour of a truly random sequence [7], and so are adequate for simulation purposes.

There are several types of commonly used pseudo-random number generators. All of these are defined by a recursive relationship between two or more consecutively generated numbers as follows

$$x_n = f(x_{n-1}, x_{n-2}, \dots) \tag{3.1}$$

where x_n is the current number in the sequence, and $x_{n-1}, x_{n-2}, \dots$ are the previously generated numbers in the sequence [7]. Thus, given the function, f, and an initial value,

x_0, or a finite sequence of initial values, $x_0, \ldots, x_m$ where $m < \infty$, the entire sequence can be obtained. That is, a deterministic sequence of numbers is generated [7]. We will focus on the case where a single initial value is sufficient. The number x_0 is referred to as the *seed*, and the sequence of numbers corresponding to x_0 is called a *stream*. It should be noted that some generators can produce multiple streams, with each stream having its own seed.

The pseudo-random number generator used to produce the training and test sets required for this project is a prime modulus multiplicative linear congruential generator. We will now discuss exactly what this means.

A linear congruential generator (LCG) has the form

$$x_n = (ax_{n-1} + b) \mod m \tag{3.2}$$

where a is a constant called the multiplier, b is an additive constant, and m is called the modulus. A multiplicative LCG, or MLCG, is an LCG where the additive constant b is equal to zero. That is, an MLCG involves only multiplication, and not addition. Hence, an MLCG has the form

$$x_n = (ax_{n-1}) \mod m. \tag{3.3}$$

The particular generator implemented in this project can now be obtained by substituting $a = 630360016$ and $m = 2^{31} - 1$ into (3.3), and is thus given by

$$x_n = (630360016x_{n-1}) \mod 2^{31} - 1. \tag{3.4}$$

Since the modulus, m, is prime in this case, the generator is referred to as a prime modulus multiplicative LCG. The C code for this generator is available online at [8]. This particular generator has the capacity to generate 100 streams, each of length 100000, which is adequate for the purposes of this project. Another important property of this generator is that it generates decimal numbers in the range (0,1), which makes it ideal for reasons that will now be discussed.

3.1.2 The Generation of Random Variates

As stated in Chapter 2, we wish to investigate the performance of empirically optimal quantizers designed for several commonly used distributions. These distributions are uniform, Gaussian, or more specifically, standard normal, Laplacian, and beta. Thus, training and testing sets were drawn from all four of these distributions. We will now discuss the methods by which values drawn from each of these distributions are generated. In the discussion that follows, we will use the term *data set* to denote both training and test sets.

Uniform Random Variates

In this work, we are concerned with the uniform distribution on the interval (0,1). The generation of data sets drawn from this distribution is straightforward. The pdf of the uniform distribution, denoted $f_u(x)$, is given by

$$f_u(x) = \begin{cases} 1 & x \in [0,1] \\ 0 & \text{otherwise} . \end{cases} \tag{3.5}$$

Since the pseudo-random number generator returns uniformly distributed values in the range (0,1), the values it returns can be taken as the sought random values and so no manipulation of the data is required. It should be noted that, due to the simplicity of the pdf in (3.5), the actual optimal quantizer and the optimal distortion for the source can be calculated. In order to simplify our search for this optimal quantizer, we rely on the following theorem.

Theorem 9 *Let a given continuous probability density of finite second moment obey the inequality*

$$\frac{d^2}{dx^2} \ln p(x) < 0$$

over its support. Then, the mean squared distortion of a K-level quantizer has a unique stationary point.

This theorem, which is due to [3], applies to all distributions with bounded support that are log concave, including the uniform distribution. The theorem implies that, for such distributions, there is a unique optimal quantizer. As we saw earlier, if there exists an optimal quantizer, then there must exist an optimal nearest neighbour quantizer. The optimal nearest neighbour quantizer for a uniform source is the uniform quantizer [16]. Thus, we can easily determine the optimal codepoints, given K. For $K = 2$, the optimal decision boundary occurs at $x = \frac{1}{2}$, halfway between the ends of the support. The optimal codepoints are then given as the centroids of each decision region. These are $y_1 = 0.25$ and $y_2 = 0.75$. Using the codepoints, we can now compute the optimal distortion. The distortion for a quantizer, q, is given by

$$\begin{aligned} D(\mu, q) &= E\left[(X - q(X))^2\right] \\ &= \sum_{i=1}^{K} P(X \in S_i) E\left[(X - q(X))^2 | X \in S_i\right] \\ &= \frac{\Delta^2}{12} \\ &= \frac{\left(\frac{1}{2}\right)^2}{12} \\ &= 0.0208\dot{3} \end{aligned} \tag{3.6}$$

By similar means, the optimal distortions for rates of 2 and 3 bits/sample were determined to be $\frac{\left(\frac{1}{4}\right)^2}{12} = 0.005208\dot{3}$ and $\frac{\left(\frac{1}{8}\right)^2}{12} = 0.00130208\dot{3}$, respectively.

As we have seen, the generation of uniform random variates is quite simple. The generation of non-uniform random variates, however, is somewhat more complicated. In order to obtain data sets drawn from the desired probability distributions, we must process the data produced by the pseudo-random number generator. There are several methods by which non-uniform random variates can be generated, with different methods being better suited to certain distributions. We will now describe the methods used in this work, and see how they are applied to obtain random variates from the desired distributions.

Gaussian Random Variates

The first non-uniform distribution we will consider is the Gaussian. In this work, we use the standard normal distribution, in lieu of a more general case. This distribution was selected because of its pervasiveness in the modeling and simulation of naturally occurring sources. The density function of the Gaussian distribution, denoted $f_g(x)$, is given by:

$$f_g(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{3.7}$$

where μ is the mean and σ^2 is the variance. In order to obtain the pdf for the standard normal distribution, we set $\mu = 0$ and $\sigma^2 = 1$, giving

$$f_g(x) = \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}}. \tag{3.8}$$

The technique used to generate Gaussian random variates in this work is based on the Box-Muller Method. This method was chosen because of its ease of implementation and high level of accuracy. The original Box-Muller Method consists of the following steps:

Step 1: We begin by generating two independent pseudo-random numbers from a uniform distribution on the interval (0,1). These numbers are generated using different streams in order to ensure their independence. Call them u_1 and u_2.

Step 2: Using u_1 and u_2, we then generate two independent Gaussian variates, with mean μ and standard deviation σ, as follows.

$$\begin{aligned} x_1 &= \mu + \sigma \cos(2\pi u_1)\sqrt{-2\ln(u_2)} \\ x_2 &= \mu + \sigma \sin(2\pi u_1)\sqrt{-2\ln(u_2)}. \end{aligned} \tag{3.9}$$

Thus, the Box-Muller Method allows us to generate one Gaussian data set for each stream of the pseudo-random number generator. This is a desirable result, as the fewer streams used per data set, the larger the number of independent sets that can be generated. Unfortunately, however, we cannot use this method exactly as it is outlined here. The reason is that, when the uniform variates, u_1 and u_2, are generated using an LCG, the resulting standard normal variates, x_1 and x_2, may be correlated [7]. This is unacceptable since we require all data sets to be independent of one another. To circumvent this potential problem, we simply change step two to the following.

Actual Step 2: Using u_1 and u_2, a single Gaussian variate with mean $\mu = 0$ and standard deviation $\sigma = 1$ is generated as follows:

$$x_1 = \cos(2\pi u_1)\sqrt{-2\ln(u_2)} \tag{3.10}$$

By not computing x_2, the possibility of correlation is avoided. However, to ensure the independence of the data sets, we pay a price. We now need two streams to generate each independent data set. This reduces the number of independent data sets that can be generated using our pseudo-random number generator by 50 %.

Laplacian Random Variates

The next non-uniform distribution we will consider is the Laplacian distribution. More specifically, we wish to consider a Laplacian distribution with parameter $a = 1$. As is the case for the Gaussian distribution, this distribution is also widely used for the modeling and simulation of naturally occurring sources. The pdf of the Laplacian distribution, denoted $f_L(x)$, is given by

$$f_L(x) = \frac{1}{2a} e^{-\frac{|x|}{a}}. \tag{3.11}$$

In this project, we chose to take $a = 1$, and so the pdf of our Laplacian distribution is

$$f_L(x) = \frac{1}{2} e^{-|x|}. \tag{3.12}$$

The technique used to generate Laplacian random variates in this work is called the composition method. This technique is useful when the sought distribution, $F(x)$, can be expressed as a weighted sum of some other distributions,

$$F(x) = \sum_{i=1}^{n} p_i F_i(x) \tag{3.13}$$

where p_i is non-negative for every i and the p_i sum to 1 [7]. This method can be used to efficiently generate random variates from a Laplacian distribution since this distribution is actually the composition of exponential distributions.

The following steps outline how composition is implemented to generate the Laplacian data sets:

Step 1: We begin by generating two independent pseudo-random numbers from a uniform distribution on the interval (0,1). These numbers are generated using different streams in order to ensure their independence. Call them u_1 and u_2.

Step 2: If $u_1 < 0.5$ then return $-\ln(u_2)$, else return $\ln(u_2)$.

By repeating these steps, data sets can be created.

Beta Random Variates

The fourth and final distribution we will consider is the beta distribution. This distribution is not as widely used as the other non-uniform distributions investigated in this work. It is, however, useful in the modeling of random proportions, such as the percentage of packets that require retransmission in a data network. It is also appropriate for random variables that vary between two finite limits [5]. It is investigated because it is log convex, whereas the other distributions investigated thus far are log concave. Also, it is the only non-uniform distribution with a bounded support investigated in this work. Since it is log convex, Theorem 9 does not apply to it. The pdf of the beta distribution, denoted $f_b(x)$, is given by

$$f_b(x) = \frac{x^{a-1}\,(1-x)^{b-1}}{\int_0^1 x^{a-1}\,(1-x)^{b-1}\,dx} \tag{3.14}$$

where the parameters a and b are both taken to be 0.5. Thus, this pdf can be simplified to

$$f_b(x) = \frac{x^{-0.5}\,(1-x)^{-0.5}}{\int_0^1 x^{-0.5}\,(1-x)^{-0.5}\,dx}\ . \tag{3.15}$$

The method used to generate beta random variates is based on the work of Johnk (1964) [7]. It consists of the following steps:

Step 1: We begin by generating two independent pseudo-random numbers from a uniform distribution on the interval (0,1). These numbers are generated using different streams in order to ensure their independence. Call them u_1 and u_2.

Step 2: The next step is to compute the following values: $x = \sqrt{u_1}$ and $y = \sqrt{u_2}$.

Step 3: If $x + y > 1$, then we return to step one and repeat the process. Otherwise, we return $\frac{x}{x+y}$ as a value drawn from a beta distribution.

Once each data set is obtained, it has to be formatted to match the required form for the input to the quantizer design algorithm. Our quantizer design algorithm imposes the restriction that the training sets must be sorted in ascending order. This constraint requires the implementation of a sorting algorithm, which will be the topic of the following subsection. There are no restrictions on the test sets.

3.1.3 Ordering Training Sets

Since each training set must be listed in ascending order, a sorting algorithm is required. The next step in the implementation process is to decide which type of sorting algorithm to use. The main requirement is that the sorting algorithm be able to perform the sort quickly. At the outset, the running time of the quantizer design algorithm was determined to be $\mathcal{O}(KN \log N)$, where K is the number of quantization levels and N is the cardinality of the training set. For a detailed investigation of the computational complexity of our quantizer design algorithm, see Chapter 2. Thus, anything substantially slower than $\mathcal{O}(KN \log N)$ could potentially cause a bottleneck in our simulations.

Some of the most common, and arguably simplest sorting algorithms are bubble, insertion, selection and shell sort [13]. These algorithms are very easy to implement, and are ideal for performing quick tests, provided the amount of data to be sorted is relatively small. Unfortunately, despite the fact that the algorithms listed above are fast enough for small data sets, they all have average and worst case asymptotic runtimes of $\mathcal{O}(N^2)$, which is substantially slower than $\mathcal{O}(KN\log N)$ for large N. Fortunately, faster sorting routines are available. Heap, merge and quick sort all run in $\mathcal{O}(N\log N)$ time in the average case, making them substantially faster than the simpler algorithms mentioned above.

Due to its relative pervasiveness, quicksort was selected for implementation from among these faster sorting algorithms. It is faster, on the average, than merge sort and heap sort, although in the worst case, it is slower than heap or merge sort. This worst case occurs when the data to be sorted is already listed in ascending order. For our purposes, this was not a concern, and so quicksort was the obvious choice.

3.1.4 Size of Data Sets

Another important issue to address is the size of the training and test sets. We will first discuss training set size. It is important to select a range of training set sizes that incorporates both sizes that are small enough to produce quantizers that are substantially sub-optimal, and sizes that are large enough to produce near-optimal quantizers. We decided on a minimum training set size of 50 points, and a maximum size of 50000 points. In all cases, the largest sets produced near-optimal quantizers, whereas the smaller sets produce quantizers with noticeably sub-optimal performance. Generating training sets from the low and high ends of this range is not sufficient, however. It is also very important to generate many sets of different sizes throughout this range.

Two issues arise when determining exactly which training set sizes to use. Firstly, we want to maximize the information that is available for analysis. In order to do this, we must have a higher concentration of training set sizes in the range where most of the change in training distortion is expected to occur. This occurs at the lower end of our overall range. The second issue is that of time. The design of optimal scalar quantizers from large training sets is a computationally intensive process. Thus, it is not practical to generate many different large training sets, where $N > 10000$. In order to accommodate both issues presented here, sets of sixteen different sizes were used, with fourteen of these sizes being less than or equal to 10000.

Unlike training sets, test sets of various sizes are not generated. The size of the test sets is still very important, though, as is the number of test sets. If the test sets are too small, then they will not provide an adequate representation of the source, and so our results will not be accurate. If the test sets are too large, we will have wasted time, memory, and streams by using them. In this work, we use test sets of size 100 000. This choice is dually motivated. Firstly, for the distributions being considered, experience shows that 100 000 randomly generated samples are a sufficient number to guarantee that the test sets are representative of the source. Secondly, each stream of our random number generator is capable of producing 100 000 independent pseudo-random numbers. Thus, by choosing this size, we ensure that no more than one stream is used for each test set. This leaves sufficient streams available for the generation of both training sets and additional test sets. It should be noted that the method by which beta random variates are produced necessitates discarding approximately twenty percent of all data produced by the LCG. Thus, two streams are required to generate each beta test set.

In order to obtain more reliable test distortion values, we use ten independent test sets. We then average the results for all ten sets. This allows us to reduce the effects of any anomalies in the distribution of isolated test sets.

3.1.5 Nesting of Training Sets

Another important decision, from an experimental design point of view, is how exactly the training sets should be generated. As discussed in the previous section, training sets of various sizes are required in order to design quantizers with varying performance. To keep the number of streams in use to a minimum, we generate nested training sets. Nesting is accomplished by appending additional points to small training sets in order to create larger ones. Using this technique, we are able to generate an entire group of training sets (one of each size) using a single stream.

3.1.6 Number of Training Sets

The number of training sets used is a very important factor to consider. If the number of training sets is too small, then there may be a high level of variability among the resulting codebooks and training distortions. If too many training sets are used, then the marginal change in accuracy will be negligible. This means that very little new information is obtained. Since a limited amount of training data is available, it is prudent to use only what is necessary. Also, the quantizer design process is complex and time-consuming, so it is desirable to minimize the number of quantizers designed.

We decided to generate as many training sets as was practically possible for small values of N. For larger values of N, fewer sets were required. The following table shows the number of training sets of each size generated for each distribution.

	Number of Training Sets			
N	Uniform	Gaussian	Laplacian	Beta
50	75	40	40	30
100	75	40	40	30
250	75	40	40	30
500	50	40	40	30
1000	50	40	40	30
2000	50	40	40	30
3000	50	40	40	30
4000	50	40	40	30
5000	25	25	25	25
6000	25	25	25	25
7000	25	25	25	25
8000	25	25	25	25
9000	25	25	25	25
10000	25	25	25	25
20000	25	25	25	25
50000	15	15	15	15

Table 3.1: Number of training sets.

3.2 Number of Quantization Levels

Another important implementation decision to be considered is the number of quantization levels. This problem is equally well described as a rate problem, since the number of levels and the rate are related via

$$R = \log_2 K \tag{3.16}$$

where R is the rate of the quantizer and K is the number of levels. Thus, there is a 1-1 correspondence between rate and the number of quantization levels.

We chose to run our simulations at rates of 1, 2, and 3 bits/sample. The reason for this is twofold. Firstly, these values are commonly encountered in simulations. Secondly, as the rate increases, the time required to design empirically optimal quantizers increases dramatically.

3.3 Analysis of Results

The goal of this work is to determine the rate at which the distortion of an empirically optimal quantizer converges to the distortion of the optimal quantizer for the source as a function of training set size. The empirical data accrued consists of training and test distortion values for ascending values of N. It is important to interpret this data in an intelligent and meaningful way. The most appropriate way to interpret the data, given our end goal, is to fit curves to the distortion values.

The basic goal of curve fitting, known more formally as data modeling, is to find the function or model that best approximates a set of data points. There are several different types of data modeling, the most common of which are interpolation and regression. Interpolation requires that every data point lie on the curve. Since empirically obtained data is subject to experimental error, an exact fit is usually not possible, and so interpolation is not appropriate here. Regression, unlike interpolation, fits a curve to the data set by minimizing the distance from the data set to the curve, for some distance measure. This method is the most appropriate for our purposes because we wish to find a best-fit curve that does not necessarily pass through all of the data points.

There are two broad categories of regression: linear regression and non-linear regression. It was apparent from the results we obtained that the sought relationship was not linear, and so non-linear regression was selected.

Several methods exist for performing non-linear regression. In this project, we use the Levenberg-Marquardt method. This method finds the coefficients in the data model by minimizing the distance from the data set to the fitted curve. It begins by using a steepest descent approach. This works well so long as we are still far away from the minimum. It then uses a Taylor series approach once the minimum is close. This method lends itself well to regression problems in which the coefficients in the model are unconstrained, as is the case with our model [12].

The first step in any curve fitting procedure, once the appropriate routine has been selected, is to specify an assumed form for the curve. Examples of curve forms include polynomial, power, and exponential. In this work, we seek a relationship of the form αN^{β}, where $\alpha,\ \beta \in \mathbb{R}$ and $\alpha > 0$. That is, we are interested in fitting power curves to our data points. Once an expression for the desired form has been specified, the curve fitting routine is then used to determine any unknown parameters. In this work, we are interested in curves of the form

$$y = \alpha N^{\beta} \tag{3.17}$$

so the routine returns values for α and β.

The results obtained using the technique described above are given in Chapter 4.

3.3.1 Programming Language

One of the first implementation decisions to make is that of programming language selection. In this work, we use the C programming language for several reasons. Firstly, being a simple procedural language, C is able to perform calculations much more quickly than are object-oriented languages, such as Java. Secondly, since C is a lower level language than Java, it gave us more control over resources such as the allocation of memory. This is important because we are dealing with very large arrays of numbers, and we would like to be able to allocated exactly the amount of memory they require and no more. Thus, we opted to use C, as it is well-suited for the needs of this project.

Chapter 4

Results and Discussion

As stated in Chapter 2, lower bounds on the difference between both test and training distortions and the optimal distortion for the source show that a convergence rate proportional to $\frac{1}{\sqrt{N}}$ is the best we can guarantee. However, these bounds are minimax in nature, and so do not apply to specific distributions. Our goal in this project is to determine whether these bounds can be tightened for certain important source distributions.

In order to accomplish this goal, we investigate the performance of empirically optimal quantizers. In the current chapter, we present the results of our simulations. We compare the test and training distortions of empirically optimal quantizers to the optimal distortion for the source for three different rates and four different source distributions.

4.1 Simulation Environment

For our simulations, training sets of various sizes, ranging from 50 to 50000 samples were used to design empirically optimal quantizers. Each quantizer was then used to quantize ten independent test sets, each of size 100 000. All data sets were generated using a prime modulus MLCG pseudo-random number generator and appropriate processing. For each

	Source Distribution		
Rate (bits/sample)	Gaussian	Laplacian	Beta
1	0.3632	1.000	0.023695
2	0.1175	0.3522	0.005225
3	0.0346	0.109	0.001233

Table 4.1: Approximate optimal distortion values for non-uniform sources.

distribution, the performance for rates 1, 2, and 3 bits/sample was investigated. For all figures shown in this chapter, training set size, N, is taken to be the independent variable, while mean squared error distortion, MSE, is taken to be the dependent variable.

4.2 Simulation Results

We now present and interpret our simulation results. As discussed above, we ran simulations using four different source distributions. We also considered three different rates for each source distribution, namely 1, 2, and 3 bits/sample.

It is important to note that, for each source distribution, a different optimal distortion value exists at each rate. As shown in Chapter 3, when the source is uniformly distributed, we can calculate this optimal value. Unfortunately, when the source, μ, has a non-uniform distribution, a closed form expression for $D_K^*(\mu)$ does not, in general, exist. Thus, approximate values for the optimal distortion of quantizers on each of the non-uniform sources considered must be obtained. These values, which can be determined by approximating the MSE value at which the test and training distortion curves meet for each distribution and each rate, are listed in Table 4.1.

The results of our simulations are presented in Figures 4.1 to 4.12. We will now examine these results.

	Source Distribution			
Rate (bits/sample)	Uniform	Gaussian	Laplacian	Beta
1	90	100	250	90
2	400	550	800	600
3	800	1500	2000	800

Table 4.2: Minimum training set size for test distortion to be within 3% of optimal distortion for the true source.

Firstly, as the rate increases, the optimal distortion value for a given source distribution does indeed decrease, as expected. This is expected since, intuitively, as the number of representative points increases, the average distortion should decrease. Also, for increasing rates, both the test and training distortion curves become smoother. We expect the test distortion to be monotonically non-increasing and the training distortion to be monotonically non-decreasing. In the former case, our results confirm our expectations. However, in the latter case, our results deviate from what is expected. The erratic behaviour of the training distortion curve for a rate of 1 bit/sample is attributed primarily to the LCG used in our simulations. Another possible cause is that the number of training sets used was insufficient. We are uncertain as to the true cause of this erratic behaviour. Due to this erratic behaviour, we feel that, while our results are promising, they are also somewhat inconclusive.

Another observation common to all distributions considered in this work is that, for smaller rates, the test distortion converges to the optimal value more quickly than it does for higher rates. Table 4.2 lists the number of samples required before the test distortion approaches to within 3% of the optimal value.

From Table 4.2, we can see that the Gaussian and Laplacian distributions seem to converge slower than do the uniform and beta distributions. One possible reason that

Rate (bits/sample)	α	β
1	0.01673	-0.7436 (-0.796, -0.692)
2	0.02687	-0.9368 (-1.027, -0.847)
3	0.00911	-0.8583 (-0.892, -0.825)

Table 4.3: Test distortion results for uniform source.

Rate (bits/sample)	α	β
1	0.06671	-1.03 (-1.319, -0.7421)
2	0.02373	-0.8895 (-0.9688, -0.8102)
3	0.00972	-0.7896 (-0.8181, -0.7611)

Table 4.4: Training distortion results for uniform source.

the Gaussian and Laplacian distributions are slower to converge is that they have the largest non-negligible supports. By this, we mean the portion of the support that is non-negligible for simulation purposes.

Our most important conclusions can only be drawn from the data once it has been properly interpreted. As stated in Chapter 3, we used curve fitting to fit power curves to $E[D(\mu, q_N^*)] - D_K^*(\mu)$ and $D_K^*(\mu) - E[D(\mu_N, q_N^*)]$. The curve fitting algorithm fit a curve of the form $y = \alpha N^{\beta}$ to our data. The values of α and β that were returned are shown in Tables 4.3 to 4.10.

The results presented in Tables 4.3 to 4.10 are given at a confidence level of 95%. The ranges shown in the β columns are the 95% confidence intervals for the true values of β.

We will now consider the test distortions, and more specifically, the exponent, β. Recall, the minimax lower bound in Theorem 7 states that

Rate (bits/sample)	α	β
1	0.7563	-0.8967 (-0.9423, -0.851)
2	0.6216	-0.8529 (-0.8948, -0.8111)
3	0.2713	-0.801 (-0.8502, -0.7519)

Table 4.5: Test distortion results for standard normal source.

Rate (bits/sample)	α	β
1	0.2353	-0.7303 (-1.293, -0.1676)
2	0.42	-0.7606 (-0.8948, -0.6264)
3	0.4149	-0.8107 (-0.8926, -0.7289)

Table 4.6: Training distortion results for standard normal source.

Rate (bits/sample)	α	β
1	1.195	-0.6979 (-0.7398, -0.656)
2	1.041	-0.6695 (-0.7472, -0.5919)
3	1.063	-0.7424 (-0.797, -0.6877)

Table 4.7: Test distortion results for Laplacian source.

Rate (bits/sample)	α	β
1	22.11	-1.33 (-2.204, -0.4556)
2	4.115	-0.8884 (-1.028, -0.7483)
3	0.8963	-0.6381 (-0.7047, -0.5716)

Table 4.8: Training distortion results for Laplacian source.

Rate (bits/sample)	α	β
1	0.1122	-1.091 (-1.28, -0.9028)
2	0.03048	-0.9144 (-1.059, -0.7697)
3	0.007696	-0.8011 (-0.8372, -0.765)

Table 4.9: Test distortion results for beta source.

Rate (bits/sample)	α	β
1	0.09582	-0.9884 (-1.797, -0.1801)
2	0.04307	-0.9736 (-1.118, -0.8293)
3	0.01133	-0.8108 (-0.8623, -0.7593)

Table 4.10: Training distortion results for beta source.

$$\sup_{\mu \in \mathcal{M}(M)} E\left[D\left(\mu, q_N\right)\right] - D_K^*(\mu) \geq \frac{C_2}{\sqrt{N}} = C_2 N^{-0.5}.$$

From Tables 4.3, 4.5, 4.7, and 4.9, we can see that for all three rates and all four source distributions, an exponent less than -0.5 was obtained. These exponents lead us to suspect that the actual convergence rate of the test distortion of an empirically optimal quantizer to the optimal distortion for the sources considered here is substantially faster than that which is suggested in Theorem 7. Thus, we suspect that the actual rate of convergence of the test distortion to the optimal distortion for all sources considered here is faster than $\frac{1}{\sqrt{N}}$.

We will now consider the training distortion results. In Theorem 8, we saw that the following minimax lower bound holds when the distribution is not known

$$\sup_{\mu \in \mathcal{M}(M)} D_K^*(\mu) - E\left[D\left(\mu_N, q_N^*\right)\right] \geq \frac{C_3}{\sqrt{N}} = C_3 N^{-0.5}.$$

The exponents in Tables 4.4, 4.6, 4.8, and 4.10 for rates 1, 2, and 3 bits/sample are substantially lower than the value of -0.5 from Theorem 8. Thus, we suspect that the actual rate of convergence of the training distortion to the optimal distortion for all sources considered here is faster than $\frac{1}{\sqrt{N}}$.

It should be noted that, due to the erratic behaviour of the training distortion curves, the entries for a rate of 1 bit/sample in Tables 4.6, 4.8, and 4.10 are not very reliable

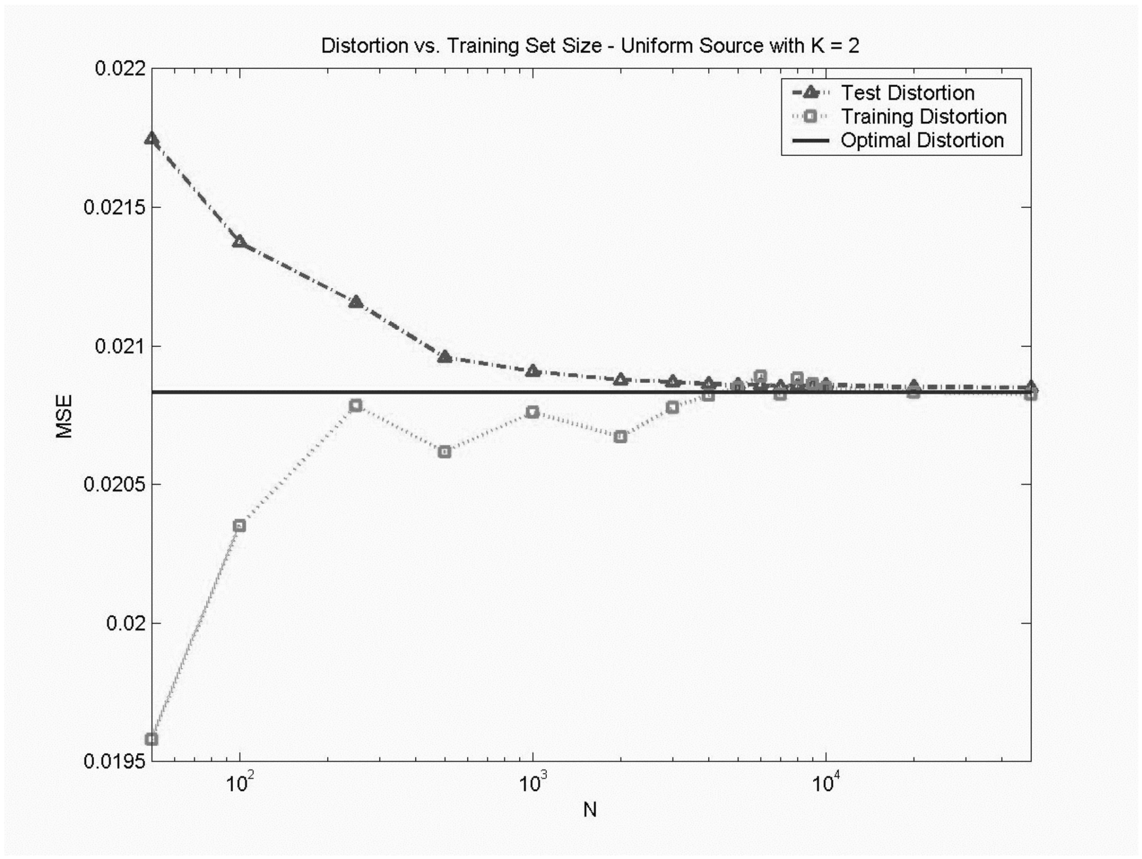

Figure 4.1: Effect of training set size on test and training distortion; uniform source; rate = 1 bit/sample.

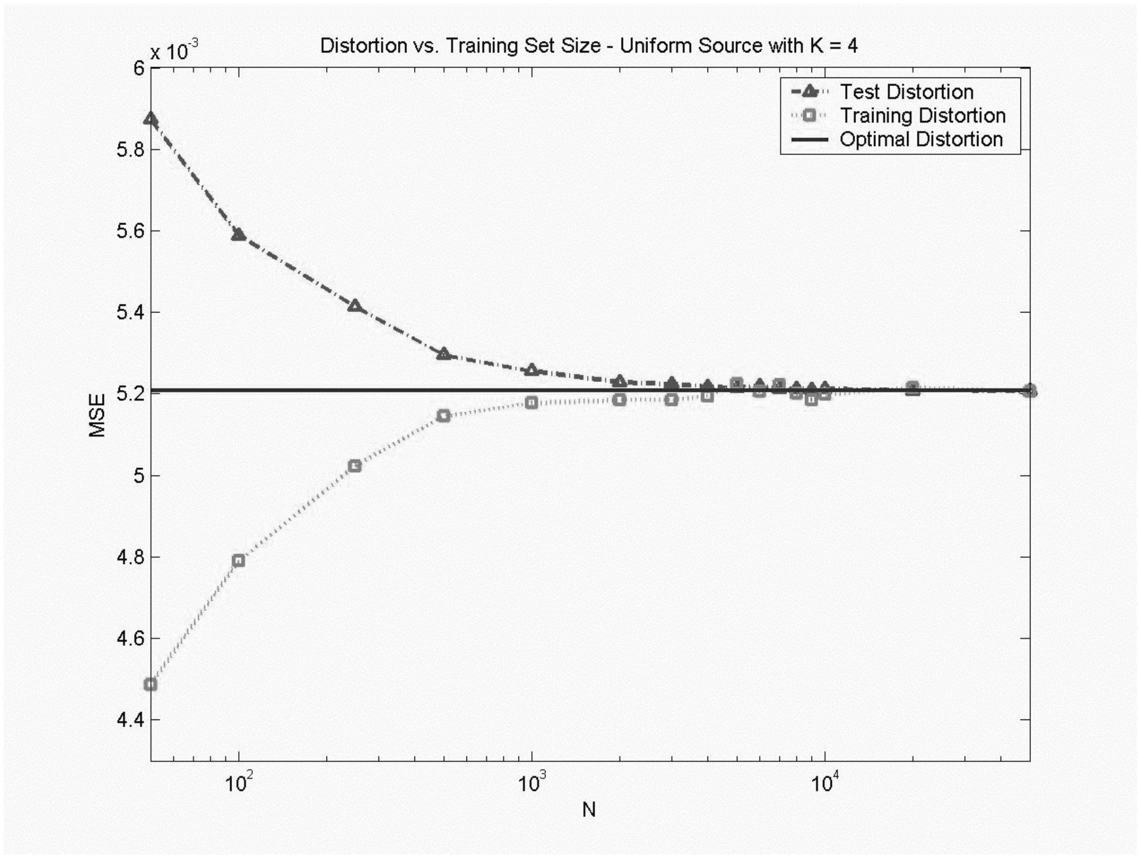

Figure 4.2: Effect of training set size on test and training distortion; uniform source; rate = 2 bits/sample.

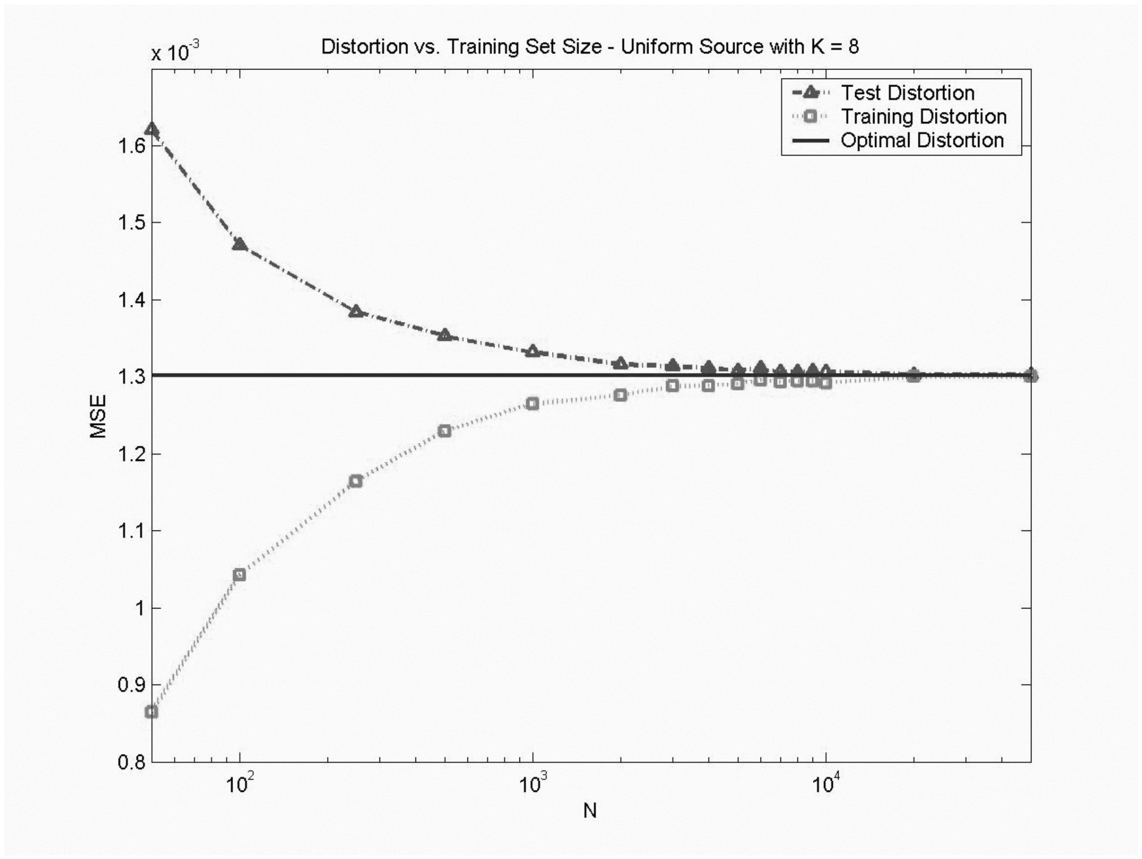

Figure 4.3: Effect of training set size on test and training distortion; uniform source; rate = 3 bits/sample.

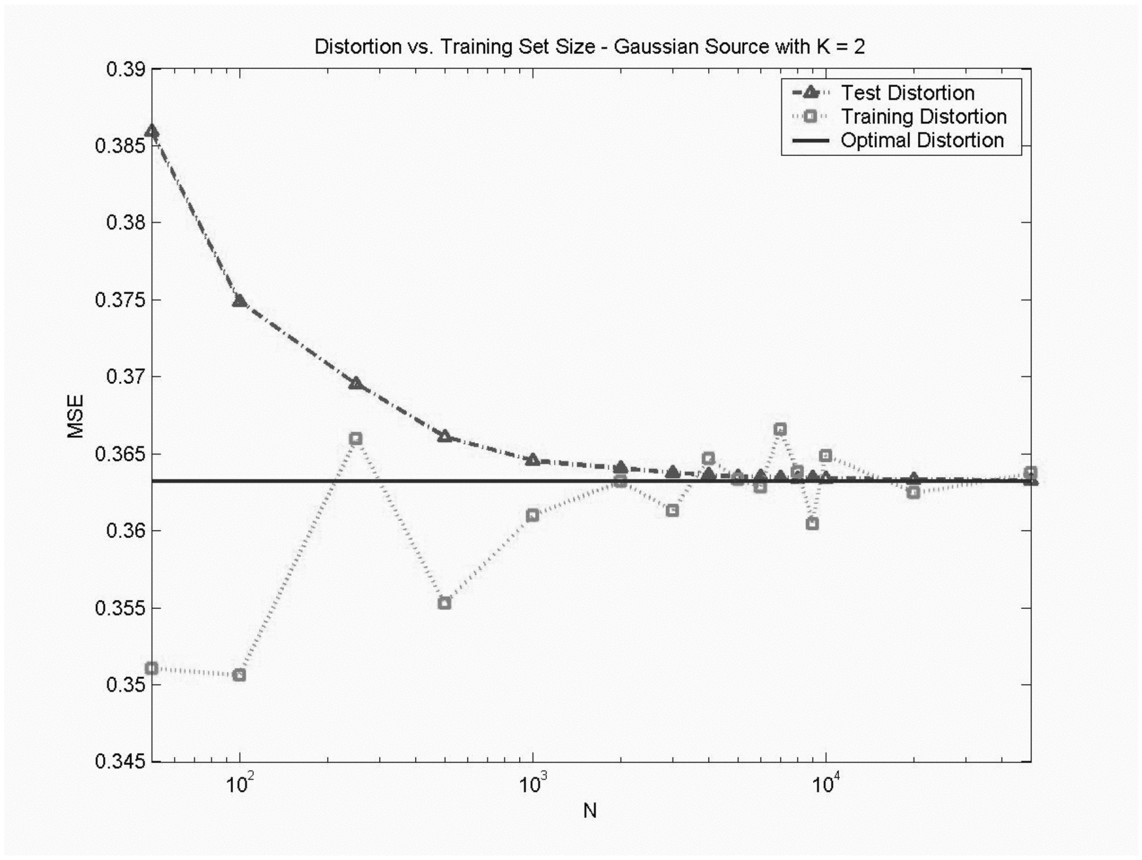

Figure 4.4: Effect of training set size on test and training distortion; Gaussian source; $\mu = 0$, $\sigma^2 = 1$; rate = 1 bit/sample.

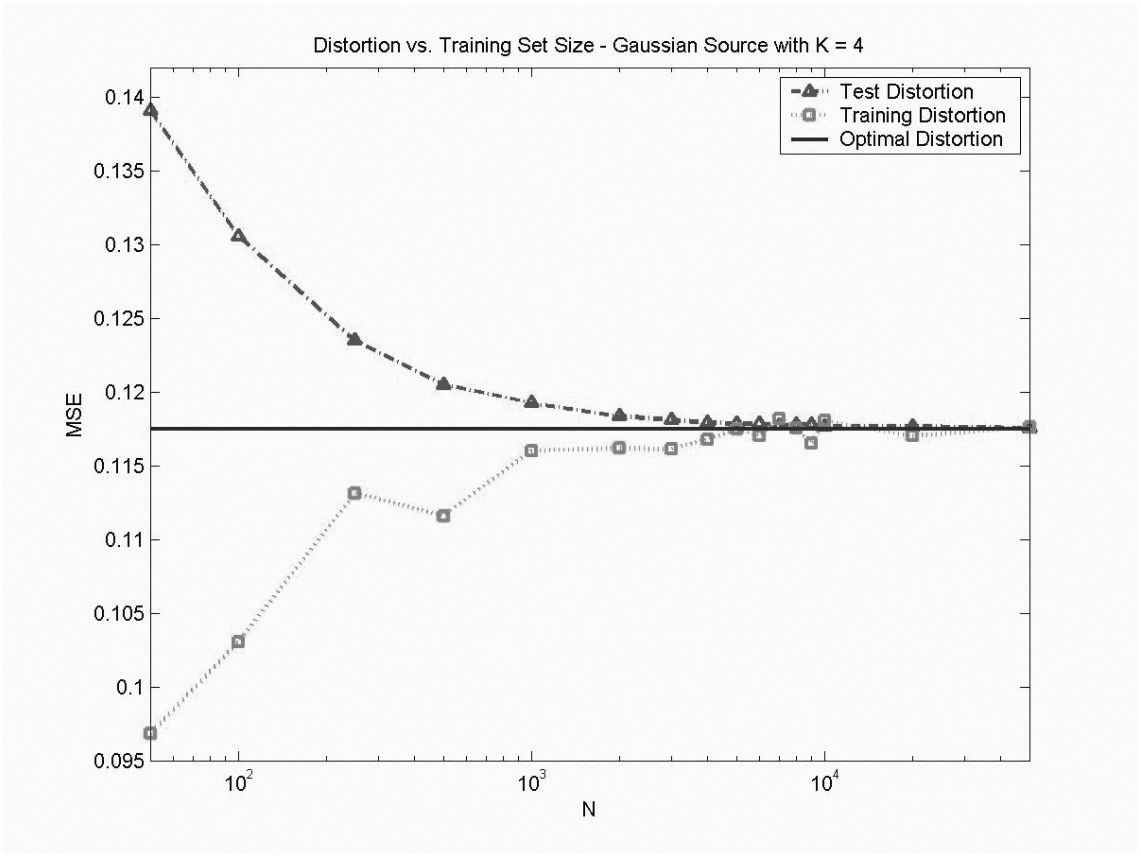

Figure 4.5: Effect of training set size on test and training distortion; Gaussian source; $\mu = 0$, $\sigma^2 = 1$; rate = 2 bits/sample.

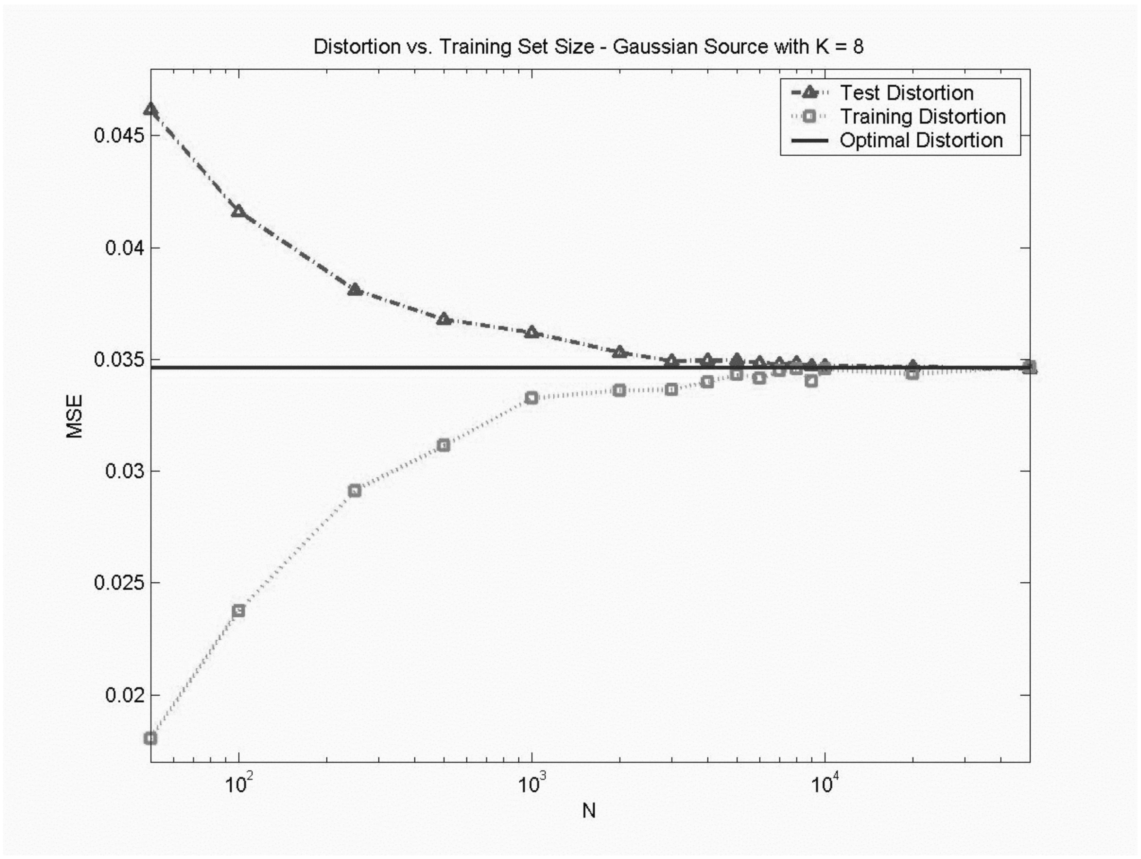

Figure 4.6: Effect of training set size on test and training distortion; Gaussian source; $\mu = 0$, $\sigma^2 = 1$; rate = 3 bits/sample.

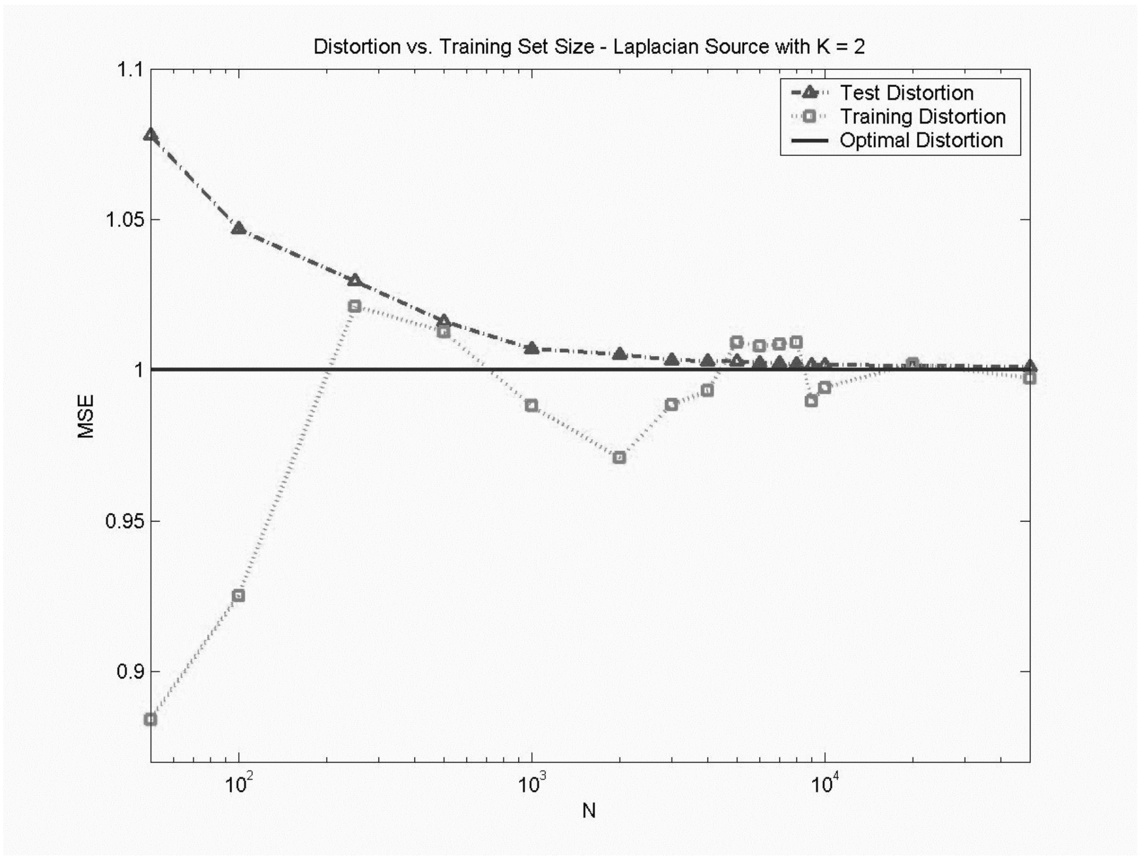

Figure 4.7: Effect of training set size on test and training distortion; Laplacian source; $a = 1$; rate = 1 bit/sample.

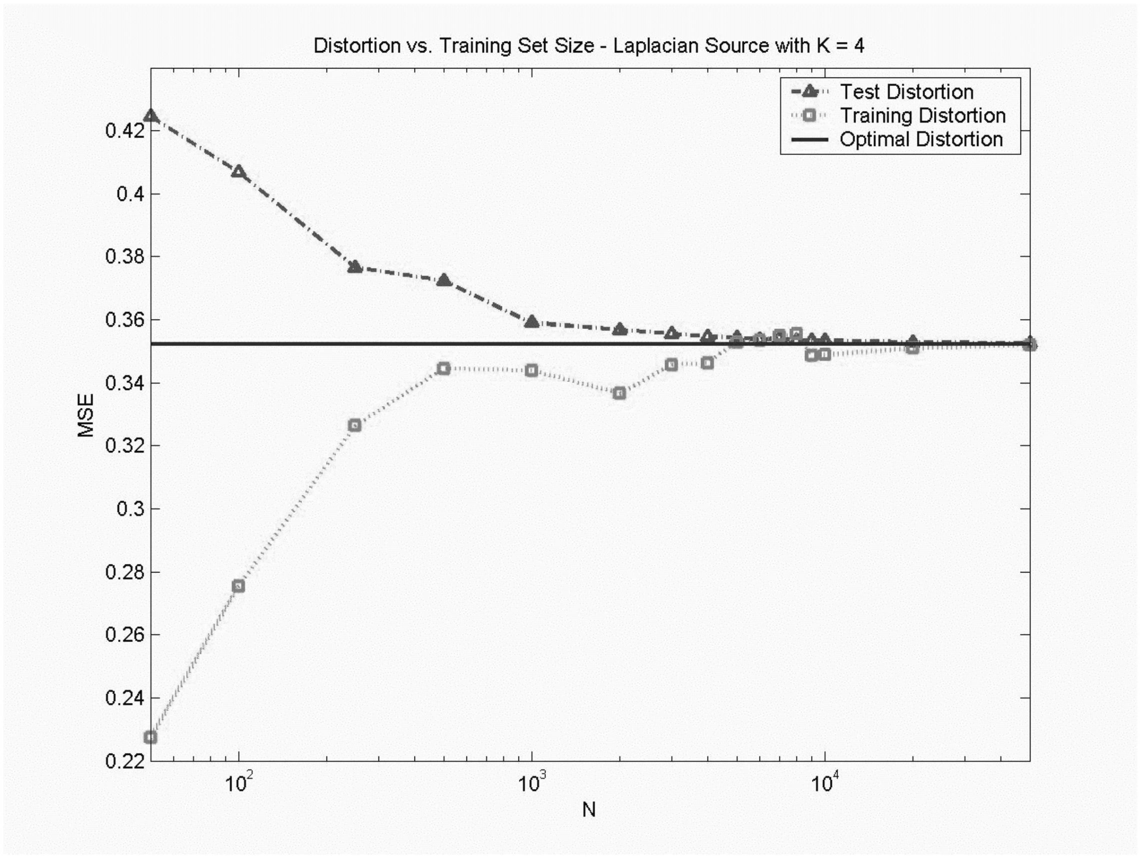

Figure 4.8: Effect of training set size on test and training distortion; Laplacian source; $a = 1$; rate $= 2$ bits/sample.

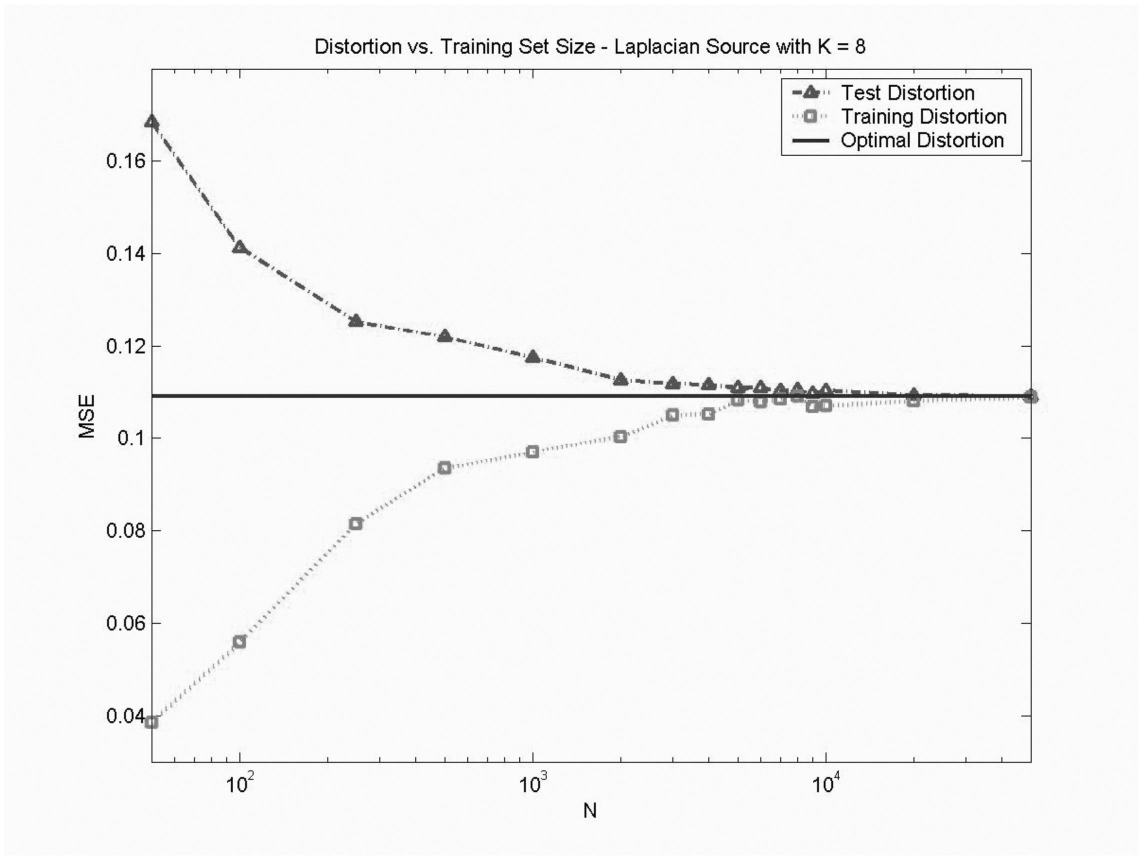

Figure 4.9: Effect of training set size on test and training distortion; Laplacian source; $a = 1$; rate = 3 bits/sample.

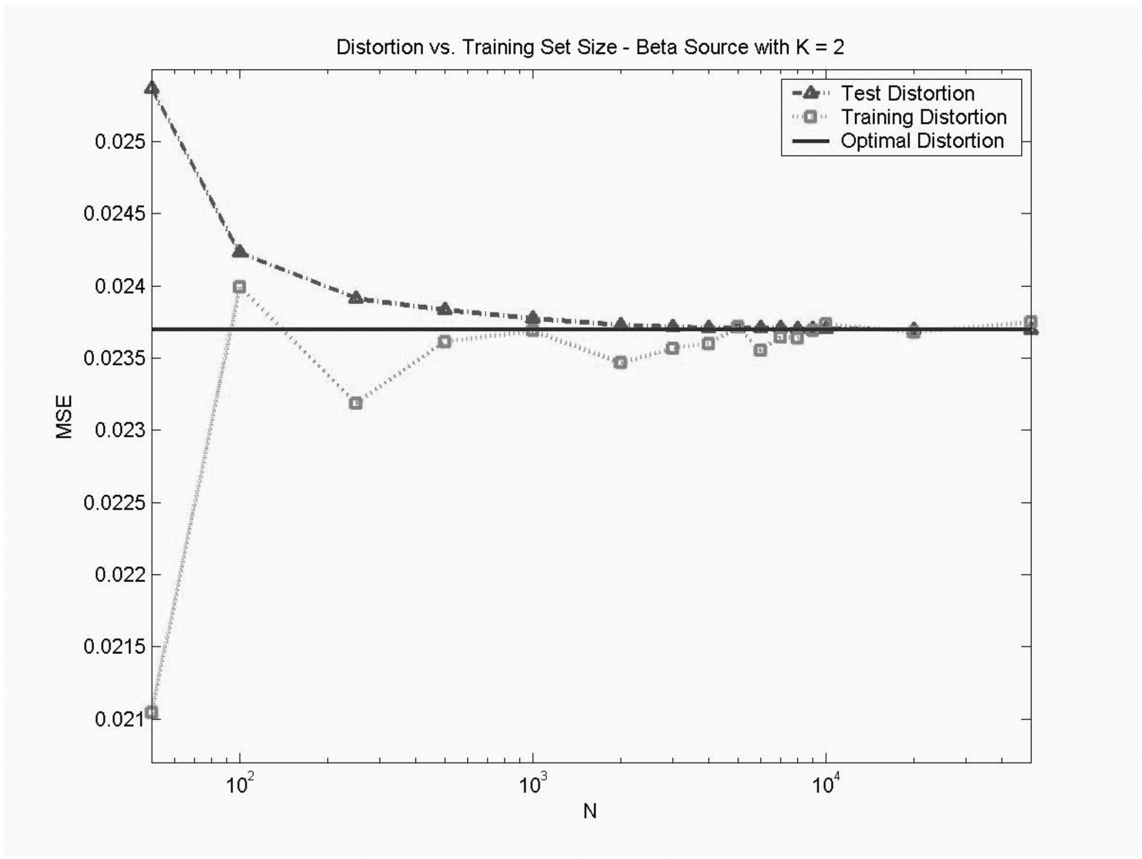

Figure 4.10: Effect of training set size on test and training distortion; beta source; $a = b = \frac{1}{2}$; rate = 1 bit/sample.

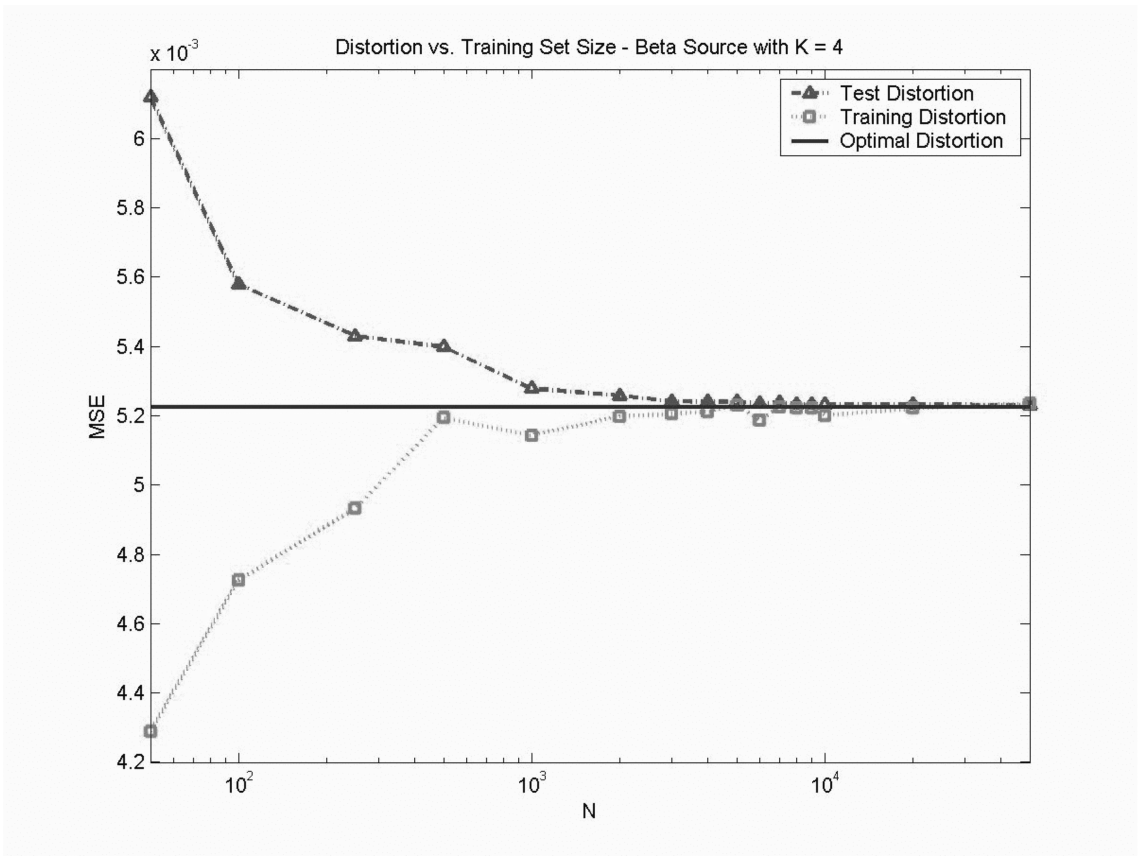

Figure 4.11: Effect of training set size on test and training distortion; beta source; $a = b = \frac{1}{2}$; rate = 2 bits/sample.

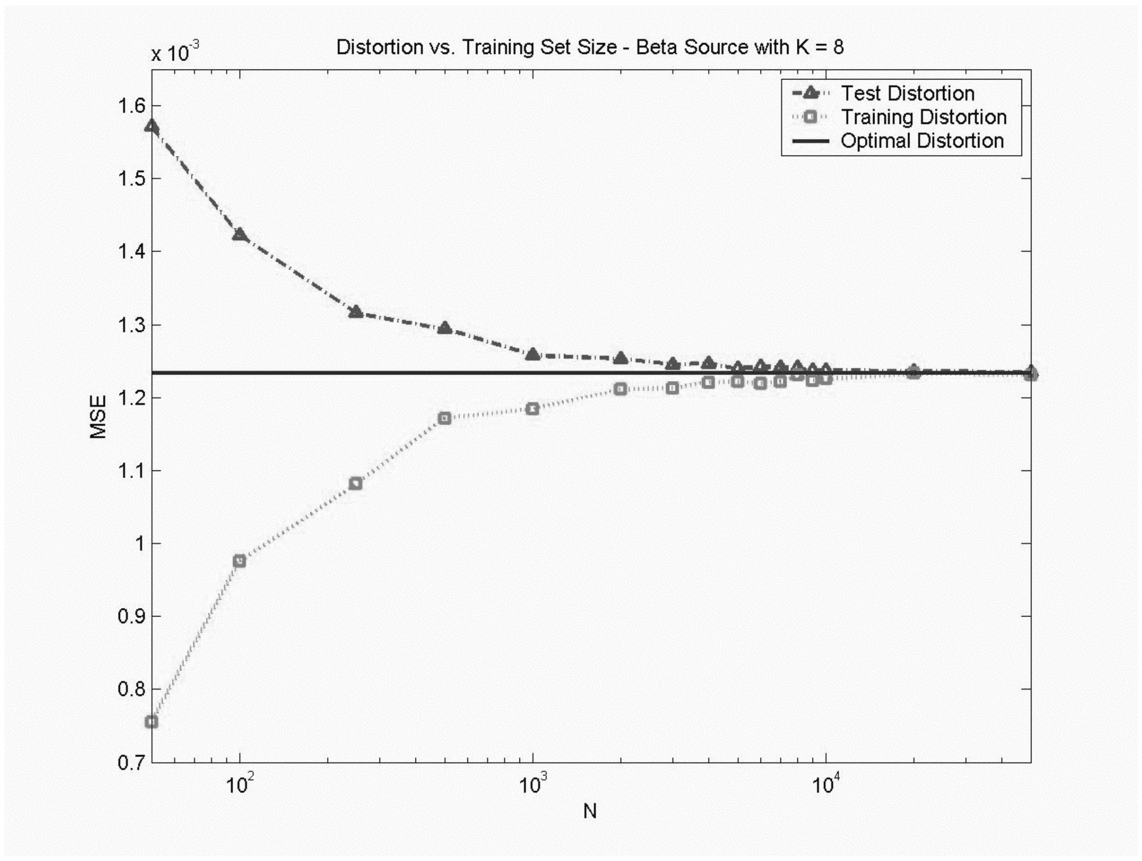

Figure 4.12: Effect of training set size on test and training distortion; beta source; $a = b = \frac{1}{2}$; rate = 3 bits/sample.

Chapter 5

Conclusions

5.1 Concluding Remarks

In this project, we sought an answer to the following problem: What is the true rate of convergence of the test and training distortions of empirically optimal quantizers for certain sources to that of the optimal quantizer for the source as a function of training set size?

In Chapter 2, we described the topic of quantization and presented upper and lower bounds on the performance of empirically optimal quantizers. The motivation for this project can be brought to light by observing that the lower bounds are given in a minimax sense. They imply that bad distributions exist for which the convergence rate is slow, but say nothing of the performance of empirically designed quantizers for specific source distributions.

In Chapter 3, we considered relevant implementation issues such as the method of data set generation and the choice of curve fitting algorithm.

Upon completion of our discussions regarding design and implementation, our results were presented in Chapter 4. The effect of training set size on test and training distortion was examined for four different source distributions. For each source distribution, rates of 1, 2, and 3 bits/sample were considered. As expected, increasing the rate decreased the optimal distortion.

As the rate of the quantizer increased, so did the rate at which the aforementioned distortion values converged to the optimal value for the source. In general, we found that the source distributions with unbounded supports, namely Gaussian and Laplacian, exhibited slower convergence rates than did those with bounded supports. It is important to note that the artificial versions of these distributions do, in fact, have bounded supports since computers use finite precision values. This implies that Theorem 9 applies to these distributions. Interestingly, for rates of 2 and 3 bits/sample, the results for beta and uniform sources were quite similar.

For each source distribution and each rate, the quantities $E[D(\mu, q_N^*)] - D_K^*(\mu)$ and $D_K^*(\mu) - E[D(\mu_N, q_N^*)]$ exhibited a power function dependence on the training set size of the form αN^β.

For the first quantity, namely $E[D(\mu, q_N^*)] - D_K^*(\mu)$, we observed values for β ranging from -1.091 to -0.6695. Since these values are less than -0.5, we have reason to believe that the convergence rate proportional to $\frac{1}{\sqrt{N}}$ given in Theorem 7 can be improved on for all four distributions considered.

For the other quantity of interest, namely $D_K^*(\mu) - E[D(\mu_N, q_N^*)]$, we observed values for the exponent β ranging from -1.33 to -0.6381. Again, these values are less than -0.5, and so we suspect that the lower bound claimed in Theorem 8 can be improved on for all four distributions considered.

5.2 Future Work

As shown in Chapter 2, existing performance bounds on empirically optimal quantizers are weak. Furthermore, reasonable lower bounds have been found only in a minimax sense, and so do not apply to specific distributions. As demonstrated through our work, for several commonly used distributions, the lower bounds presented in Chapter 2 can potentially be improved. It is thus our opinion that this is an area that merits further research. Tightening the lower bounds in Chapter 2 from either a theoretical or an empirical standpoint is an important step towards a more comprehensive understanding of the empirical design process. This would allow more educated decisions regarding training set size to be made in many practical settings, resulting in more accurate results and/or savings in terms of both cost and time.

The results obtained via our simulations are potentially unreliable for a rate of 1 bit/sample. We feel that this lack of reliability resulted from the type of pseudo-random number generator used. Thus, it would be of interest to repeat our experiments using a more appropriate generator. It should be noted that selecting the appropriate random number generator is not a trivial task, as there are many subtleties associated with each implementation.

It would also be of interest to apply the Lloyd-Max algorithm to our training sets and compare the resulting distortion values with the results obtained in our work.

Finally, we would like to build on our results, and possibly investigate the behaviour of empirically optimal quantizers for additional source distributions and rates.

Bibliography

[1] A. Aggarwall, Maria M. Klawe, Shlomo Moran, Peter Shor, and Robert Wilber, "Geometric Applications of a Matrix-Searching Algorithm," *Algorithmica*, pp. 195-208, 1987.

[2] P. C. Cosman, K. O. Perlmutter, S. M. Perlmutter, R. A. Olshen, R. M. Gray, "Training Sequence Size and Vector Quantizer Performance,"

[3] P. E. Fleischer, "Sufficient Conditions for Achieving Minimum Distortion in a Quantizer," *Bell Telephone Laboratories, Inc.*, 1964.

[4] Allen Gersho and Robert M. Gray, "Vector Quantization and Signal Compression," *Kluwer Academic Publishers*, 1992.

[5] S. Ghahramani, "Fundamentals of Probability," *Prentice Hall*, 2000.

[6] A. Antos, L. Györfi, A. György, "Improved Convergence Rates in Empirical Vector Quantizer Design," *Submitted to: IEEE Trans. on Inform. Theory*, September 2003.

[7] R. Jain, "The Art of Computer Systems Performance Analysis," *John Wiley & Sons, Inc.*, 1991.

[8] A. M. Law and W. D. Kelton, Website for: "Simulation Modeling and Analysis," Online at: *http://www.mhhe.com/engcs/industrial/lawkelton/*, 2001.

[9] Tamás Linder, Gabor Lugosi, Kenneth Zeger, "Rates of Convergence in the Source Coding Theorem, in Empirical Quantizer Design, and in Universal Lossy Source Coding," *IEEE Trans. Inform. Theory*, Vol. 40, No. 6, pp. 1728-1740, November 1994.

[10] Tamás Linder, "Learning Theoretic Methods in Vector Quantization," *Lecture Notes for the Advanced School on the Principles of Nonparametric Learning*, 2001.

[11] P. Bartlett, T. Linder, G. Lugosi, "The Minimax Distortion Redundancy in Empirical Quantizer Design," *IEEE Trans. on Inform. Theory*, Vol. 44, No. 5, pp. 1802-1813, September 1998.

[12] MATLAB 6.5 Help, *The Mathworks, Inc.*, 2002.

[13] Kurt Melhorn, "Data Structures and Algorithms I: Sorting and Searching", *Springer-Verlag*, 1984.

[14] Daniel Nagy, "Fast Scalar Quantization, Step by Step", 2003.

[15] D. Pollard, "Quantization and the Method of k-means", *IEEE Trans. Inform. Theory*, IT-28:199-205, Mar. 1982.

[16] K. Sayood, "Introduction to Data Compression," Morgan-Kaufmann, 1996.

[17] Shant Sethian, Jay Shah, Cynthia Thomas, "Channel-Optimized Vector Quantization," Undergraduate Thesis, Queen's University, 2002.

[18] Dhiraj K. Sharma, "Design of Absolutely Optimal Quantizers for a Wide Class of Distortion Measures," *IEEE Trans. Inform. Theory*, Vol. IT-24, No. 6, pp. 693-702, November 1978.

[19] Xiaolin Wu, "Optimal Quantization by Matrix Searching," *Journal of Algorithms*, Vol. 12, pp.663-673, 1991.

[20] Xiaolin Wu, Kaizhong Zhang, "Quantizer Monotonicities and Globally Optimal Scalar Quantizer Design," *IEEE Trans. Inform. Theory*, Vol. 39, No. 3, pp. 1049-1053, May 1993.

Appendix