First Draft: March 1994
Current Version: April 4, 1995

# Forecasting Foreign Exchange Market Movements Via Entropy Coding

**Arman Glodjo**
*Duke University, Durham, NC 27708, USA*

*and*

**Campbell R. Harvey**
*Duke University, Durham, NC 27708, USA*
*National Bureau of Economic Research, Cambridge, MA 02138, USA*

This paper bridges the important recent work in computer science on information theory and data quantization to the forecasting of high frequency financial data. The technique of vector quantization has found its primary application in data compression algorithms. We argue that this technique is ideally suited for forecasting. Indeed, our paper shows that popular forecasting techniques such as, neural nets, are sub-classes of the more general vector quantization. Importantly, the neural nets are not just subclasses but are surely suboptimal. The vector quantization provides much more flexibility and a framework for efficient algorithmic approximation. In addition, vector quantization provides a way to incorporate conditioning information into the forecasting exercises. While much of our proposal details the theoretical motivation for entropy (amount of information) based coding, our empirical work is designed to implement forecasting models on intraday exchange rate data.

Harvey's research was supported by the Batterymarch Fellowship. We appreciate the comments of Frank Diebold and seminar participants at the First International Conference on High Frequency Data in Finance in Zurich. E-mail: arman@cs.duke.edu and charvey@mail.duke.edu. We appreciate the programming assistance from Greg J. Badros.

## 1 Introduction

One of the more striking advances in information theory has been the advent of provably optimal data compression techniques for rapid encoding/decoding and transmission of information [‡]. In this paper we take advantage of the recent progress in data compression and information-theoretic algorithms developed in the field of predictive computing, such as pre-fetching (fetching of data from slow memory to fast memory before the data is referenced by a program [26, 7]), to build models which forecast high-frequency financial data. The algorithms can be thought of as entropy-based encoding [27, 4, 5], where the encoding is a function of the entropy of the source (Entropy can be thought of a the measure of uncertainty of a random event), which may or may not be conditioned on side information. This paper not only presents rigorous derivations of information theoretic concepts, but also analyzes comprehensive entropy based trading strategies implemented on one year of spot transactions data on the U.S. dollar-Japanese yen, U.S. dollar-Deutschemark and Deutschemark-Japanese yen exchange rates, along with the countries respective interest rates, all provided by Olsen and Associates.

In section 2, we introduce the necessary information-theoretic notions that in turn will provide the foundation for provably optimal encoding techniques. We introduce the notion of a probability space and derive the analytic form for the amount of information associated with an event, followed by a derivation of the entropy measure of a random source. We give several examples of entropy for particular probability densities, followed by a brief discussion of the relationship between the likelihood function and corresponding entropy notions. We thereby provide a link between the two schools of thought and proceed to give a computationally tractable method of encoding a random source.

In section 3, we introduce the concept of vector quantization(VQ) and prove that any multidimensional data clustering and/or pattern recognition technique is properly contained in the set of all VQ algorithms over a Hilbert space. In particular, any nonparametric forecasting and neural-net based forecasting is some (and often not the most optimal) VQ technique. In section 4, we present our results on forecasting foreign exchange returns given no side information using scalar quantization and entropy encoding algorithms. Our aim is to develop reliable trading algorithms whose running time will allow us to make high-frequency trading decisions. Due to the potential high dimensionality of the data set, we rely on vector quantization as a means of compressing the data and thereby overcoming this constraint. The final section details our ideas for future research and offers some concluding remarks.

[‡]For example, see [5, 29, 13, 12, 18, 17, 14, 24, 27].

# 2 Fundamentals of information theory

Forecasting of high-frequency data presents an interesting challenge, in that the forecasting algorithms are required to process the incoming information and make a prediction faster than the data is being supplied. In other words, the algorithms need to function in an "on-line" fashion, processing and making trading decisions in a timely manner. Our research has led to a development of forecasting algorithms that are based on current state-of-the-art methods in predictive computing and asymptotically optimal data compression algorithms [12, 26]. To motivate our rationale for choosing entropy encoding algorithms as the skeleton of forecasting, we present some fundamental notions from information theory.

## 2.1 Basics of probability theory

In order to properly introduce the concept of entropy and other necessary information-theoretic notions we give a short exposition on probability theory. We aim to present only the most relevant and crucial definitions that should make the reader properly interpret the forthcoming entropy notions.

### 2.1.1 Probability spaces

In order to properly define the notion of *entropy*, we posit that a probability space contains all the necessary mathematical structure to give a quantitative measurement of uncertainty associated with a random source.

A *probability space* [§] is given by a triple $(\mathbf{\Omega}, \mathcal{F}, \mathcal{P})$ where $\mathbf{\Omega}$ is a set, $\mathcal{F}$ is a $\sigma$-field [¶] of subsets of $\mathbf{\Omega}$, and $\mathcal{P} : \mathcal{F} \mapsto \Re$ a nonnegative real-valued function such that if $\{E_n\}$ is a countable pairwise disjoint collection of members of $\mathcal{F}$, then

$$\begin{aligned} \mathcal{P}(\bigcup E_n) &= \sum \mathcal{P}(E_n); && (1) \\ \mathcal{P}(\mathbf{\Omega}) &= 1. && (2) \end{aligned}$$

The triple $(\mathbf{\Omega}, \mathcal{F}, \mathcal{P})$ is often referred to as the sample space, elements of $\mathbf{\Omega}$ are *outcomes*, members of $\mathcal{F}$ are *events* and $\mathcal{P}$ is the *probability* of an event.

**Definition 2.1** *A countable collection, $\Xi$ of measurable* [‖] *sets is said to be a* basis *for the probability space $(\mathbf{\Omega}, \mathcal{F}, \mathcal{P})$ if*

- *For every $\omega, \omega' \in \Omega$, there exists a $B \in \Xi$ such that $\omega \in B, \omega' \notin B$ or vice versa.* [**]

[§]For a more thorough treatment of the mathematical theory of entropy, measure and probability theory see [2, 23, 20].

[¶]that is $\mathcal{F}$ is closed under countable unions, complements and contains $\mathbf{\Omega}$.

[‖]A set is measurable if it is in the domain of $\mathcal{P}$.

[**]A basis is well-defined as $\mathbf{\Omega}$ is a Hausdorff space [28].

- *The completion of the $\sigma$-field generated by $\Xi$ is $\mathcal{F}$.*

**Definition 2.2** *A space is a Lebesgue space if it is a totally finite measure space with respect to some basis.* [††]

### 2.1.2 Information theory

Let A be an event in $\mathcal{F}$, such that $(\boldsymbol{\Omega}, \mathcal{F}, \mathcal{P})$ is a Lebesgue space. We remind the reader that the Lebesgue space is most easily interpreted as an abstract model of a stochastic source.

Suppose an event A occurs. As a result we have gained some information about the source. We quantify this notion by introducing a function over the Lebesgue space, called the *information function.* The function so defined can also be interpreted as giving a measure of the *uncertainty* of an event A output by the source. Formally, we give the following definition.

**Definition 2.3** *Information function (or uncertainty function), $\mathcal{I} : \Re \mapsto \Re$ is a real valued function of probabilities of the outputs of a random source, with the following properties:*

- $\mathcal{I}(\mathcal{P}(A)) = 0$, *if* $\mathcal{P}(A) = 1$.
- *If* $\mathcal{P}(A) < \mathcal{P}(B)$, *then* $\mathcal{I}(\mathcal{P}(B)) < \mathcal{I}(\mathcal{P}(A))$.
- $\mathcal{I}(\mathcal{P}(A \wedge B)) = \mathcal{I}(\mathcal{P}(A)) + \mathcal{I}(\mathcal{P}(B))$.

To derive an analytic expression for $\mathcal{I}$, note that information only depends on the probability of events, and not on the events themselves. In addition, the domain of $\mathcal{I}$ is the unit interval $[0, 1]$ and $\mathcal{I}(1) = 0$. In order to satisfy the last property, for two independent events $A, B$, we need $\mathcal{I}(\mathcal{P}(A \bigcap B)) = \mathcal{I}(\mathcal{P}(A)) + \mathcal{I}(\mathcal{P}(B))$. Since $\mathcal{P}(A \bigcap B) = \mathcal{P}(A) + \mathcal{P}(B)$, the information function has to satisfy

$$\forall x, y \in \Re; \qquad \mathcal{I}(xy) = \mathcal{I}(x) + \mathcal{I}(y) \tag{3}$$

It is well known that the only function satisfying the above algebraic relationship is for $\mathcal{I}(t) = K \log t$. In order for information to be a monotone nonincreasing function, we need $K < 0$. We formally define the information associated with an event $E \in \mathcal{F}$ as:

$$\mathcal{I}(t) = \begin{cases} -\gamma \log t, & \text{for } \gamma > 0, t \in [0, 1] \\ \infty, & t = 0, \end{cases}$$

where $t = \mathcal{P}(E)$, and $\log(x)$ is taken to be base $e$. We also note that information is a measurable function with respect to $\mathcal{F}$.

We now formally define the *entropy* of a source which will in turn lead us to introduce entropy encoding algorithms for forecasting.

[††]For a more through treatment of Lebesgue spaces, the reader should consult [20].

### 2.1.3 Information function and entropy

We define the entropy functional of a discrete-time stochastic source and present the definitions of joint and conditional entropies followed by a brief description of the relationship between the entropy and mutual information. Given the basic definitions, we introduce a family of encoding algorithms whose encoding of a given random variable is independent of the underlying distribution and achieves an asymptotic complexity equal to that of the conditional entropy of the encoded source.

In the previous section, we defined a general information function whose domain was the range of a probability function over a Lebesgue space. We now focus on a particular form of that function by considering the information contained in the output of a random source. Theoretically, one can regard that output as a partition of the underlying Lebesgue space. This, in turn will define the constant $\gamma$.

Let $\chi$ be a countable partition of the Lebesgue space $(\boldsymbol{\Omega}, \mathcal{F}, \mathcal{P})$. The information function of such partition is a function defined on $\boldsymbol{\Omega}$ whose value for any $\omega \in \Omega$ is the information obtained from the realization of that element of $\chi$ which contains $\omega$. Formally,

$$\mathcal{I}(\chi)(\omega) = -\gamma \sum_{A \in \chi} 1_A(\omega) \log \mathcal{P}(A), \tag{4}$$

where $1_A$ is the indicator function of the set A. We assert that $\mathcal{I}(.)$ have constant values over each output of the random source, so that the collection of all outputs induces a partition $\chi$, which in turn results in information being a step function.

Let $\Pi$ be the collection of all triplets $(\boldsymbol{\Omega}, \mathcal{F}, \mathcal{P})$ over a Hilbert space. For any such probability space there corresponds a probability distribution $F(x) \in \Pi$ induced by the measure $\mathcal{P}$ on $\boldsymbol{\Omega}$. We formally define the *entropy* functional $\mathcal{H}(\mathcal{P})(x) : \Pi \mapsto \Re$.

**Definition 2.4** *The entropy of a countable measurable partition $\chi$ of a Lebesgue space $(\boldsymbol{\Omega}, \mathcal{F}, \mathcal{P})$ is the expected value of the information function of the partition. The entropy of the partition can be interpreted as the entropy of the distribution induced by the partition, and is given by*

$$\begin{aligned} \mathcal{H} &= \int_{\boldsymbol{\Omega}} \mathcal{P}(d\omega)\mathcal{I}(\chi)(\omega) && (5) \\ &= -\gamma \sum_{A \in \chi} \mathcal{P}(A) \log(\mathcal{P}(A)). && (6) \end{aligned}$$

Therefore, the entropy of a distribution underlying the output of a source is the average amount of information contained in the output, which is represented by the distribution.

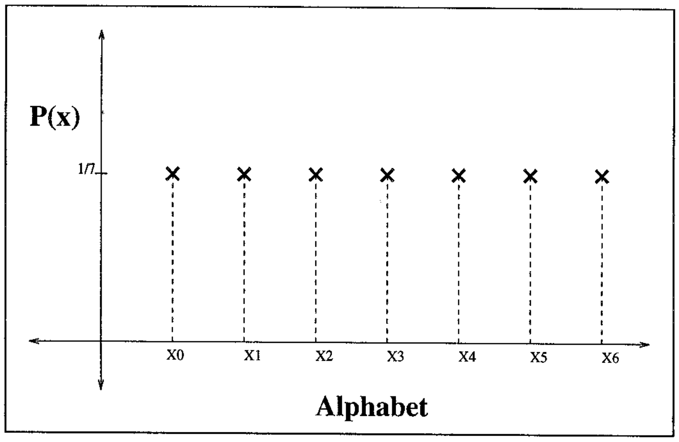

Figure 1: Uniform distribution over 7 letters

The entropy of a source is therefore a measure of the degree of randomness associated with the output of the source. We provide several examples of entropies for particular distributions.

**Example 1** - Let $R$ be a source over a finite alphabet with $n$ letters, $\{x_0, ..., x_n\}$, such that each letter has equal probability being output by $R$. Then, $\mathcal{P}(x_i) = \frac{1}{n}$, and the entropy is

$$\begin{aligned} \mathcal{H}(\mathcal{P}) &= -\sum_{j=1}^{n} \frac{1}{n} \log(\frac{1}{n}) && (7) \\ &= -n\frac{1}{n} \log(\frac{1}{n}) && (8) \\ &= -\log(\frac{1}{n}) && (9) \\ &= \log n. && (10) \end{aligned}$$

Note that as $n \mapsto \infty$, the entropy tends to infinity. On the other hand, the entropy of a deterministic process (for example a toss of a two-headed coin) is 0. In other words, there is no uncertainty associated with the toss.

**Example 2** - Let $R$ be a source whose underlying distribution is normal with mean $\mu$ and variance $\sigma^2$, or $N(\mu, \sigma^2)$. The entropy is independent of the mean and is given by:

$$\mathcal{H}(N(\mu, \sigma^2)) = -\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \log(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}) \qquad (11)$$

$$
\begin{aligned}
&= -\log(\frac{1}{\sqrt{2\pi}\sigma})\int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x-\mu)^2}{2\sigma^2}} && (12)\\
&\quad +\frac{1}{2}\int_{-\infty}^{\infty}\frac{(x-\mu)^2}{2\sigma^2}\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x-\mu)^2}{2\sigma^2}} && (13)\\
&= \frac{1}{2}+\log(\sqrt{2\pi})+\log\sigma && (14)\\
&= \log\sigma+\frac{1}{2}\left[\log(2\pi)+1\right]. && (15)
\end{aligned}
$$

In the next section, we restrict ourselves to discrete time sources, and define *mutual information*, a measure of the amount of information one discrete random variable contains about another. One of the best interpretations of entropy is the amount of "*self-information*" in a random variable. Mutual information can be viewed as a special case of the more general measure coined "*relative entropy*", which is used to metrize the space of distributions.

## 2.2 Entropy of discrete sources

Let $X$ be a discrete random variable over a finite alphabet $\mathcal{A}$, and a probability mass function $f_X(x) = Pr\{X = x\}$, $x \in \mathcal{A}$. For notational convenience, we omit the subscript indicating the random variable and, if necessary, we use different arguments such as $f(x)$ and $f(y)$ to denote $f_X(x)$ and $f_Y(y)$, respectively. We define the **entropy**, $\mathcal{H}$ of a discrete random variable $X$ as:

$$\mathcal{H}(X) = -\sum_{x\in\mathcal{A}} f(x)\log f(x).$$

Let $E$ denote the expectation functional. Therefore for $X \simeq f(x)$, we define

$$E_f g(X) = \sum_{x\in\mathcal{A}} g(x)f(x).$$

Armed with this notation, we can interpret the entropy of a random variable as the expected value of $\log\frac{1}{f(x)}$. Some basic properties of the entropy functional are:

$$\mathcal{H} \geq 0,$$
$$\mathcal{H}_b(X) = (\log_b a)H_a(X).$$

We now turn towards joint and conditional entropies. Define the *joint entropy*, $\mathcal{H}(X,Y)$ of random sources $X$ and $Y$, whose joint distribution $f(x,y)$ is assumed to be well-defined:

$$\mathcal{H}(X,Y) = -\sum_{x\in\mathcal{A}}\sum_{y\in\mathcal{B}} f(x,y)\log f(x,y),$$

which has the expectational interpretation: $\mathcal{H}(X,Y) = -E(\frac{1}{\log f(X,Y)})$.

We also define the conditional entropy $\mathcal{H}(Y|X)$ of a source $Y$, conditioned on an information source, $X$ as:

$$\begin{aligned}
\mathcal{H}(Y|X) &= \sum_{x\in\mathcal{A}} f(x)\mathcal{H}(Y|X=x) && (16)\\
&= -\sum_{x\in\mathcal{A}} f(x) \sum_{y\in\mathcal{B}} f(y|x)\log f(y|x) && (17)\\
&= -\sum_{x\in\mathcal{A}}\sum_{y\in\mathcal{B}} f(x,y)\log f(y|x) && (18)\\
&= -E_{f(x,y)}\log f(Y|X). && (19)
\end{aligned}$$

In short, the conditional entropy of a random source given another is the expected value of the entropies of the conditional distributions, averaged by the conditioning information. From here, it is easy to see that the entropy of a set of random sources is the entropy of one source plus the conditioning entropy of the rest. For example, given a pair of sources, $X$ and $Y$, we have:

$$\mathcal{H}(X,Y) = \mathcal{H}(X) + \mathcal{H}(Y|X).$$

[‡‡] Next, we introduce yet another fundamental notion, that of relative entropy and mutual information.

## 2.3 Relative entropy and mutual information

As we have noted above, the entropy functional can be regarded as a measure of uncertainty or randomness of a random variable. In other words, it is the expected amount of side information required to completely describe a random variable.

Consider a linear functional on the space of distributions, called the Kullback-Leibler functional, also known as the relative entropy:

**Definition 2.5** *The Kullback-Leibler functional $D(a||b)$ = the "distance of $a$ to $b$" on the distribution space is given by:*

$$\begin{aligned}
D(f||g) &= \sum_{x\in A} f(x)\log\frac{f(x)}{g(x)}\\
&= E_f \log\frac{f(X)}{g(X)}
\end{aligned}$$

Note that the complement of the support set for the Kullback-Leibler functional is all $f, g$, such that $f = g$, almost surely in $\mathcal{A}$. One attractive feature of the relative entropy is that it gives us a measure of *the inefficiency of assuming a*

[‡‡]See [5] for a full proof.

*random variable has as its underlying distribution $g$, when the true distribution is in fact $f$.* We will show that this measure has direct links to statistical likelihood function.

Suppose the true distribution of a source is known. Then, there exists a program describing the source with an expected average length equal to $\mathcal{H}(f)$. If we by mistake used the program for distribution $g$, we would need on the average $\mathcal{H}(f) + D(f||g)$ nats to describe the source. Entropy is usually calculated using log() base 2, and is given in units of "bits". We opt for the natural logarithm, and hence our entropy is in the units of "nats". In the next section we rigorously derive the relationship between the relative entropy and the standard likelihood estimators in statistics.

To put some metric on the amount of information one random variable contains about another, we introduce the *mutual information* functional on $\mathcal{P}$ [5].

**Definition 2.6** *Let $f(x,y)$ be the joint probability density function (pdf) of two random variables $X$ and $Y$, and let $f(x)$ and $f(y)$ denote their the respective marginal pdf's. The* mutual information *functional $I(X;Y)$ is defined as the relative entropy between the joint distribution and the product distribution $f(x)f(y)$, i.e,*

$$\begin{aligned} I(X;Y) &= \sum_{x\in\mathcal{A}}\sum_{y\in\mathcal{B}} f(x,y)\log\frac{f(x,y)}{f(x)f(y)} \\ &= D(f(x,y)||f(x)f(y)) \\ &= E_{f(x,y)}\log\frac{f(X,Y)}{f(X)f(Y)} \end{aligned}$$

### 2.3.1 Information theory and statistics

In this section we establish the relationship between information-theoretic notions derived above and the standard hypothesis testing in statistics as given in [5]. We also discuss the work done in establishing the relationship between the concept of maximum entropy of a random source and to Bayesian nonparametric statistics.

Suppose $\{X_0,...,X_n\}$ are independent and identically distributed random variables, with the distribution $\mathcal{P}(x)$. Consider the following hypothesis:

$$\begin{aligned} \bullet H_1 &: \mathcal{P} = \mathcal{P}_1. \\ \bullet H_2 &: \mathcal{P} = \mathcal{P}_2. \end{aligned}$$

Consider a decision function $\psi(x_1,...,x_n)$, such that $\psi(x_1,...,x_n) = 1$ implies that that hypothesis $H_1$ is accepted and $\psi(x_1,...,x_n) = 2$ implies $H_2$ is accepted. Let the set A be such that $H_1$ is accepted for all $x_i$ in A. Let $A^c$ be its complement. Define the probability of error:

$$\alpha = \Pr\{\psi(X_1,...,X_n) = 2|H_1\} = \mathcal{P}_1(A^c) \quad (20)$$

$$\beta = \Pr\{\psi(X_1,...,X_n) = 1|H_2\} = \mathcal{P}_2(A^c) \quad (21)$$

We wish to minimize both probabilities. Best achievable result is given by Stein's lemma [5]. First, we state the following theorem.

**Theorem 2.1** *(Neyman-Pearson): Let $X_1, ..., X_n$ be drawn i.i.d. with a probability mass function (pmf), $\mathcal{P}$. Consider the above decision problem. For a given $T \geq 0$, define a set*

$$A_n(T) = \left\{ \frac{\psi_1(x_1, ..., x_n)}{\psi_2(x_1, ..., x_n)} > T \right\}$$

*Let $\alpha$ and $\beta$ be as defined above. Let $B_n$ be any other region with its corresponding probabilities of error $\alpha'$ and $\beta'$. If $\alpha \leq \alpha'$, then $\beta \geq \beta'$.*

*Proof:* See [5] for a full proof.

The Neyman-Pearson lemma tells us that the optimum test for the distribution hypothesis is of the form

$$\frac{\psi_1(x_1, ..., x_n)}{\psi_2(x_1, ..., x_n)} > T \tag{22}$$

This, of course is the well known likelihood ratio test, and the ratio $\frac{\psi_1(x_1,...,x_n)}{\psi_2(x_1,...,x_n)} > T$ is the likelihood ratio. As an example, consider the test applied to two normal distributions, say $\psi_1 = N(1, \sigma^2)$, and $\psi_2 = N(-1, \sigma^2)$. The corresponding likelihood ratio is

$$\begin{aligned} \frac{\psi_1(x_1, ..., x_n)}{\psi_2(x_1, ..., x_n)} &= \frac{\Pi_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(X_i-1)^2}{2\sigma^2}}}{\Pi_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(X_i+1)^2}{2\sigma^2}}} & (23) \\ &= e^{\frac{2\sum_{i=1}^n X_i}{\sigma^2}} & (24) \\ &= e^{\frac{2n\overline{X}_n}{\sigma^2}} & (25) \end{aligned}$$

We see that for this case the likelihood test consists of comparing the sample mean with a threshold. For the two probabilities of error to equal, $T$ should be set to 1. Let us now rewrite the likelihood test by looking at the *log-likelihood* ratio:

$$\begin{aligned} L(X_1, ..., X_n) &= \log \frac{\psi_1(x_1, ..., x_n)}{\psi_2(x_1, ..., x_n)} & (26) \\ &= \sum_{i=1}^n \log \frac{\psi_1(X_i)}{\psi_2(X_i)} & (27) \\ &= \sum_{a \in \mathcal{F}} n\psi_{X^n}(a) \log \frac{\psi_1(a)}{\psi_2(a)} & (28) \end{aligned}$$

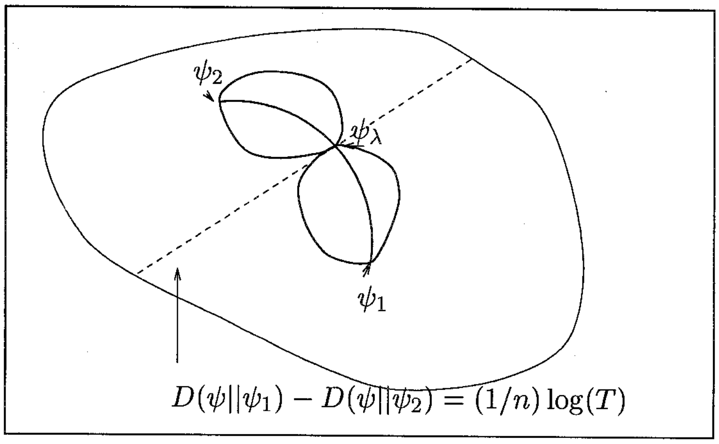

Figure 2: Likelihood ratio test in the probability space

$$= \sum_{a\in\mathcal{F}} n\psi_{X^n}(a)\log\frac{\psi_{X^n}(a)}{\psi_2(a)} - \sum_{a\in\mathcal{F}} n\psi_{X^n}(a)\log\frac{\psi_{X^n}(a)}{\psi_1(a)} \tag{29}$$

$$= nD(\psi_{X^n}||\psi_2) - nD(\psi_{X^n}||\psi_1), \tag{30}$$

which is simply the difference in the "distances" of the sample to each of the distributions. Therefore, we infer that the standard likelihood test

$$\frac{\psi_1(x_1,...,x_n)}{\psi_2(x_1,...,x_n)} > T \tag{31}$$

is equivalent to

$$D(\psi_{X^n}||\psi_2) - D(\psi_{X^n}||\psi_1) > \frac{\log T}{n}. \tag{32}$$

The likelihood test has an interesting geometrical interpretation. The optimum region is defined by the above equation, whose boundary is the region equidistant to the differences of the two distributions, as given in Figure 2. We briefly comment on the optimal choice of threshold. Let $B$ be the set of events on which hypothesis 1 is accepted. Then the probability of error is $\alpha = \psi(\psi_{X_n} \in B^c)$. It can be shown [5] that the complement of $B$ is a convex set, and hence the error is given by the distance of the set $B^c$ to $\psi_1$. Let $\psi^*$ be such point of $B^c$ closest to $\psi_1$. Then,

$$\alpha_n = 2^{-nD(\psi_1^*||\psi_1)}. \tag{33}$$

Similarly,

$$\beta_n = 2^{-nD(\psi_2^*||\psi_2)}, \tag{34}$$

where $\psi_2^*$ is the closest point in $B$ to $\psi_2$. We then minimize $D(\psi||\psi_2)$, subject to the linear constraint $D(\psi||\psi_2) - D(\psi||\psi_1) = \frac{\log T}{n}$. This can be accomplished by writing down the lagrangian,

$$\mathcal{J}(\psi) = \sum \psi(x) \log \frac{\psi(x)}{\psi_2(x)} + \lambda \sum \psi(x) \log \frac{\psi(x)}{\psi_2(x)} + \nu \sum \psi(x). \tag{35}$$

The optimal $\psi_{\lambda^*}$ is given by:

$$\psi_2^* = \psi_{\lambda^*} = \frac{\psi_1^\lambda(x)\psi_2^{1-\lambda}(x)}{\sum_{a\in\mathcal{F}} \psi_1^\lambda(a)\psi_2^{1-\lambda}(a)}. \tag{36}$$

Note that the errors are exponential in $D(\psi^*||\psi_1)$ and $D(\psi^*||\psi_2)$.

Continuing with development of computational bounds in likelihood estimation, we state the Stein's lemma which gives the optimal error bounds of the likelihood estimator.

**Theorem 2.2** *(Stein's Lemma) - Let the assumptions of the Neyman-Pearson lemma hold. Furthermore, let $D(\psi_1||\psi_2) < \infty$. Let $A_n \subset \mathcal{F}$ be an acceptance region for the hypothesis 1. The probabilities of error are given by*

$$\alpha_n = \psi_1^n(A_n^c), \qquad \beta_n = \psi_2^n(A_n^c), \tag{37}$$

*and define* $\beta_n^\epsilon = \overset{Min}{A_n \subset \mathcal{F}}, \alpha_n < \epsilon\, \beta_n$. *Then,*

$$\lim_{\epsilon\mapsto 0} \lim_{n\mapsto\infty} \frac{\log \beta_n^\epsilon}{n} = -D(\psi_1||\psi_2). \tag{38}$$

Therefore, *no* sequence of sets $B_n$ can have an associated exponent better than $D(\psi_1||\psi_2)$. It can be shown that the if we define

$$A_n = \left\{ x \in \mathcal{F} : 2^{n(D(\psi_1||\psi_2)-\delta)} \le \frac{\psi_1(x)}{\psi_2(x)} \le 2^{n(D(\psi_1||\psi_2)+\delta)} \right\}, \delta > 0, \tag{39}$$

the sequence of sets $A_n$ achieves the exponent $D(\psi_1||\psi_2)$. Therefore, the sequence is asymptotically optimal, and the best error exponent is $D(\psi_1||\psi_2)$. Hence, Stein's lemma gives tight computational bounds of the goodness of the likelihood estimator in terms of *computable* quantities $D(\bullet||\bullet)$.

Next, we briefly touch on the important connection in Bayesian approach to hypothesis testing and the equivalent computable notion of Chernoff information. We assign a priori probabilities to both hypothesis and minimize the overall probability of error that is given by a weighed sum of the individual probabilities of error. That exponent turns out to be precisely the Chernoff information.

Let $\{X_1, ..., X_n\}$ be i.i.d., distributed according to $\psi(x)$. Let hypothesis 1 be: $\psi = \psi_1$, with prior probability $p$ and hypothesis 2: $\psi = \psi_2$, with probability $q$. Then, the overall probability of error is given by

$$\psi_e^n = p\alpha_n + q\beta_n. \tag{40}$$

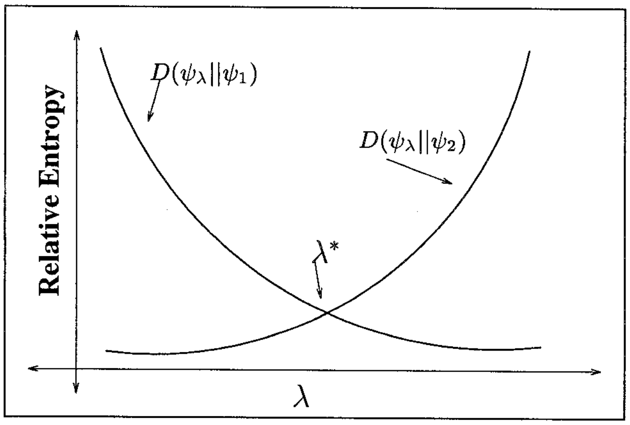

Figure 3: Relative entropies as functions of $\lambda$.

Let $D^* = \lim_{n\mapsto\infty} \min_{A_n \subset \mathcal{F}} -\frac{\log \psi_e^n}{n}$. Then, we can state the following theorem:

**Theorem 2.3** *(Chernoff): The best achievable exponent in the Bayesian probability of error is $D^*$, where*

$$D^* = D(\psi_{\lambda^*}||\psi_1) = D(\psi_{\lambda^*}||\psi_2), \tag{41}$$

*such that*

$$\psi_\lambda = \frac{\psi_1^\lambda(x)\psi_2^{1-\lambda}(x)}{\sum_{a\in\mathcal{F}}\psi_1^\lambda(a)\psi_2^{1-\lambda}(a)} \tag{42}$$

*and $\lambda^*$ is such that*

$$D(\psi_{\lambda^*}||\psi_1) = D(\psi_{\lambda^*}||\psi_2). \tag{43}$$

*The value $D(\psi_{\lambda^*}||\psi_1) = (\psi_{\lambda^*}||\psi_2) = C(\psi_1, \psi_2)$ is called the Chernoff information, and is the highest achievable exponent for the probability error.*

Note that the maximum a posteriori probability decision rule minimizes the Bayesian probability of error and that the Bayes exponent does not depend on the a priori probabilities. The effect of the prior is negligible in large samples. The optimal decision rule is to choose the maximal a posteriori probability which can be written as the test

$$\frac{p\psi_1(X_1, ..., X_n)}{q\psi_2(X_1, ..., X_n)} \geq 1. \tag{44}$$

Taking logs and dividing by $n$ and taking limits, we get

$$\lim_{n \mapsto \infty} (\frac{1}{n} \log \frac{p}{q} + \sum_i \log \frac{\psi_1(X_i)}{\psi_2(X_i)} \leq 0) \tag{45}$$

$$\tag{46}$$

is $D(\psi_1||\psi_2)$ if $\psi_1$ is the true distribution, or the limit tends to $-D(\psi_1||\psi_2)$ otherwise. The first term above tends to zero, verifying the fact that the prior probabilities do not contribute to the statistic.

We now introduce some fundamental relationships between entropy and mutual information, the notion of complexity and present an asymptotic compression algorithm that is independent of the distribution of the source and achieves asymptotic complexity equal to that of the unconditional entropy of the encoded source.

It can be shown that the mutual information $I(X;Y)$ is the reduction in the uncertainty of $X$ due to knowledge of $Y$. That is, $I(X;Y) = \mathcal{H}(X) - \mathcal{H}(Y)$. By symmetry we have the following properties:

$$\begin{aligned} \mathcal{I}(X;Y) &= \mathcal{H}(X) - \mathcal{H}(X|Y), & (47) \\ \mathcal{I}(X;Y) &= \mathcal{H}(Y) - \mathcal{H}(Y|X), & (48) \\ \mathcal{I}(X;Y) &= \mathcal{H}(X) + \mathcal{H}(Y) - \mathcal{H}(X,Y), & (49) \\ \mathcal{I}(X;Y) &= \mathcal{I}(Y;X), & (50) \\ \mathcal{I}(X;X) &= \mathcal{H}(X). & (51) \end{aligned}$$

The above relationships completely characterize mutual information and entropy. Our final functional, the conditional relative entropy is defined below.

**Definition 2.7** *The* conditional relative entropy functional $D(f(y|x)||g(y|x))$ *is the average of the relative entropy between the conditional pmf's* $f(y|x)$ *and* $g(y|x)$*, averaged over the probability mass function* $p(x)$*. I.e.,*

$$\begin{aligned} D(f(y|x)||g(y|x)) &= \sum_{x \in \mathcal{A}} f(x) \sum_{y \in \mathcal{B}} f(y|x) \log \frac{f(y|x)}{g(y|x)} & (52) \\ &= E_{f(x,y)} \log \frac{f(Y|X)}{g(Y|X)} & (53) \end{aligned}$$

We now turn out attention to entropy-based compression and present a result relating the entropy of a source and its fundamental limit of compression of the random variable.

## 2.4 Prefix coding

In this section we present some important relationships between the optimal compressibility of a source and its entropy. We begin by defining the notion of an instantaneous code, followed by the statement of Kraft inequality [5], asserting that the exponentiated codeword length mappings must look like probability mass functions. This would imply that the expected codeword length must be greater than or equal to the entropy of the encoded source. Following, we note an asymptotically optimal encoding, the Huffman code. After establishing the entropy measure as the data compression limit, we will introduce a provably optimal encoding algorithm for a stationary ergodic source, on which we base our foreign exchange forecasting model.

**Definition 2.8** *A source code $C$ for a random variable $X$, is a mapping from $\mathcal{A}$, the range of $X$, to $\mathcal{D}^*$, the set of finite length strings of symbols from a D-ary alphabet. Let $C(x)$ denote the encoding of $x$, and let $l(x)$ denote the length of the encoding.*

**Definition 2.9** *The expected length $L(C)$ of a source code $C(x)$ for a random variable $X$ with pmf $f(x)$ is*

$$L(C) = \sum_{x \in \mathcal{A}} f(x) l(x),$$

*where $l(x)$ if the length of the encoding corresponding to $x$.*

**Definition 2.10** *A code is a* prefix *or* instantaneous *code if no codeword is a prefix of any other codeword.*

**Theorem 2.4** *(Kraft inequality): For any prefix code over an alphabet of size D, the codeword lengths satisfy:*

$$\sum_i D^{-l_i} \leq 1.$$

See [5] for a full proof. We now discuss the concept of finding shortest instantaneous codes.

## 2.5 Optimal codes

The Kraft inequality is a sufficient condition for existence of an encoding of a random variable with a specified set of codeword lengths. Any codeword set that satisfies the prefix condition has to satisfy the Kraft inequality. Generating a prefix code with minimal expected length is equivalent to minimizing:

$$\min L = \sum f_i l_i,$$

over all integer $l_1, l_2, ..., l_m$ such that,

$$\sum D^{-l_i} \leq 1.$$

This problem is solved by forming the Lagrangian:

$$J = \sum f_i l_i + \lambda(\sum D^{-l_i}).$$

Solving, it can be shown that the optimal codelengths $l_i^* = -\log_D f_i$, where $f_i = D^{-l_i}$. Note that this optimization was solved over continuous codeword lengths. We therefore conclude that the optimal expected codeword length $L^*$ is given by:

$$L^* = \sum f_i l_i^* = -\sum f_i \log_D f_i = \mathcal{H}_D(X).$$

The above result is easily extended to integer codeword lengths.

The main result implied by the above is: to find an optimal encodings we need to find a D-adic distribution that is the closest, under the relative entropy measure, to the real distribution of $X$. In turns, this distribution will provide a set of encoding lengths. The encoding construction can be found in [5].

The obvious problem is finding the appropriate D-adic distribution. The well known optimal procedure for actually finding the optimal code, the Huffman code, can be found in [5]. We do not investigate Huffman encoding because it assumes a-priori knowledge of the probability distribution of the source, and therefore is inappropriate for forecasting on financial data whose distribution in not known. Instead, we investigate an encoding algorithm whose asymptotic rate approaches the entropy of the source, and which is independent of the distribution.

## 2.6 Entropy encoding

An example of lossless encoding, the entropy code is ideally suited as a forecasting algorithm of a set of quantized or raw data. Its complexity is logarithmic in the size of the input, and hence is very promising for on-line forecasting of high-frequency data. We introduce the entropy encoding via an example.

Suppose $X$ is a random source over the alphabet $\{0, 1\}$. Let $z_i = \{X_1, ..., X_i\}$, where $X_k = 1 \vee 0$. Suppose $z_i = \{100011110\ 011101\ 1001011\ 10010101101\}$ (Note that the entropy encoding can be extended over an arbitrary alphabet). The algorithm breaks the source into substrings that had not been parsed so far. In our example, the string $z_i$ would be parsed as: [1]; [0]; [00]; [11]; [110]; [01]; [1101]; [10]; [010]; [111]; [001]; [0101]; [101]. Such parsing builds a tree, where a child-node is added whenever a new string has been parsed. A tree-diagram for this particular string is included. In figure 4, each node has two attributes, namely $\alpha, \beta$, where $\alpha$ is either 0 or 1, and $\beta$ denotes the number of times the particular node has been visited during the construction of the tree. Here we can see the tree structured version of the encoding for the string. Interpreting the encodings as trees gives a rise to a natural measure of the asymptotic entropy

An encoding is said to be "lossless" if the decoded output is the same as the original input. In other words, there is no loss of information.

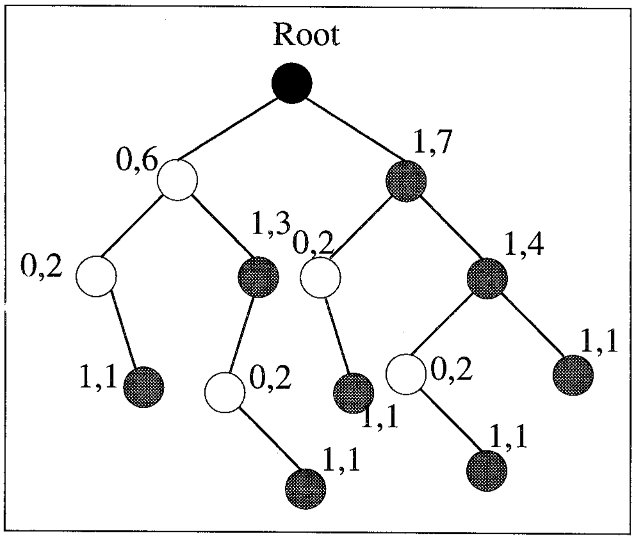

Figure 4: Tree encoding

of the source as the inverse of the ratio of the length of the original input to the length of its encoding, also known as the compression ratio of the source. For example, a string $l = 1111111111111...$ would be parsed as $[1]; [11]; [111]; [1111]$ etc. The corresponding tree would be a *chain*. The compression ratio as a function of $l$ is $\Theta(\sqrt{n})$, for this particular string, and the entropy therefore $\mathcal{H}(l) = \Theta(\frac{1}{\sqrt{n}})$ bits, which goes to zero as $n$ tends to infinity. On the opposite end of the spectrum, given a "randomly" generated string, over 0,1 the asymptotic structure of the tree would be a full binary tree, whose compression ratio is $\Theta(1)$, and whose entropy is $\Theta(1)$ bits. In other words, for a totally random input, there is no predictability. Conversely, for sources whose compression ratio tends to infinity, we would have complete (asymptotic) predictability.

The algorithm as given above is an example of an entropy based algorithm that forecasts the output of a random source, given no a priori assumptions about the underlying distribution. Naive forecasting can be done by keeping a count of the number of times a node is traversed in descending down a tree, and taking the more frequently occurring path. For example, if we are at the node $(1, 7)$, the probability that next digit is a 0 is $1/3$, while probability that it is a 1 is $2/3$.

The encoding as given above is a lossless encoding. Led by our intuition in financial trading, we now focus on a class of "*quantizing*" or "*lossy*" encodings, where we regard the output of a source subject to noise and/or the rate of the incoming information is high enough that the techniques are forced to lose some information in order to remain operating in an on-line (timely) fashion.

## 2.7 Lossy encodings

In finance, forecasting of high frequency data is complicated due to the inevitable noise trading. More importantly, the frequency is high enough that the trading algorithm, in order to remain feasible, has to quantize the data. To develop a reliable forecasting methodology, we study the entropy-based quantization algorithms, where the information produced by a source is arriving over a noisy channel. We briefly introduce the concept of scalar quantization (SQ), followed by a more in-depth discussion of vector quantization (VQ). We discuss the basic theoretical justifications behind some of the techniques, followed by a discussion of why neural-net based algorithms for forecasting can be viewed as (simple) special cases of the more general predictive VQ.

### 2.7.1 Scalar quantization

In information theory, quantization can be regarded as the crux of the analog to digital conversion. To put it simply, quantization is a process of computing the best representative (drawn from a discrete vocabulary) for a given input (for example a real number), under a set of constraints. More formally,

**Definition 2.11** *A monotone map $\pi(x) : \Re \mapsto \mathcal{V}$, $\mathcal{V} \subset \Re$, card$(\mathcal{V}) = N$, is called an $N$-point scalar quantizer. The set $V$ is known as the vocabulary or the codebook for $\pi(x)$.*

In all practical cases $V$ is a finite (an often as small as possible) set of numbers. We also define the distortion measure $\rho$, of a scalar quantizer $\pi$, as $\rho = \log N$, which is a measure of the number of bits needed to uniquely parse a quantized value. Quantization can be fixed-rate or variable-rate, depending on the particular choice or $\rho$. For every monotone map $\pi$, there exists a unique finite element of the power set of $\Re$, which we denote as $\Pi$. Each member of $\Pi$, can be written as: $R_i = \{x \in \Re | \pi(x) = v_i\}$, which is exactly the inverse image of $v_i \in V$, under the map $\pi$. It is clear that $\bigcup R_i = \Re$, and that they are mutually disjoint. The partition sets, in general can be open, closed or half-open intervals. A partition is called regular iff all elements of the partition set are open, and no scalar is mapped onto the separator point between two intervals. The separator points are sometimes called boundary or decision points. Every quantizer can be viewed as a composition of two maps: An encoder map $E : \Re \mapsto \mathcal{Z}$, and a decoder map, $D : \mathcal{Z} \mapsto V$. Therefore, $\pi(x) = D(E)(x)$. We now introduce the concept of Vector Quantization, a generalization of SQ to the quantization of a vector.

## 3 Vector quantization

The generalization of SQ to a general $N$-dimensional vector space results in a powerful quantization technique and introduces some intriguing computational and geometric issues. Whereas SQ is used mostly for analog to digital conversion, VQ is used primarily for state-of-the-art digital processing and in data compression. We will argue that neural-nets are a subclass of vector quantization, and that quantization in the VQ sense provides much more flexibility and a framework for efficient algorithmic implementation. VQ also provides some ground for using entropy-based encoding in $N$-dimensional vector space, whereas neural nets cannot.

VQ is usually, but not exclusively used for the purpose of data compression. A *vector* can be used to describe virtually any type of input data, for example a collection of currencies with side information, simply by forming a vector of the incoming data. VQ has been used, traditionally for encoding of speech and image data [13, 18, 17, 14, 6, 3, 9, 10, 11, 15, 1, 16]. There exist many ways of quantizing multidimensional data. For example, VQ can be viewed as a form of *pattern recognition* where an input pattern is "approximated" by an element in a codebook of patterns. In other words, the pattern produced by a source is matched with a pattern stored in the codebook. VQ can also be viewed as a technique to reduce the space complexity of the input data. We present the VQ in the light of data encoding, pattern recognition and forecasting.

**Definition 3.1** *A mapping $\Pi : \Re^n \mapsto \mathcal{V}$, such that card($\mathcal{V}$) is a finite, k-dimensional subset of $\Re^n$, is called a Vector Quantizer, or VQ.*

The points in $\mathcal{V}$ are called codewords. Parameter $\rho = \frac{logn}{n}$ is called the resolution or the code rate and is used to represent the bit-complexity of the vector components. $\rho$ gives a measure of the accuracy of VQ, provided a codebook is well-defined. However, note that for a codebook of a fixed dimension, $k$, the resolution is determined by the dimension of the input set, and not the number of bits used to specify the elements of the codebook. The codebook is typically represented as a table or a tree data structure.

We turn to discuss some geometrical properties of hypercells * induced by spatial tesselations of the VQ codebook construction. Similar to SQ, the vector quantizer induces a hypercell partition of $\Re^n$. As before, each hypercell can be regarded as an open, bounded subset of $\Re^n$. The boundary hyperplanes are the "decision hyperplanes". A bounded hypercell(or cell, for short) is called a regular cell, an unbounded cell is called an overload cell. The collection of all regular cells is called a regular region.

**Definition 3.2** *A vector quantizer is called "regular" if each hypercell is convex, and each element of $\mathcal{V}$ is an interior point of the hypercell.*

*A hypercell is a d-dimensional region where any point in its interior is quantized to the codebook entry in the cell

It is useful to identify an another quantizer, a polytopal vector quantizer, which is a subset of the regular quantizers, such that the boundary of each of the hypercells are segments of hyperplane surfaces in $n$ dimensions. Or, equivalently, each partition is a regular polytope(a hyper-polyhedra), and consists of an intersection of a finite number of half-spaces. These half-spaces can be written in the form $\{x \in \Re^n : u_\nu \cdot x + \beta_\nu \geq 0\}$.

The faces of a polytopal hypercell are hyperplane segments of dimension less than $n$ that bound the cell, so that it completely determines membership of a point and a cell. In other words, a point of one "side" of a hyperplane belongs to a different hypercell than a point on the other side of the hypercell. A face is the $(n-1)$-dimensional boundary of a hypercell. Note, trivially, SQ always induces a regular quantization.

A vector quantizer is bounded if $\Pi : \mathcal{W} \mapsto \mathcal{V}$, and $\mathcal{W} \subset \Re^n$. A bounded VQ has no overload hypercells. Just as with SQ, a vector quantizer can be viewed as a composition of two functions on $\Re^n$; A *vector encoder*, $E : \Re^n \mapsto \mathcal{I}$, and a *vector decoder*, $J : \mathcal{I} \mapsto \mathcal{V}$. Then, the VQ can be written as $\Pi(x) = J(E)(x)$. Note that a given partition of the $n$-space into hypercells completely determine the encoding map. On the other hand, a given codebook completely determines how the decoder will generate a decoded output vector from a given index. One of the tasks of the encoder is to implicitly or explicitly identify hyper-cell a given vector lies. It is not immediately obvious, but true however, that the encoder does not need to be aware of the codebook. The decoder, similarly, does not need to know the geometry of the hypercell to function. It is interesting to note that most practical VQ algorithms retain only the codebook as enough information to provide a cell partition of $n$-space.

## 3.1 Properties of VQ

In multidimensional quantizing, it is well known [13] vector quantization is the ultimate solution to the quantization of a data vector. No other coding(including neural-nets) exists that can do better than VQ. The following theorem shows that VQ can at least match the performance of any arbitrarily given coding system that operates an a vector of data.

**Theorem 3.1** *For any given coding system that maps a vector into one of $N$ binary words and reconstructs the approximate vector from the binary word, there exists a vector quantizer with codebook size $N$ that gives exactly the same performance, i.e., for any input vector, it produces the same reproduction as the given coding system.*

*Proof:* [13] Enumerate the set of binary words produced by the coding system, as indexes $1, 2, ..., N$. For the $i^{th}$ binary word, let the decoded output of the given coding system be the vector $y_i$. Define the codebook, $C$ as the ordered set of code vectors $y_i$. Then, a VQ decoder achieves equivalent performance to

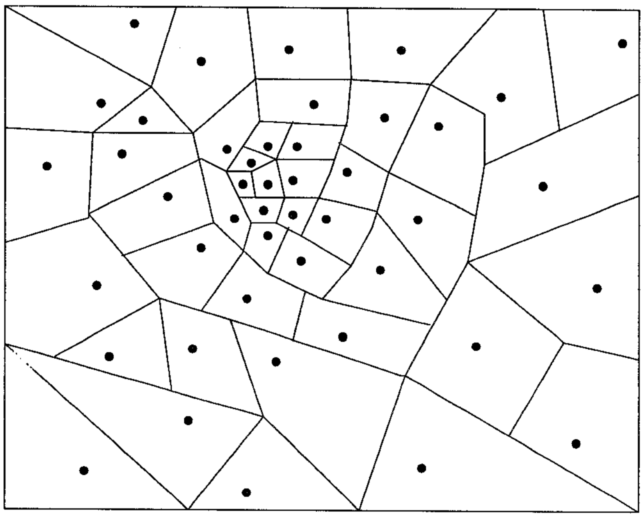

Figure 5: A regular vector quantization on the plane

the decoder of the given coding system, and a VQ encoder can be defined to be identical to the encoder of the given coding system. ▮

Furthermore, in general, any heuristic coding technique, such as neural-nets, encoding a set of $n$ data points with $b$-bits, is a *suboptimal* way to map a $k$-dimensional input vector into one of $N = 2^b$ index values and into one of the reproduction vectors. Since VQ encompasses all such possible encoding schemes, it is only natural encode our source by VQ.

We will introduce the problem of joint optimization of the encoder and predictor in this paper. It should be clear that if an optimal VQ is found for a given performance objective, *no other* encoding system will be able to achieve a better performance.

## 3.2 Examples of VQ

To illustrate VQ, we present the quantization of a bivariate random variable. Given a codebook consisting of $N$ points in the plane, the VQ will assign any particular realization of the source to the appropriate codeword. We show two examples of a quantization in two dimensions. A regular quantizer may separate the cells into perhaps what is shown in the figure below. A non-regular VQ can in general, arbitrarily subdivide the plane. The figure below demonstrates an example of a non-polytopal subdivision.

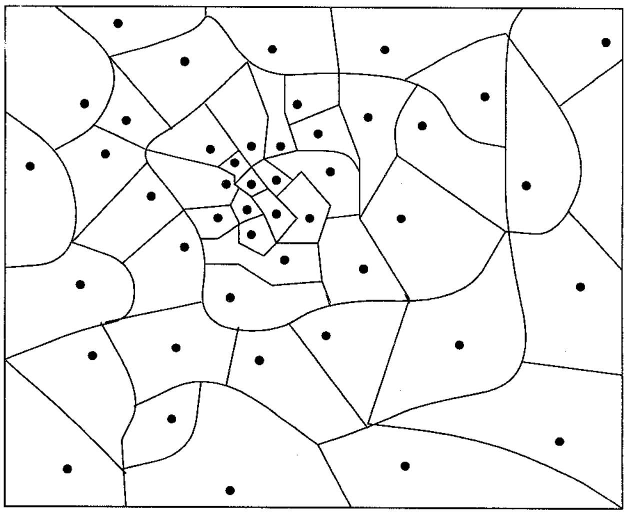
Figure 6: A non-regular vector quantization on the plane

## 3.3 Hypergeometry of VQ

For a given input, the primary structure of a vector quantizer is to identify the appropriate dictionary element and map the input onto the point. In order to achieve this, a VQ needs to be able to identify in which hypercell a particular input point belongs. In one dimension, this question is trivial, as it must lie between two separators. In higher dimensions, this question is nontrivial. For certain classes of VQ algorithms, such as nearest-neighbor or polytopal VQ, such techniques have been developed [13, 21, 22]. For a general VQ, this question can be arbitrarily hard. We examine this question for a polytopal VQ.

A polytopal VQ is such that each cell in the partition is a convex polytope. Consider the half-space $H_\nu$ defined as:

$$H_\nu = \{x \in \Re^d : u_\nu^t x + \beta_n u \geq 0\}, \tag{54}$$

Let $\Lambda_n u$ be the hyperplane bounding the half-space $H_\nu$. Then, any polytopal region can be represented by a finite intersection of half-planes. Therefore, each polytope can be written as:

$$R_i = \bigcap_{\nu=1}^{L_i} H_\nu, \tag{55}$$

such that $L_i$ is the number of $n-1$-dimensional hyperfaces of the cell. For such a polytopal VQ, we define an indicator function $1_{H_n u}(x)$ for the half-space $H_\nu$.

In vector notation,

$$T_\nu(x) = \psi(u_\nu^t x + \beta_n u), \tag{56}$$

where $\psi(v) = 1$ if $v \geq 0$, and equals 1 otherwise. We then define a cell selector function which basically answers the question: "*Is $x$ on the positive side of the hyperplane $\Lambda_\nu$*". To draw some parallels to neural networks, the hyperplane decision functions are known as the binary threshold values, or linear decision functions.

## 3.4 VQ performance

Let $dist(\bullet, \bullet)$ be a metric on the vector space. We define the average distortion measure $D = E(dist(x, \overline{x})$, where $x$ is the input vector and $\overline{x}$ is its encoded value. The distortion measure can be expanded for a sequence of inputs as the average distortion over $n$ inputs, or simply

$$\overline{d} = \lim_{n \mapsto \infty} \frac{1}{n} \sum_{i=1}^{n} dist(x_i, \overline{x}_i). \tag{57}$$

It is worth noting that for a stationary ergodic processes, the above limit is the expectation $D$.

Suppose $x$ is the outcome of a d-variate random source with the a pdf $f_x(x)$. Let $VQ(x)$ the the quantized value of any input. Then, the average distortion is

$$\begin{aligned} D = E(dist(x,y)) &= \int_{\Re^d} dist(x, VQ(x)) f_x(x) dx && (58) \\ &= \sum_{j=1}^{N} \int_{R_j} dist(x,y) f_x(x) dx && (59) \\ &= \int_{R_j} \sum_{j=1}^{N} \mathcal{P}_j dist(x, y_j) f_{x|j}(x) dx && (60) \\ &= \sum_{j-1}^{N} \mathcal{P}_j (E(dist(x, y_j) : x \in R_j)), && (61) \end{aligned}$$

where $\mathcal{P}_j$ is the probability that $x$ is in the hypercell $R_j$.

We comment that as many different distortion measures can be constructed as there are metrics on $\Re^d$. Ones we consider have to be readily computable, in order to give a realistic measure on the goodness of quantization. A common measure is the *Euclidian distortion*,

$$\begin{aligned} dist(x, \overline{x}) &= ||x - \overline{x}||^2 && (62) \\ &= \sum_{i=1}^{d} (x_i - \overline{x}_i)^2. && (63) \end{aligned}$$

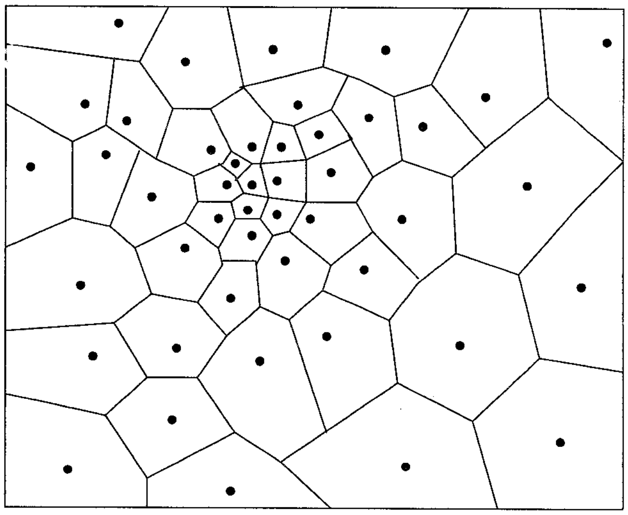

Figure 7: A Vornoi vector quantization of the plane

Then, the average Euclidian distortion is simply $E(||x - \overline{x}||^2)$. An important generalization of the Euclidian distortion is the *generalized* or *weighted Euclidian distortion*,

$$dist(x, \overline{x}) = (x - \overline{x})^t W(x - y), \tag{64}$$

where $W$ is a matrix of weights.

## 3.5 Vornoi quantizers

The Vornoi Vector quantization, or the *Nearest neighbor* quantization is of particular interest, for efficiency and excellent quantizing properties. The Vornoi VQ is completely characterized by the codebook set of vectors and the distortion measure. We shall show that the Vornoi VQ is the optimal VQ in the sense of minimizing distortion. A vector quantizer is *Vornoi* if the partition cells are of the form

$$R_i = \{x : dist(x, y_i) \leq dist(x, y_j) \text{for all} j\}. \tag{65}$$

The following diagram is an example of a Vornoi quantization of the plane. We now present several different quantizer designs, for a given set of points in $\Re^d$. Afterwards, we point out some parallels between VQ and nonparametric statistical techniques in high dimensions.

## 3.6 Vector quantizer methods

This section discusses several iterative methods for optimal codebook design. The quantization techniques described herein are provably optimal, under the aforementioned assumptions. The objective in quantization is to find a codebook and a hyperspace partition rule for a given set of inputs while minimizing some chosen distortion measure. As mentioned, VQ performance can be measured by a statistical average of a distortion measure or a worst-case metric. The statistical average of the distortion of a vector quantizer, $VQ(x)$ is given by

$$D = E(dist(x, VQ(x))) = \int dist(x, VQ(x)) f_X(x) dx \tag{66}$$

where $F_x(x)$ is the joint pdf of X. Naturally, for a discrete source, the above definition ca'ı be stated as

$$D = E(dist(X, VQ(x))) = \sum_j dist(x_j, VQ(x_j)) \mathcal{P}(x_i). \tag{67}$$

In order to find the optimal VQ for a given source, we search for the optimal encoder, given a fixed decoder, and then for a fixed encoder, we find the optimal decoder. The encoder is completely determined by the hyperspace tessellation, and the decoder is completely determined by the associated codebook.

Given a decoder, the optimal partition is one that satisfies the Vornoi condition: For each cell, all inputs closer to the codebook assigned to that cell should be encoded as the codebook value. Thus, we state

**Definition 3.3** *(Vornoi condition) For a given set out outputs, the optimal partition cells satisfy*

$$R_i \subset \{x : dist(x, y_i) \leq dist(x, y_j); for\ all j\}. \tag{68}$$

*Equivalently, we have*

$$VQ(x) = y_i \ \ only\ if\ dist(x, y_i) \leq dist(x, y_j),\ \ for\ all\ j. \tag{69}$$

*Therefore, for a given decoder, the encoder is a minimum distortion mapping, and we have:*

$$dist(x, VQ(x)) = \min_{y_i \in C} dist(x, y_i). \tag{70}$$

Next, we state the centroid condition which ensures that a quantizer is optimal for a given decoder. Define the *centroid* $cent(R_i)$ of any region $R_i$ to be that vector $y^*$ that minimizes the distortion between any interior point in $R_i$ and $y$, averaged over the probability distribution of the source. In other words,

$$y^* = cent(R) \text{ if } E(dist(x, y^*) : x \in R) \leq E(dist(x, y) : x \in R), \tag{71}$$

for all codebook elements $y$. Equivalently, we write

$$cent(R) = \min_{y} E(dist(x,y) : x \in R). \tag{72}$$

The easiest interpretation of the centroid is that the centroid is the best representative of a region under some pre-supposed probability distribution over the region. For the Euclidian distortion, $cent(R) = E(x : x \in R)$. For a bounded cell, we can directly compute the centroid, provided a probability distribution of the source is known, or approximated, if we estimate pdf. In order to estimate the probability distribution, we usually consider some training sequence of data, say the history of a collection of currencies and side information. This is equivalent to the statistical notion of a learning set. In the next section we discuss several methods of determining the distribution of a learning set, and cross-reference these methods to standard nonparametric statistical techniques, but first we complete this section formally defining the so-called centroid condition, which completely characterizes the optimal codebook for a given partition.

**Definition 3.4** *For a given partition $\{R_i : i - 1, ..., N\}$, the optimal codebook satisfies*

$$y_i = cent(r_i) \tag{73}$$

In order to complete the optimality criterion, we formally state the so-called *zero probability boundary condition*, which states that the probability of a source output being equidistant to two codebook entries is zero almost surely.

To this end, let $B_j^i$ be a collection of $M_i$ hyperplanes bounding the cell $R_i$. An obvious condition for optimality of a VQ is that

$$\mathcal{P}(\bigcup_{i=1}^{N} \bigcup_{j=1}^{M_i} B_j^i) = 0,, \text{ a.s.} \tag{74}$$

Equivalently,

$$\mathcal{P}(x : dist(x, y_i) = dist(x, y_j) \exists i \neq j) = 0,, \text{ a.s.} \tag{75}$$

The zero boundary condition is trivially satisfied if the source is continuous, as a countable collection of hyperfaces is of measure zero. For discrete source, the zero condition may be violated with positive probability. To ensure optimality for a VQ, we simply break the tie by re-quantizing. It is easy to see that for a countable number of input vectors, the zero condition can always be avoided.

### 3.6.1 Local and global optimality

An important requirement for a given quantizer is to be globally(and hence locally) optimal. We formally define these notions.

**Definition 3.5** *A quantizer is said to be locally optimal if the distortion measure does not decrease given any local perturbation of the codebook vectors. A quantizer is globally optimal is there does not exist any other codebook resulting in a lower distortion.*

It is well known [13] that a quantizer that satisfies the three optimality conditions will be locally optimal. Unfortunately, a quantizer that is locally optimal may be severely suboptimal globally. Global optimization is a research area on its own, and is very much a function of the random source. For forecasting we intend to do, the quantizer is carefully designed to be globally optimal over a window of time. Before discussing several design issues, we briefly state some important implications of optimal quantizers,

**Theorem 3.2** *A vector quantizer that satisfies optimality conditions under Euclidian distortion is regular.*

*Proof:* Regions satisfying the Vornoi condition are necessarily convex. Given any pdf over subset of $\Re^d$, the expectation is also in the set. Therefore, by the centroid condition, each codebook vector is in the interior of its corresponding partition cell. Hence, the VQ is regular [13]. ▮

**Theorem 3.3** *A quantizer satisfying the centroid condition under the Euclidian distortion satisfies:*

$$\begin{aligned} E(VQ(x)) &= E(x) \qquad (76)\\ E(x^tVQ(x)) &= E(||VQ(x)||^2) \qquad (77)\\ E(||VQ(x)||^2) &= E(||x||^2) - E(||x - VQ(x)||^2). \qquad (78) \end{aligned}$$

*Proof:* We follow [13]. Let $S_i = 1_{R_i}(x)$ denote the indicator function for the region $R_i$. Since the codebook is optimal for the given partition, we have $E((x - VQ(x))S_i) = 0$ for all $i$. Taking a sum over all hypercells, we get $E((x - VQ(x))(\sum_i S_i)) = 0$. Since the support sets of $S_i$ are disjoint, and each $S_i = 1$ on its support set, we have $\sum_i S_i = 1$. Therefore, $E(x - VQ(x)) = 0$. Next, since the codebook is a linear combination of the indicator functions, we immediately have that $E((x - VQ(x))^tVQ(x)) = 0$. Hence the quantization error is not correlated with the output vector. Therefore, $E(x^tVQ(x)) = 0$. Lastly, $E(||x - VQ(x)||^2) = E((x - VQ(x))^tx)$, so therefore we can write $E(||x - VQ(x)||^2) = E(||x||^2) - E(x^tVQ(x)) = E(||x||^2) - E(||VQ(x)||^2)$. ▮

Therefore, we conclude that without loss of generality, optimal quantization can always be designed for zero-mean random variables. For an optimal decoder, the quantization error is always correlated with the input vector.

## 3.7 Vector quantizer algorithms

In this section we briefly mention some state-of-the-art methods for designing optimal quantizers for a given training set of data. We base the design on the optimality conditions, as a basis for an iterative improvement an initial vector quantizer. Every quantizer will be initialized with an initial choice for a codebook and the Vornoi hypercell subdivision. We then search for a new codebook that is optimal for that partition. We seek to minimize the average distortion over each cell. We start by mentioning some of the more popular techniques for an initial codebook choice.

### 3.7.1 Random poling

Given a training set consisting of $N$ points, we randomly choose a set of points $K < N$ representatives, according to the source distribution. This approach has been used extensively in standard pattern recognition literature [13].

### 3.7.2 Pruning

Given a starting set, the idea of pruning is to selectively remove training vectors as potential codewords. The technique proceeds in a sequential fashion by inserting a new vector as the codeword, computing the distortion measure for that codeword, and if the distortion is less than a specified tolerance we continue, otherwise we do not add another codevector. This procedure is continued until the codebook has enough vectors. This technique is also well known in statistical clustering methods [13, 25].

### 3.7.3 Pairwise Vornoi design

One of the more powerful design techniques is the pairwise nearest-neighbor or Vornoi clustering algorithm [8]. For a given set of training vectors, we initialize a set of clusters, each cluster containing a single training vector. The methodology is to *merge* vectors into groups until a suitable codebook is formed. The codebook will contain centroids of the clusters of input vectors. The codebook so gotten is provably optimal [13]. To start, one simply computes the distortion between each pair of vectors. Two training vectors with smallest distortion are clustered together, forming a new vector, it being the centroid of the region(in this case a line) defined by the two vectors. The algorithm proceeds by computing the increase in average distortion, for each pair of clusters, that results by replacing the pair of clusters by the corresponding centroid. We continue in this fashion until a suitable size of the codebook is reached.

### 3.7.4 Product codes

This procedure computes the optimal scalar quantizer codebook, and forms the VQ codebook by taking the Cartesian product of the SQ points. In practice, product codes are not optimal.

### 3.7.5 The generalized Lloyd procedure

Equivalent to the k-means statistical clustering technique [19], this algorithm is based on iteratively improving the codebook modification based on Lloyd iteration [13] for scalar quantization. For a given codebook, by the means of a Vornoi condition, we find the optimal partition of the search space into quantization cells. In case of a tie, we assign the input to the cell arbitrarily. Using the centroid condition, we find the optimal codebook for the newly formed cells. It can be shown that for a known, continuous random source, the Lloyd iteration does not result in an increase of the distortion rate. Lloyd algorithm has become more or less the standard in codebook design for a known distribution source.

It is worth noting that a vast array of techniques exist [13] that attempt to modify the Lloyd algorithm, or some of the earlier ones in order to overcome the problem of local minima in distortion rate, to attempt to find global minima and optimize performance. The algorithms are stochastic relaxation, simulated annealing and fuzzy clustering. In our opinion none of these methods are satisfactory. We note that Lloyd algorithm is equivalent to a nonlinear Gauss-Siedel iteration, and in order to generate optimal codes a full nonlinear minimization must be performed. In other words, the centroid and codebook criteria have to be simultaneously satisfied. This work is in progress.

## 4 Forecasting by scalar quantization and entropy coding

Our empirical application uses the standard data set provided by Olsen and Associates. The data include the USD-DM, USD-JPY and the DM-JPY exchange rates. For the purposes of this paper, we sample these data at a tick to tick and 5 minute intervals, and we forecast each of the three random sources independently. Our analysis is conducted on data from October 1, 1992 through August 31, 1993. Although the granularity of encoding of each source is fixed at a tick-to-tick or 5 minute intervals, we present the output data daily. The outputs, per market and per granularity of sampling include the following

- Tree statistics for the compression ratio of the source, giving a measure of average randomness of the binary output of the sources.
- Tree statistics giving the ratio of the number of internal nodes divided by the number of leaves, giving a measure of diversity of binary patterns

output by the source.

- Prediction statistics for the maximum likelihood forecasting strategy. Here we estimate the probability of a "1", updated on a daily basis and forecast with that probability throughout the next day.
- Prediction statistics for the entropy encoding algorithm, growing a single tree for the entire year and calling the next move based on the position in the tree and the probabilities assigned to each child of the current node.

## 4.1 Results

Given the encoding procedure described above, we encoded the binary strings from one year worth of data for all three currencies, at a tick-to-tick and five minute fashion. It is widely accepted that high-frequency foreign exchange markets pass most martingale tests. Hence, there should be no predictability in the binary strings above and beyond the maximum likelihood estimate, in other words, flipping a slightly unfair coin.

Each data-point in the prediction figures represents the cumulative prediction rate at the end of a day (midnight GMT), although the encoding is done for each and every point in the data-set. Such quantizing is done in order to save plotting tens of thousands of points, figures which are informationally equivalent to our plots.

To this end, let us first concentrate on the USD-DM stream. The maximum likelihood ratio yields a roughly 50% probability of a rise in the mean bid/ask spread (figure 10) for the tick-to-tick observations, and five-minute sampling (figure 14). Running the encoder we see that for the tick-to-tick ratio, the compression of the strings is constant at about 1.15:1, as shown in figure 8. Figure 11 shows that the encoder's forecasting is consistently above 64%, or a 12% premium over the MLE estimate, and growing. The encoder's performance grows monotonly towards 66% towards the end of the sample. Figure 34 shows the performance of the MLE estimate compared to the encoder. The difference is significant.

It is interesting to note that, while the compression ratio stays about constant for the duration of the run, the encoder is improving, hinting that a large (but less than exponential) number of states in the transition chain are repetitive and describe high-frequency trading evolution over the period. We infer that high-frequency bid/ask posts are incrementally repetitive, through different days and times of the day.

At a five minute resolution, however, the compression ratio falls monotonly towards 1.04:1 (figure 12), and the encoder's performance (figure 15 degrades to a bit above the MLE estimate (figure 14). The five-minute calls are therefore highly context-free and unpredictable over a coin toss.

For the USD/JPY stream, the compression ratio (figure 16) of the binary output converges to just about 1.15:1 much like the USD/DM. The maximum likelihood ratio is correct about 50.5%(figure 18) while the encoder's performance is steadily improving from 62% to 64% over the lifetime of the sample (figure 19). As with USD/DM stream, the five-minute stream shows no significant context under the encoding.

The interesting case is the JPY/DM stream. The compression ratio is low during the entire simulation at 1.05:1 (figure 24), while the encoding grows steadily to about 53% (figure 27). The five minute encoding shows the same compression at 1.05:1 (figure 28), while the encoder is performing at around 50.5% (figure 30). This poor performance of the encoder for this particular stream is no setback, as we will show, the multitree encoder performs well above 70% for this currency.

The encoder presented above is the simplest on-line predictive strategy for entropy encoding a source in a context sensitive manner. The multitree encoder is one of the encoding strategies we have implemented to further enhance the forecasting power of the encoding strategy.

The multitree encoder encodes the stream in a parallel fashion, by staggering the encoding over several periods and then each encoder is "super-encoded" into a "managing" encoder which predicts the stream as a single encoder. The encoders are started over fixed periods of time, each encoding the source independently. Each encoder casts a vote for the next prediction, which are encoded further by our manager. This manager acts as a simple encoder of the votes. The manager casts his forecast and trades accordingly. This hierarchical encoding is common as performs quite well. Figure 4.1 shows a time series of encoders prediction successes. We see that each encoder starts out at zero and quickly learns. Encoders are also "fired" if their relative compression ratio is low, and new ones are started. Figure 33 shows the prediction rate of the encoders superimposed over the prediction rate of the encoded encoders predictions. We see a dramatic improvement to 70%, over the average performance of a single encoder of about 64%. The procedure is, however, recursive, and a "super" manager can be coded to encode the votes of several managers and so on. These results are left to future work.

Another statistic is of interest. An encoder makes a prediction with a certain probability. A "1" is called if the probability of a "1" is greater than 50%. Figure 35 shows a description of the performance of an encoder versus how sure the encoder is of its own prediction. If an encoder was right about the call, we record is probability. If it was wrong, we add a -1 to the value, thereby separating the rights from the wrongs. We can see that the majority of rights was in the 60% to 80% range. The wrongs are in the 50% to 65%. This data can be used to optimize an encoder call. by not trading unless the prediction is in a certain range. Future work involves constructing a super-encoder to deal with this issue.

The encoders are universal. Therefore, our future work will also include encoding arbitrary alphabets, such as quantized returns by SQ, or quantized returns and side information by VQ.

The hierarchical possibilities are virtually limitless. A real-time on-line trading framework is the final culmination of this effort.

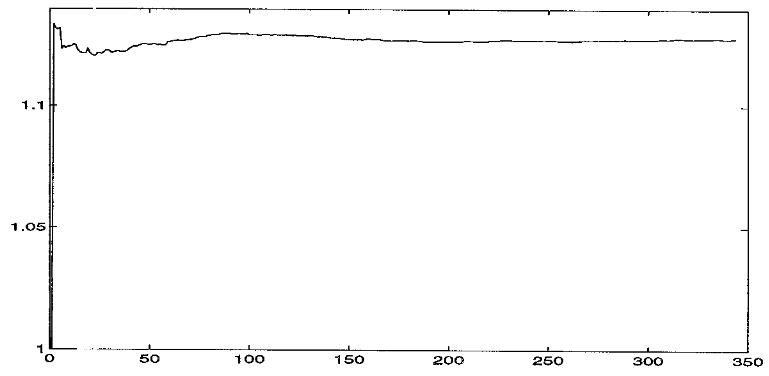

Figure 8: USD-DM binary tick-to-tick compression ratio.

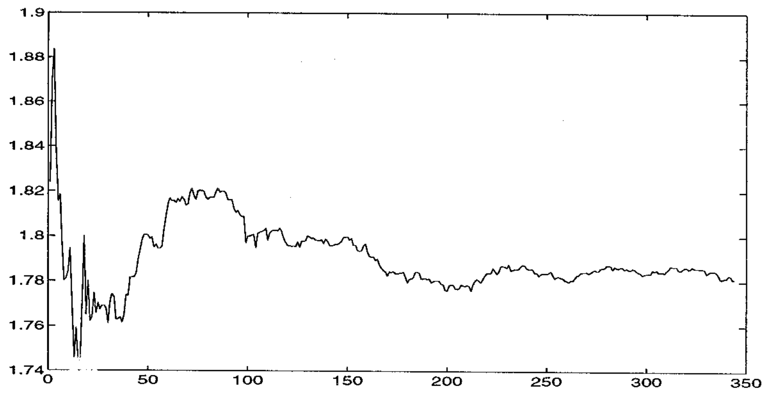

Figure 9: USD-DM tree internal node ratio.

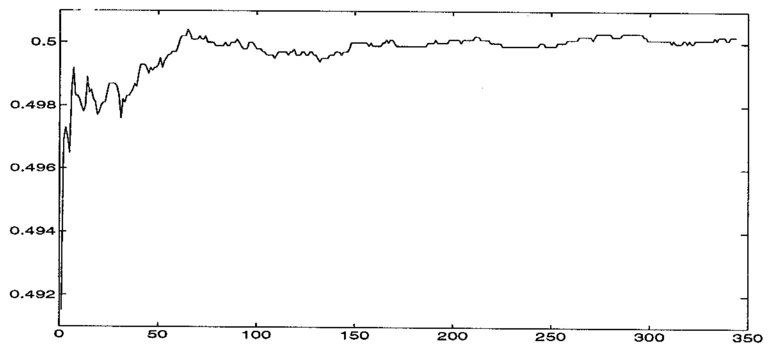

Figure 10: USD-DM Maximum likelihood ratio prediction rate.

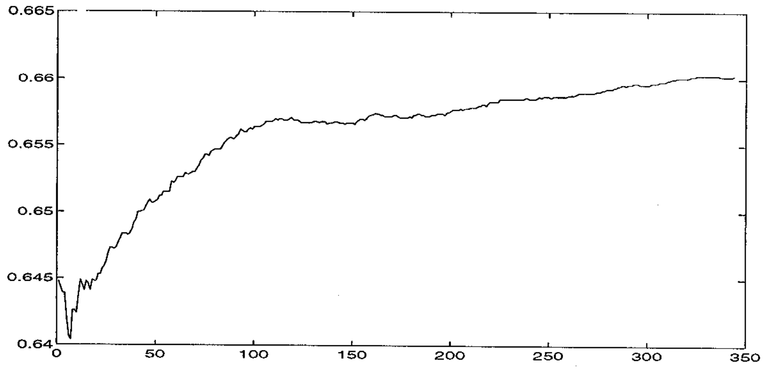

Figure 11: USD-DM Entropy encoding prediction rate.

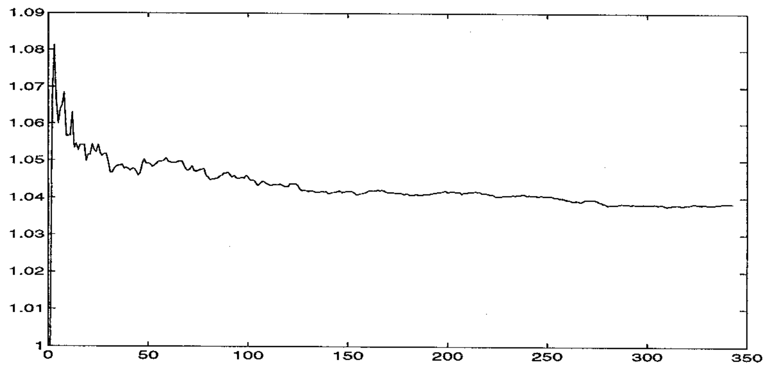

Figure 12: USD-DM binary five minute compression ratio.

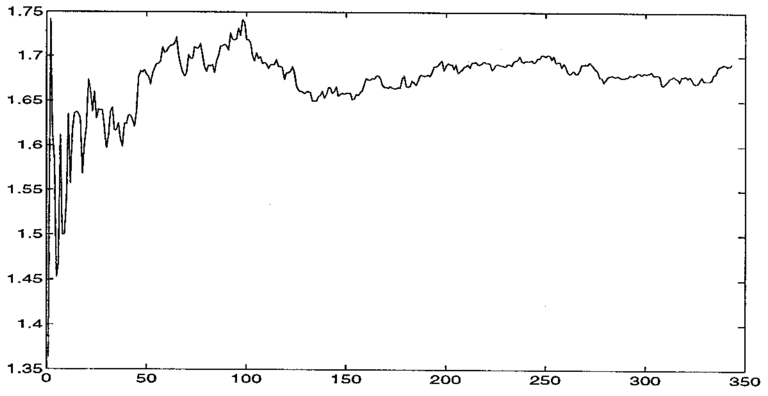

Figure 13: USD-DM five minute tree internal node ratio.

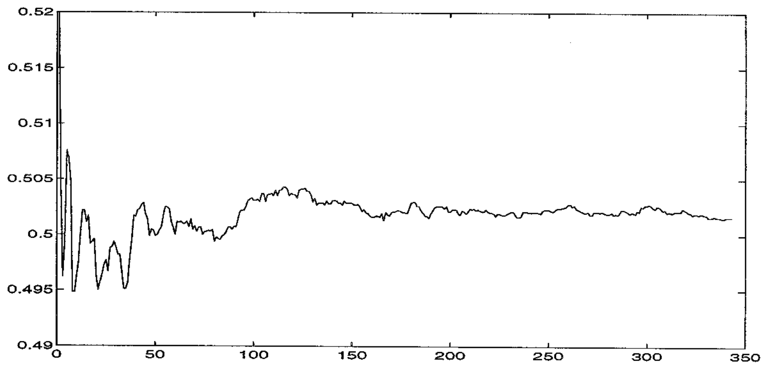

Figure 14: USD-DM five minute maximum likelihood ratio prediction rate.

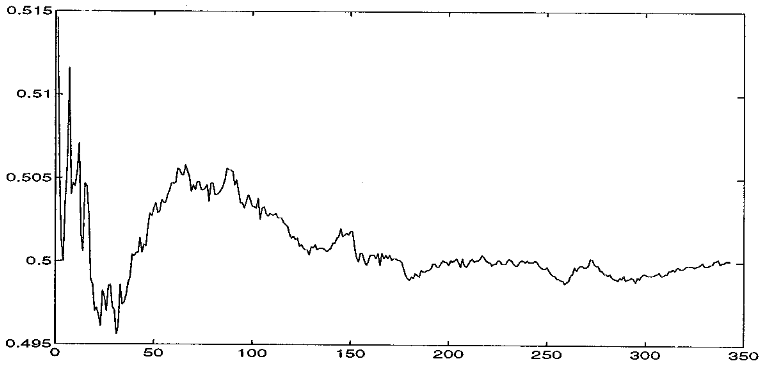

Figure 15: USD-DM five minute entropy encoding prediction rate.

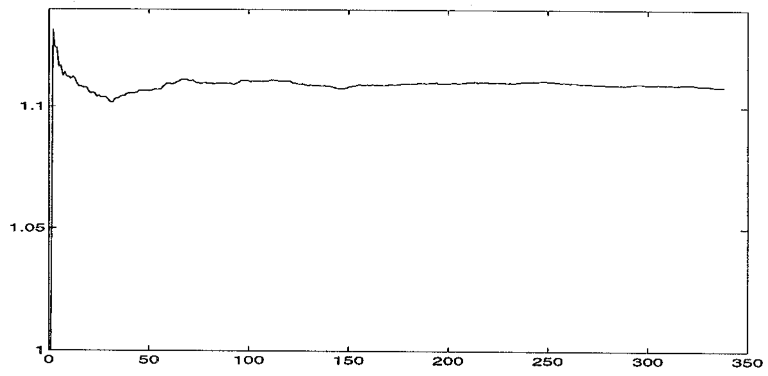

Figure 16: JPN-USD binary tick-to-tick compression ratio.

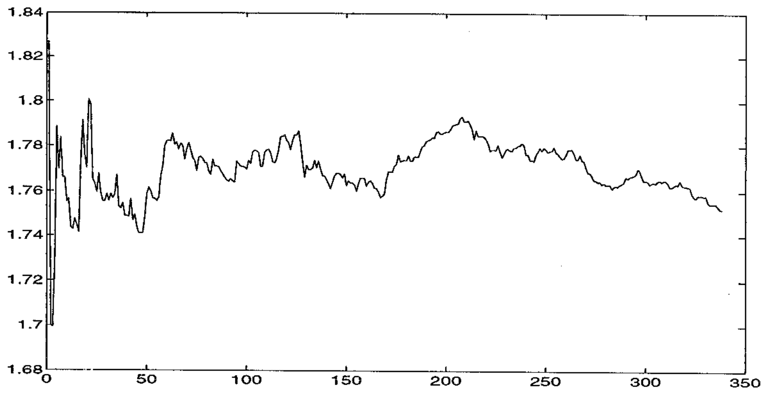

Figure 17: JPN-USD tree internal node ratio.

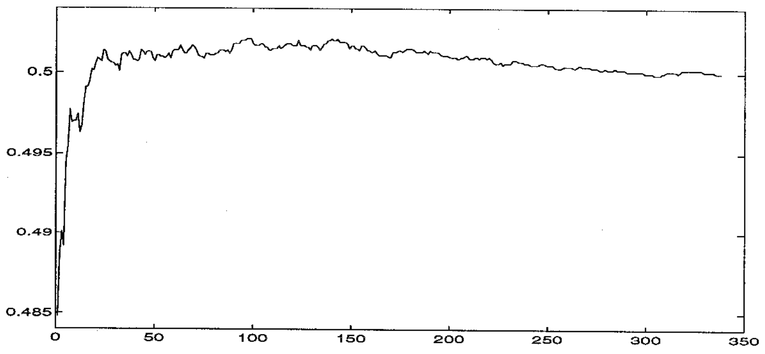

Figure 18: JPN-USD Maximum likelihood ratio prediction rate.

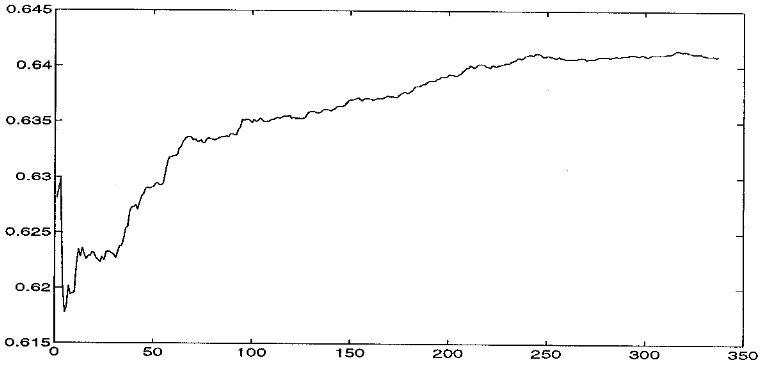

Figure 19: JPN-USD Entropy encoding prediction rate.

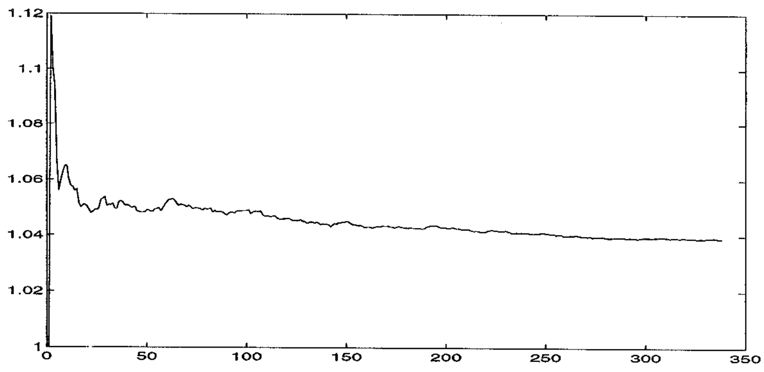

Figure 20: JPN-USD binary five minute compression ratio.

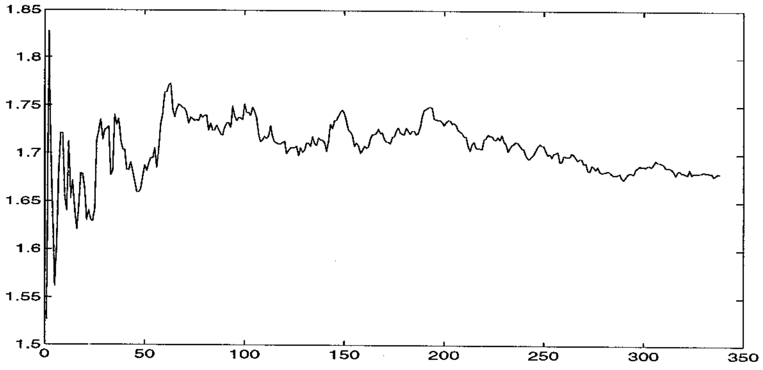

Figure 21: JPN-USD five minute tree internal node ratio.

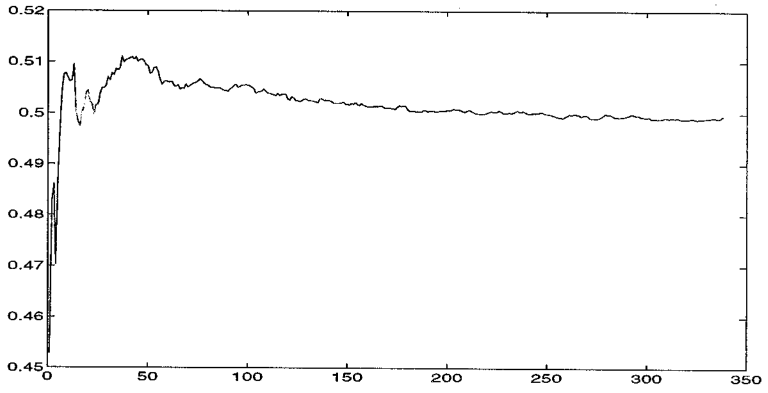

Figure 22: JPN-USD five minute maximum likelihood ratio prediction rate.

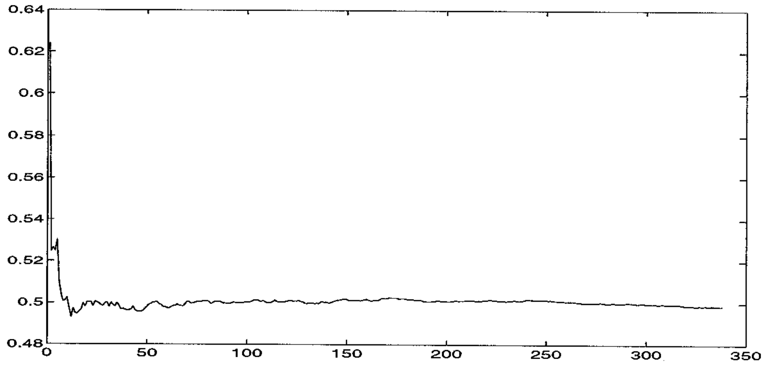

Figure 23: JPN-USD five minute entropy encoding prediction rate.

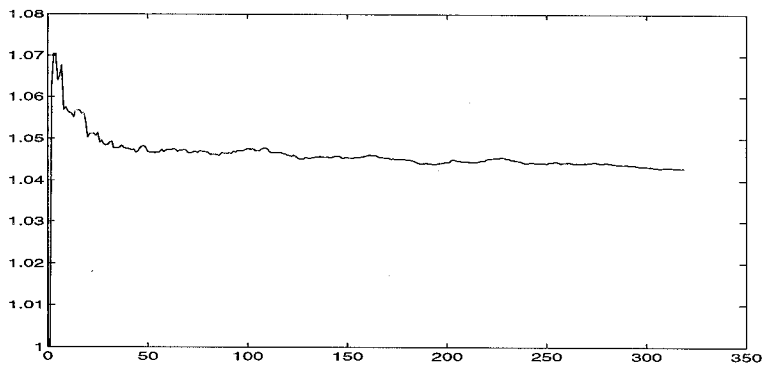

Figure 24: JPN-DM binary tick-to-tick compression ratio.

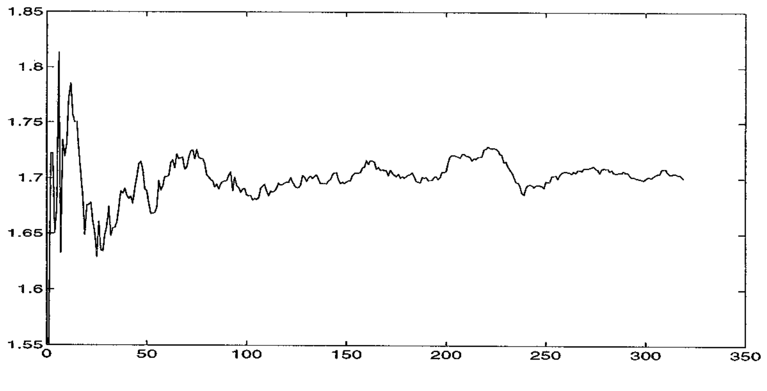

Figure 25: JPN-DM tree internal node ratio.

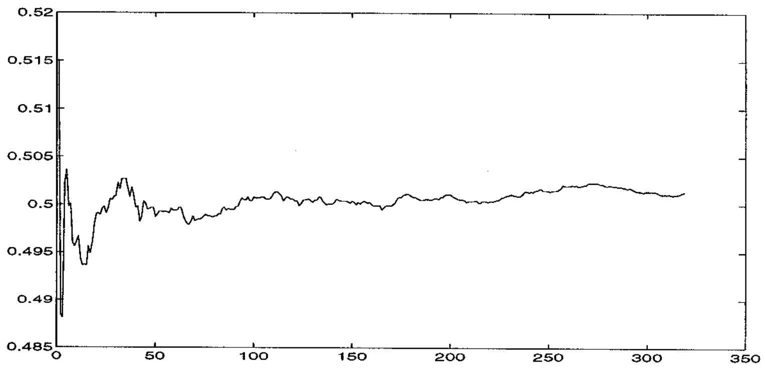

Figure 26: JPN-DM Maximum likelihood ratio prediction rate.

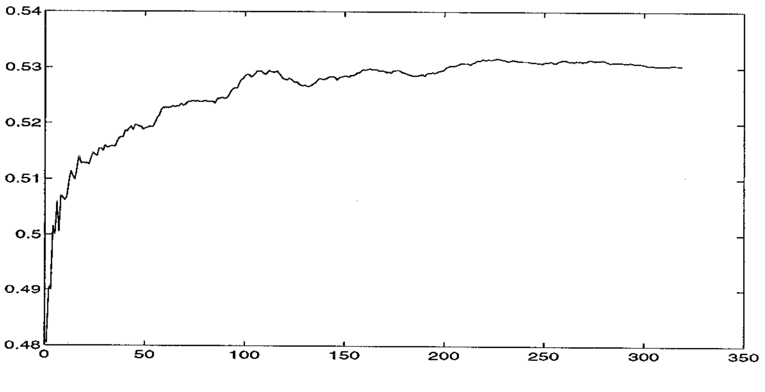

Figure 27: JPN-DM Entropy encoding prediction rate.

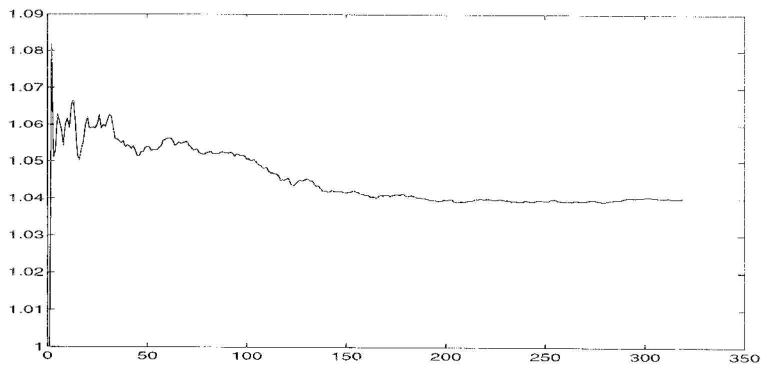

Figure 28: JPN-DM binary five minute compression ratio.

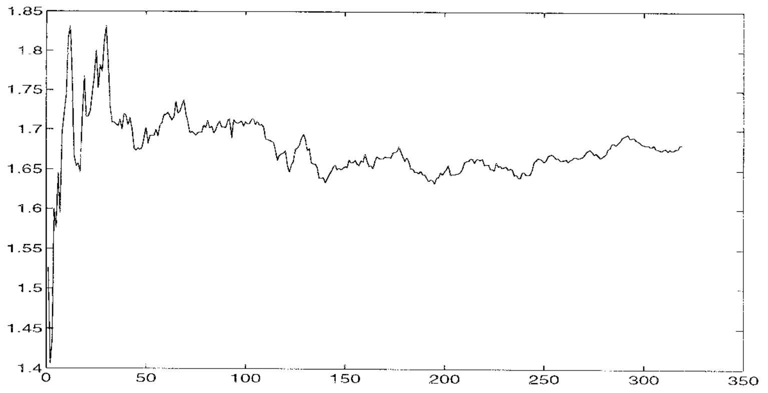

Figure 29: JPN-DM five minute tree internal node ratio.

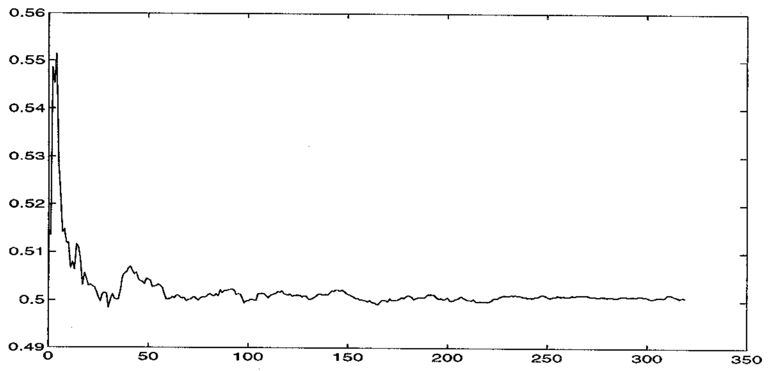

Figure 30: JPN-DM five minute maximum likelihood ratio prediction rate.

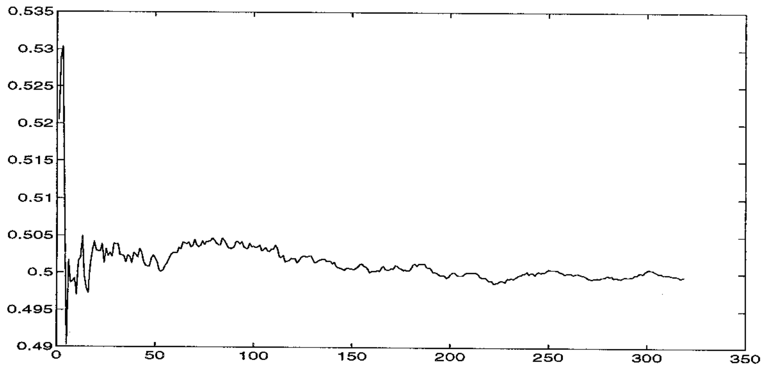

Figure 31: JPN-DM five minute entropy encoding prediction rate.

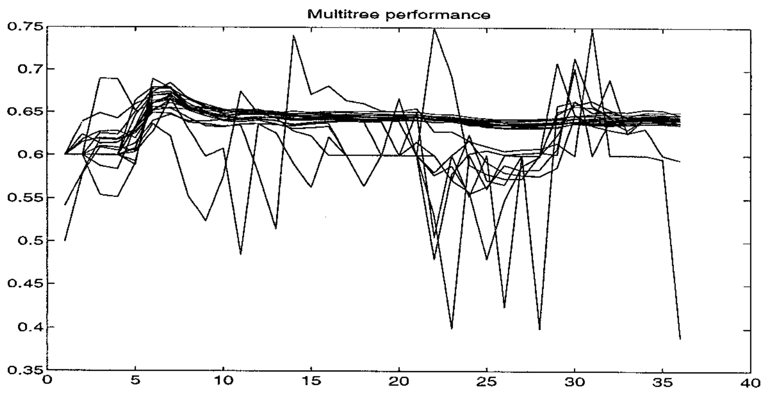

Figure 32: Multitree encoder individual performance.

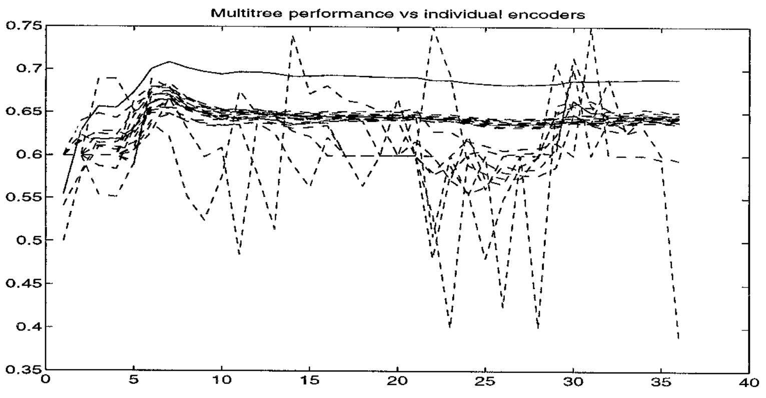

Figure 33: Multitree encoder group performance.

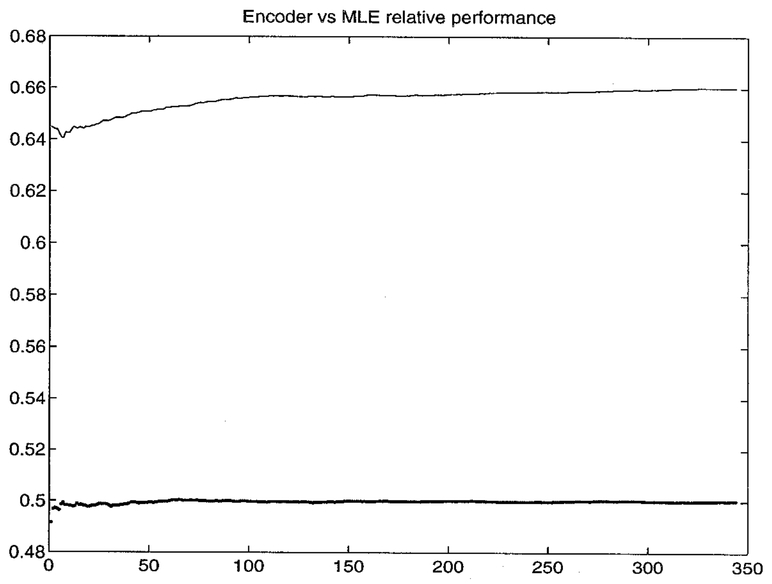

Figure 34: Encoder versus maximum likelihood performance.

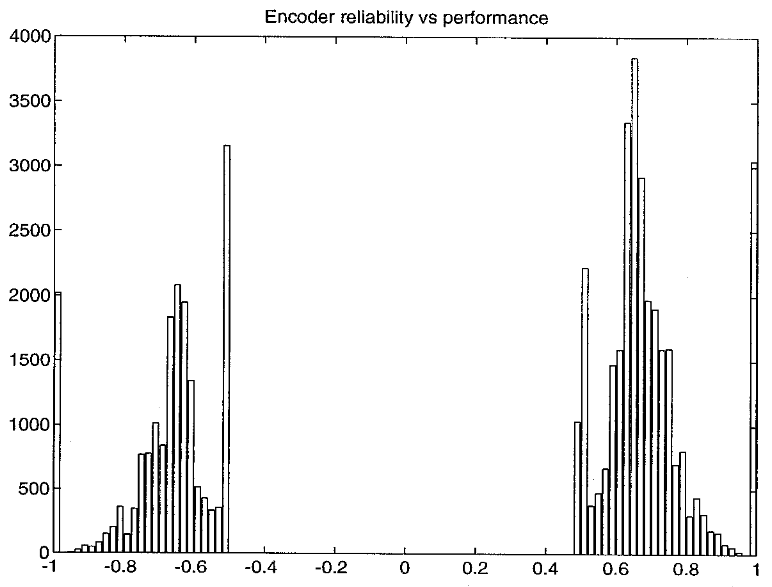

Figure 35: Encoder reliability versus performance.

## References

[1] Gersho A and R. M. Gray. *Vector Quantization and Signal Compression.* Kluwer Academic Publishers, Norwell, Massachusetts, 1992.

[2] L. Breiman. *Probability.* Addison-Wesley Publishing Corporation, Reading, Massachusetts, 1968.

[3] P. C. Chang and R. M. Gray. Gradient algorithms for designing predictive vector quantizers. *IEEE Trans. Accoust. Speech signal Processing*, ASSP-34:679–690, August 1986.

[4] R. Christensen. *Entropy Minimax Sourcebook, Volumes I through IV.* Entropy Limited, Lincoln, Massachusetts, 1980.

[5] T.M. Cover and J.A. Thomas. *Elements of Information Theory.* John Wiley and Sons, New York, NY, 1991.

[6] V. Cuperman and A. Gersho. Vector predictive coding of speech at 16 kb/s. *IEEE trans. Comm.*, COM-33:685–696, 1985.

[7] K.M. Curewitz, P. Krishnan, and J.S. Vitter. Practical prefetching via data compression. *SIGMOD*, 1993.

[8] W.H. Equitz. Fast algorithms for vector quantization picture coding. *IEEE Trans. Acoust. Speech Signal Process*, pages 1568–1575, October 1989.

[9] T. R. Fischer and D. J. Tinnen. Quantized control with differential pulse code modulation. In *Conference Proceedings: 21st Conference on Decision and Control*, pages 1222–1227, December 1982.

[10] T. R. Fischer and D. J. Tinnen. Quantized control using differential encoding. *Optimal Control Applications and Methods*, pages 69–83, 1984.

[11] T. R. Fischer and D. J. Tinnen. Quantized control with data compression constraints. *Optimal Control Applications and Methods*, 5:39–55, January 1984.

[12] A. Glodjo, S. Azhar, G. Badros, M.Y. Kao, and J.H. Reif. Data compression and stock market prediction. In *Proceedings of 4th Annual Data Compression Conference*, 1994.

[13] R. M. Gray. Vector Quantization. *IEEE ASSP Magazine*, 1:4–29, April 1984.

[14] R. M. Gray and H. Abut. Full search and tree searched vector quantization of speech waveforms. *Proc. ICASSP*, pages 593–596, May 1982.

[15] S. Gupta and A. Gersho. Feature predictive vector quantization of multipsectral images. *Trans. on Geoscience & remote sensing*, 30(3):491–501, May 1992.

[16] A. Habibi. Comparison on n-th order dpcm encoder with linear transformations and block quantization techniques. *IEEE Trans. Commun. Technol.*, COM-19:948–956, December 1971.

[17] H. Hang and J. W. Woods. Predictive vector quantization of images. *IEEE Trans. Acous. Speech and Sig. Proc.*, 33, Nov. 1985.

[18] Y. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantizer design. *IEEE Trans. on Commun.*, 28:84–95, Jan. 1980.

[19] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Fifth Berkeley Symposium on Maht. Stat. and Prob.*, volume 1, pages 281–296, 1967.

[20] N.F.G. Martin and J.W. England. *Mathematical Theory of Entropy.* Addison-Wesley Publishing Corporation, Reading, Massachusetts, 1981.

[21] A. Okabe, B. Boots, and K. Sugihara. *Spatial Tessellations.* John Wiley and Sons, New York, New York, 1992.

[22] F.P. Preparata and M.I. Shamos. *Computational Geometry.* Springer-Verlag, New York, New York, 1985.

[23] H.L. Royden. *Real Variables.* MacMillan, New York, New York, 1988.

[24] J. A. Storer. *Data Compression: Methods and Theory.* Computer Science Press, Rockville, MD, 1988.

[25] J.T. Tou and R.C. Gonzales. *Pattern Recognition Principles.* Addison-Wesley Publishing Corporation, Reading, Massachusetts, 1974.

[26] J. S. Vitter and P. Krishnan. Optimal prefetching via data compression. In *Proceedings of 32nd Annual IEEE Symposium on FOCS*, pages 121–130, October 1991.

[27] T. A. Welch. A technique for high-performance data compression. *Commun. of ACM*, pages 8–19, June 1984.

[28] S. Willard. *General Topology.* Addison-Wesley Publishing Corporation, Reading, Massachusetts, 1970.

[29] J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions of Information Theory*, 24:530–536, September 1978.