ABSTRACT

Title of Dissertation: CLASSIFICATION AND COMPRESSION OF MULTI-RESOLUTION VECTORS: A TREE STRUCTURED VECTOR QUANTIZER APPROACH

Sudhir Varma, Doctor of Philosophy, 2002

Dissertation directed by: Professor John S. Baras
Department of Electrical and Computer Engineering

Tree structured classifiers and quantizers have been used with good success for problems ranging from successive refinement coding of speech and images to classification of texture, faces and radar returns. Although these methods have worked well in practice there are few results on the theoretical side.

We present several existing algorithms for tree structured clustering using multi-resolution data and develop some results on their convergence and asymptotic performance.

We show that greedy growing algorithms will result in asymptotic distortion going to zero for the case of quantizers and prove termination in finite time for constraints on the rate. We derive an online algorithm for the minimization of distortion. We also show that a multiscale LVQ algorithm for the design of a tree structured classifier converges to an equilibrium point of a related ordinary differential equation.

Simulation results and description of several applications are used to illustrate the advantages of this approach.

CLASSIFICATION AND COMPRESSION OF MULTI-RESOLUTION VECTORS: A TREE STRUCTURED VECTOR QUANTIZER APPROACH

by

Sudhir Varma

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2002

DEDICATION

To Amma and Achchen

ACKNOWLEDGEMENTS

First and foremost, to my late father Mr. Kerala Varma Thampuran and my mother, Mrs. Manorama Varma for all the trust and encouragement they have given me in this pursuit, for which I am forever indebted.

To my advisor, Dr. John Baras, for his support, help and patience. I would have found it impossible to choose the right direction in research and career without his advice.

To Dr. P. Abshire, Dr. C. Berenstein, Dr. R. Chellappa and Dr. S. Shamma for valuable corrections and criticisms regarding the thesis and for taking time out of their schedules to be in my defence committee

I would like to thank Vijay Bharadwaj for all the interesting conversations and the opportune loans of money. Also my thanks to my office room-mates Zhang Chang, Xiaobo Tan, Maben Rabi and Jia-Shiang Jou for all the discussions and help on research topics and other matters.

To my friends, for giving me a life outside of school.

This work was supported by the following grants for which I am grateful

- ONR contract NOO149710501EE
- DARPA contract DAAD190110494
- Hughes Network Systems and the Maryland Industrial Partnerships Program contract 182918

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

One important problem in communication theory is that of signal coding. Most real-world signals of interest are real-valued scalars or vectors and most communication systems are digital in nature and for such systems mapping a real-valued signal onto a discrete alphabet is necessary before it can be transmitted.

Signal coding lies at the heart of almost all modern data transmission systems. Digital transmission offers a tremendous advantage over analog transmission in noise suppression, signal processing algorithms, error correction schemes and security issues. The easy availability and flexibility of embedded or stand-alone computing power has made digital communication and processing of signals ubiquitous.

Since, in general, a real-valued source cannot be exactly represented by a discrete alphabet, we have the notion of a distortion measure which quantifies the cost incurred by representing a given source with a given alphabet. Conventionally, lesser cost implies better performance and to find the optimal alphabet with the least distortion for a given source is always our aim. In succeeding chapters, we will give our definition of a distortion measure and present algorithms that construct optimal alphabets.

Shannon [42] introduced the idea that quantizing a real valued signal onto a discrete alphabet with the least amount of distortion is possible if this quantization is done on signal blocks, instead of scalar values. Significant improvement in distortion over scalar quantization is possible even if the values of the signal in a block are independent of each other. If the values of the signal at subsequent time points are not independent, we gain even more in performance. Then there are also times where the signal itself is a vector and quantizing each component separately would be inefficient. Such cases happen, for example, in image processing or sensor arrays.

1.1 The quantization problem

As mentioned earlier, the initial applications of vector quantization (VQ) were in the field of quantization and coding of real, vector-valued signals. Here the objective is to find a mapping from the continuous valued vector space $\mathcal{D} \subseteq \mathbb{R}^d$ to a discrete alphabet with k elements. Specifically, we look for a function $Q : \mathcal{D} \mapsto \Theta = \{\theta_1, \theta_2, \ldots, \theta_k\}$, with $\theta_i \in \mathbb{R}^d$. This gives us a partition $\mathcal{V} = \{\mathcal{V}_i\}$ such that $\mathcal{V}_i = \{x \in \mathcal{D} | Q(x) = \theta_i\}$. For any $x \in \mathcal{V}_i$, the vector θ_i, is the codeword (or representation) assigned by $Q(x)$. Our goal is to select a code-book Θ and a function $Q(x)$ such that the average quantization error, $D(\mathcal{V}, \Theta) = E(\rho(Q(x), x))$ measured by a positive, convex function $\rho(Q(x), x)$ is minimized. Some examples of ρ are the mean squared error $||x - \theta_i||^2$, the general r^{th}–norm error $(||x - \theta_i||_r)^r$ and the Itakura-Saito distortion $(x - \theta_i)^T R(x)(x - \theta_i)$ (see appendices of [9]).

One requirement that is not readily apparent from the above description is that the mapping $Q(.)$ must be mathematically tractable in the sense that given any

x, it should not be computationally burdensome to calculate $Q(x)$ nor should the memory requirements for storing this map be excessive. We will see later that the optimal mapping is easy to calculate and the storage requirements are limited to storing the values of a few centroids.

1.2 The classification problem

Mapping a vector-valued signal to a discrete alphabet has applications other than coding. One common problem in signal-processing is classification of data in the form of a vector. Specifically, assume that we have a random ordered pair (X, C_X) where X is i.i.d. and can take values in $\mathcal{D} \subset \mathbb{R}^d$, and $C_X \in \{1, 2\}$ is a class label that can be one of two classes. We have the prior probabilities $\pi_1 = P(c_x = 1)$ and $\pi_2 = P(c_x = 2)$ and probability density functions $p_1(x) = p(x|c_x = 1)$ and $p_2(x) = p(x|c_x = 2)$. We are interested in finding a predictor $\hat{C}(x)$ for the class c_x, given the vector x. $\hat{C}(x)$ must, of course, give a unique prediction for any x. In addition we would like to ensure that the classification error $P\{\hat{C}(X) \neq C_X\}$ is as small as possible.

Some real life examples of this problem include classification of radar returns and face recognition [3], texture classification [32], vowel classification [41] and tool wear analysis [45].

$\hat{C}(x)$ is a mapping from $\mathcal{D}$ onto $\{1, 2\}$ which partitions the space $\mathcal{D}$ into two areas, $\mathcal{V}_1 = \{x : \hat{C}(x) = 1\}$ and $\mathcal{V}_2 = \{x : \hat{C}(x) = 2\}$. The correspondence to the quantization problem described in the previous section is clear. Instead of mapping each x to a representative to minimize the expected distortion $\rho(., .)$ we map it to a class to minimize a classification error.

Finding a $\hat{C}(x)$ that minimizes the classification risk is easy if we know $\pi_1, \pi_2, p_1(x)$

and $p_2(x)$. But, in most cases we only have a training sample in the form of a sequence of observation vectors and corresponding class labels $(x_i, c_i),\ i = 1, 2, \ldots, n$. In Chapter 3 we will present the popular LVQ algorithm for constructing a decision function $\hat{C}(x)$ using training samples.

1.3 Tree structured quantizers and classifiers

Quantization and classification can both be seen as decision-making algorithms in the sense that given a vector x we have to choose one of several codewords or classes to assign to the vector.

Tree Structured Vector Quantizers (TSVQ) offer faster lookup speed compared to other quantizers and classifiers. With k codewords or classes, the TSVQ execution time is $O(\log(k))$ compared to $O(k)$ for an ordinary quantizer. This improvement comes with a decrease in performance, but with a carefully designed TSVQ, this decrease can be minimized.

TSVQ operate on the principle of successive refinement of knowledge. It does a progressive classification where an initial rough classification is done followed by finer and finer partitions that improve the performance.

Multi-Resolution TSVQ (MRTSVQ) is an improvement on TSVQ in cases where we can obtain the observation vector in multiple levels of detail. Coarse representations can be used to do the initial, rough classification and more and more detail can be added for finer discrimination.

1.4 Summary of new work

In this thesis we will present a greedy growing algorithm for constructing tree structured classifiers and quantizers for multi-resolution data. For the quantization case we show that under some conditions the distortion of the tree goes to zero asymptotically as the number of iterations increases without bound.

For any tree structured clustering method, we assign a new vector to its appropriate cluster by starting out with the root as the current node. Then the children of the current node are compared with the given vector to find the closest node. This node now becomes the new current node and the process repeats until we reach a leaf node. The cluster corresponding to this leaf node is assigned as the cluster to which the vector belongs. The expected number of comparisions with child nodes is a measure of the expected computational time it will take to assign a new vector to its cluster. We will call this the *rate of the tree*. It is easy to see (and we will prove it later) that as the tree grows, the rate of the tree increases.

Now assume that we stop the algorithm when the rate of the tree gets larger R. Then we will show that the algorithm stops in finite time.

We also derive the form of the online algorithm for minimization of distortion that was implied in [2] for the case of two centroids and show how it can be extended for the general K-centroid case.

For the classification case we present a multi-scale, multi-level LVQ algorithm that adapts each level of the tree simultaneously to converge to a classifier that seeks to reduce the Bayes risk. We show that the adaptation algorithm behaves like a corresponding ordinary differential equation and converges to the global equilibrium of the ODE.

Finally we present several simulation results that illustrate the implementation of these algorithms and one example of the application of these clustering algorithms to the prediction of tool wear from acoustic emissions. We also give several heuristics for practical implementation of these algorithms.

1.5 Arrangement of thesis

Chapter 2 presents background on the motivation and methodology for use of VQ and TSVQ for compression and classification. We present several well known results on algorithms and achievability of successive refinement codes for vector data.

In Chapter 3, we present our definition of a Multi-Resolution Analysis while Chapter 4 will deal with the MRTSVQ algorithm as it applies to compression and some results on asymptotic distortion, termination and online algorithms.

In Chapter 5 we describe the multi-scale LVQ algorithm for classification using an MRTSVQ and present some results pertaining to the convergence of this algorithm.

We end with simulations and descriptions of several applications of MRTSVQ in Chapter 6.

1.6 Notation

We will use *script* letters ($\mathscr{S}, \mathscr{T}, \ldots$) to denote transforms of signals. *Calligraphic* letters ($\mathcal{V}, \mathcal{S}_i, \ldots$) will be used for sets and the set of real numbers will be denoted by $\mathbb{R}$. Symbols starting with d will usually denote the dimension of vector spaces.

Chapter 2

Compression, Classification and Trees

2.1 VQ for compression

As mentioned earlier, vector quantization was initially proposed for the purpose of quantizing a real-valued signal onto a finite, discrete alphabet. Given a real-valued source, there are an infinite number of alphabets that it can be mapped onto; each such mapping producing a code of some information rate R and suffering some distortion D. Rate-Distortion theory [11] shows that the set of all possible couples (R, D) has a convex hull called the *achievable rate-distortion limit.* Points on the hull correspond to alphabets that have either the minimum rate for a given distortion or the minimum distortion for given rate.

This mapping can be done on a point-to-point basis where the output of the source at each time point is mapped onto a letter; or it can be done in a block-wise fashion where a block of data, k time steps long, is mapped onto a letter. Except for rare cases, the rate-distortion limit is achieved asymptotically as $k \to \infty$. This holds true even if the output of the source is independent from one time-step to the next. As Cover and Thomas put it, “It is simpler to describe an elephant and a chicken with one description than to describe each alone.” [11, page 336]

This also means that if the source is itself multi-variate and outputs a vector valued signal at each time, quantizing the output using a vector quantizer is more efficient than quantizing each component separately.

2.1.1 Problem definition

Consider a signal consisting of a sequence $\{x_i\}, x_i \in \mathbb{R}^d$, with x_i i.i.d. and distributed according to a probability density $p(x)$. We want a mapping $Q(x) : \mathbb{R}^d \mapsto \Theta = \{\theta_1, \theta_2, \ldots \theta_k\}$. Here Θ is a discrete alphabet with k letters $\{\theta_i\}, \theta_i \in \mathbb{R}^d$.

In addition, we have a distortion measure $\rho(\theta, x)$ which quantifies the cost incurred in representing x by θ. The expected value of this distortion measure is denoted as

$$D(Q, P) = E_p(\rho(X, Q(X)))$$

Given a probability density $p(.)$, our goal is to find a quantizer $(\mathcal{V}, \Theta)$ that minimizes the distortion $D(\mathcal{V}, \Theta)$. In the next subsection we will present the *Linde, Buzo and Gray (LBG) algorithm* that iteratively finds a locally optimal quantizer.

2.1.2 Iterative local minimization of distortion

Linde, Buzo and Gray [29] show that the following conditions are necessary for a quantizer to minimize the cost $D(\mathcal{V}, \Theta)$:

1. For a given reproduction alphabet Θ, the partition $\mathcal{V} = \{\mathcal{V}_i\}$, where

$$\mathcal{V}_i = \{x : \rho(\theta, x_i) < \rho(\theta, x_j), j = 1, 2, \ldots, k, j \neq i\} \tag{2.1}$$

   has an expected error that is not greater than any other partition

2. For a given partition $\mathcal{V}$, the reproduction alphabet $\Theta = \{\theta_i\}$, where

$$\theta_i = \arg\min_{\theta \in \mathcal{D}} \int_{\mathcal{V}_i} \rho(\theta, x) p(x) dx \tag{2.2}$$

   gives an expected error that is not greater than any other alphabet. We call θ_i the *centroid* of $\mathcal{V}_i$.

These two conditions, which are a generalization of the Lloyd-Max algorithm [30], [31] for the design of scalar quantizers, can be used in an iterative fashion in the *LBG algorithm* to obtain a partition and an alphabet that is locally minimal for $D(\mathcal{V}, \Theta)$ [29].

Condition (1) is called a *Nearest Neighbor* or *Voronoi* or *Dirichlet* partition. It should be noted that we need to store only the values of the k centroids θ_i to fully characterize this quantizer.

If $\rho(x, y) = ||x-y||^2$ is the squared error distance and we have only two centroids θ_1 and θ_2, then the Nearest Neighbor condition gives a partition $\{\mathcal{V}_1, \mathcal{V}_2\} = \{\mathcal{V} \cap H, \mathcal{V} \cap H^c\}$ where $H = \{x : ||x - \theta_1||^2 < ||x - \theta_2||^2\}$. Thus the partition is achieved by a hyper-plane $||x - \theta_1|| = ||x - \theta_2||$.

2.1.3 Asymptotic behavior for high resolution quantizers

Assume that $\rho(\theta, x) = ||x - \theta||^2$. Then it is easy to show that for a distribution p of the random variable X such that $E\{||X||^2\} < \infty$, the distortion for the optimal quantizer can be made arbitrarily small by choosing a large enough number of centroids k. Indeed, for any $\delta > 0$, choose a ball of radius $r < \infty$ centered on zero such that $E\{||x||^2 : ||x||^2 > r^2\} < \delta^2/2$. The finite variance of X implies that such an r exists. Now divide this ball using a partition composed of planes perpendicular to each axis, separated by a distance of $\delta^2/\sqrt{d}$ where d is the dimension of X. Then $B(0, r)$ is partitioned by a finite set of cuboids of side $\delta^2/\sqrt{d}$. Assign the center point of each cuboid as a centroid to all points lying in the cuboid. Since no point in the ball is farther than $\delta^2/2$ from its centroid, the distortion due to points lying in the ball is less than $\delta^2/2$. For all points lying outside the ball, assign the origin as the centroid. Then the distortion contributed by these points is $E\{||x||^2 : ||x||^2 > r^2\} < \delta^2/2$. Thus the total distortion is less than δ^2. This gives the result.

This result also shows that the minimum distortion with k number of centroids, $D(\mathcal{V}, \Theta) \to 0$ as $k \to \infty$.

2.2 Vector classification

As mentioned in the previous chapter, classification of vectors is a problem that is closely related to quantization. In quantization, we are interested in finding a mapping between a vector $x \in \mathbb{R}^d$ onto a discrete alphabet Θ that minimizes a distortion measure between x and its quantized value. In classification, we assume that X is an observation that depends on some *class* C_X and we need to find a mapping $\hat{C}(X)$ from X onto a discrete alphabet that minimizes a *classification error* between the actual class C_X and the prediction $\hat{C}$.

2.2.1 Bayes risk and Bayes optimal classifier

The most common case is when both C_X and $\hat{C}(X)$ take values in the same alphabet. Then, the *Bayes risk* is one quantitative measure of classification performance [24, Ch. 7.2]. Let us denote $\pi_1 = P\{C_x = 1\}$ and $\pi_2 = P\{C_x = 2\}$ the *a-priori* probabilities of observing a vector of a particular class and $p_1(x)$ and $p_2(x)$ the *a-posteriori* probability densities for vector x given that class 1 or 2 was observed. For the case where we have two classes the signal space is split into two disjoint regions $\mathcal{V}_1 = \{x|\hat{C}(x) = 1\}$ and $\mathcal{V}_2 = \{x|\hat{C}(x) = 2\}$. The Bayes risk is then, the probability of misclassification

$$B(\mathcal{V}_1, \mathcal{V}_2) = \int_{\mathcal{V}_1} \pi_2 p_2(x) dx + \int_{\mathcal{V}_2} \pi_1 p_1(x) dx \tag{2.3}$$

The *Bayes optimal classifier* is defined as a partition that minimizes the cost (2.3). It is easy to see that

$$\begin{aligned} \mathcal{V}_1 &= \{x : \pi_1 p_1(x) \geq \pi_2 p_2(x)\} \\ \mathcal{V}_2 &= \{x : \pi_2 p_2(x) > \pi_1 p_1(x)\} \end{aligned} \tag{2.4}$$

is such a partition.

2.2.2 Learning Vector Quantization (LVQ)

We have already mentioned that the problem with trying to implement the Bayes optimal classifier is that we need to know the *a priori* probabilities π_1 and π_2 and the *a posteriori* probability densities $p_1(x)$ and $p_2(x)$. In almost all cases of interest, all we have available is a set of training vectors along with their classes $\{(x_n, c_n) : n = 1, 2, \ldots, N\}$.

One solution to this problem is *Learning Vector Quantization* (LVQ), first proposed by Kohonen [24]. LVQ is a clustering method for approximating the Bayes regions with the Nearest Neighbor partitions of some set of centroids $\Theta = \{\theta_i\}$ and a distance function $\rho(\theta, x)$. The algorithm starts with an initial set of centroids along with corresponding classes for their partitions and iteratively updates the centroids and their classes according to the training sequence (x_n, c_n).

The update algorithm is as follows

$$\begin{aligned} \theta_i(n+1) &= \theta_i(n) - \alpha(n) \bigtriangledown_\theta \rho(\theta_i, x_n), \quad \text{if } c_n \text{ is equal to class of } \theta_i \\ \theta_i(n+1) &= \theta_i(n) + \alpha(n) \bigtriangledown_\theta \rho(\theta_i, x_n), \quad \text{if } c_n \text{ is not equal to class of } \theta_i \end{aligned} \tag{2.5}$$

for all θ_i in a neighborhood of x_n. Usually, the only θ_i updated in each iteration is the one nearest to x_n. $\alpha(n)$ is a learning parameter that determines how the past observations are weighted with respect to the present. If $\rho(\theta, x) = ||x - \theta||^2$, then $\bigtriangledown_\theta \rho(\theta, x) = 2(\theta - x)$, and it is easy to see that the θ_i is pushed towards or away from x_n, depending on whether the classes are equal or different. After a cycle of updates in this way, the assigned class of all centroids are updated according to the majority class in its partition.

It seems reasonable that if an equilibrium exists for the centroids in the above algorithm, the total “push” on the centroids by vectors belonging to a different class must be exactly balanced by the total “pull” by the vectors of the same class. Making this intuition more precise, [26] shows that under some conditions on the learning parameter $\alpha(n)$, the LVQ algorithm converges to centroids θ_i such that

$$\int_{\mathcal{V}_i} (\pi_1 p_1(x) - \pi_2 p_2(x)) \bigtriangledown_\theta \rho(\theta_i, x) dx = 0, \forall i \tag{2.6}$$

with $\mathcal{V}_i$ the Nearest Neighbor partition for θ_i.

In general, [26] shows that the behavior of the LVQ algorithm approximates that of the ordinary differential equation (ODE)

$$\dot{\theta}_i = \int_{\mathcal{V}_i} (\pi_1 p_1(x) - \pi_2 p_2(x)) \bigtriangledown_\theta \rho(\theta_i, x) dx \text{ for all } \theta_i \text{ of class 1}$$
$$\dot{\theta}_i = \int_{\mathcal{V}_i} (\pi_2 p_2(x) - \pi_1 p_1(x)) \bigtriangledown_\theta \rho(\theta_i, x) dx \text{ for all } \theta_i \text{ of class 2}$$

2.3 Tree Structured Vector Quantizers

One way of getting reduced algorithmic complexity with the same amount of computational resources, with little degradation in performance is a *Tree Structured VQ* (TSVQ). TSVQs operate on the principle of *successive refinement of knowledge* [16]. In the compression/quantization case, this takes the form of a progressive code for x [8]. A coarse partitioning in the upper layers of the tree results in the most significant bits of the code. Further refinement in the lower levels of the tree gives the less significant bits.

In the case of classification, successive refinement means that in the beginning we do a broad classification at a higher level; followed by refining of the classification at the lower levels until we get a sufficiently low level of error. The classifier is in the form of a tree where at each of the nodes a *test* is done that determines which child node it is classified into [8]. In this way a test vector ends up at a leaf, each of which is associated with a class.

2.3.1 TSVQ for compression

Using a tree structured quantizer offers several advantages. In the case where the number of letters in the codebook is very large (see [3] for example), searching through all the centroids of an ordinary, single level quantizer to find the nearest centroid might be computationally expensive. For k centroids, this search takes

$O(k)$ time which might not be feasible in real-time applications. But this process is highly parallelizable and a parallel computation of all the k distances, followed by a parallel search can bring down the time complexity to $O(\log(k))$. But this requires more computing resources. A TSVQ will achieve the same decrease in computational effort without additional processors. In many cases, the degradation in fidelity for the same rate can be minimized.

Another advantage of a TSVQ is that it naturally results in a *successive refinement code.* The structure of the code is such that a few bits give a coarse description of the source and the rest of the bits provide more and more details. This is useful in when multi-media data has to be sent over communication links to users with differing capacity. Using the same codebook, users with high capacity can enjoy high-fidelity audio and video while those with low capacity links can trade off rate for data that lacks detail, but is still intelligible.

The second way in which multi-rate codes can be useful is when a user might want to skim through several images quickly and is not interested in high-resolution, but might want to pick out some of the images to look at in higher detail.

Which one of the codes, high or low rate should we optimize first in the first case? The answer depends on factors like the relative capacities of the two classes of users and how much each is paying for the multi-media service. The answer is easier in the second case; we would want to find the optimal low-rate codebook and fix it, and then try to find the high-rate codebook that is optimal with the given constraint.

Under what conditions is it possible to optimize the codebooks for each rate simultaneously? This question is addressed in the next section.

2.3.2 Achieving successive refinement

Successive refinement is defined as a higher rate (and lower distortion) code that attains the rate-distortion (R-D) limit which is formed by appending bits to a lower rate (and higher distortion) code that itself attains the R-D limit. Suppose that we are given a real-valued random variable X and that we use two encoders to create a low-rate encoded signal X_1 of rate R_1 and expected distortion D_1, and a higher-rate X_2 of rate R_2 and distortion D_2. Then we will say that X_1 and X_2 are successive refinements of X if

1. X_2 can be obtained from X_1 by appending to it $(R_2 - R_1)$ bits on the average.

2. (R_1, D_1) and (R_2, D_2) attain the R-D limit.

Note that the bits that are appended to X_1 to obtain X_2 are derived from X and represent additional information that is not present in X_1.

Equitz and Cover discuss the achievability of a successive refinement code in [16]. They show that for a code to be a successive refinement code, it is necessary and sufficient that the sequence $X_1 \rightarrow X_2 \rightarrow X$ is Markov.

If this condition is not satisfied, the code will not be a successive refinement code. In such a case, Rimoldi [39] characterizes the set of all (R_1, D_1), (R_2, D_2) pairs that are achievable. He also extends this to general L-resolution codes.

Tree structured vector quantizers are a way of creating multi-resolution codebooks that are very efficient, though suboptimal. There are various ways of creating a tree and they are detailed in [8]. Here, we will briefly summarize different approaches and then present a *greedy tree growing algorithm*

2.3.3 Construction of TSVQ

Constructing a TSVQ for compression can be done by either the *top down* or the *bottom up* approach [13]. In the top down approach, we start with a root node that contains the whole vector space. Then this root node is split (and the vector space partitioned) to form children. These children are further split and this process continues until the desired rate is reached. In this approach, we optimize the low-rate codebook first and then find the high-rate codebook that is optimal given the restriction on the low-rate code. These kinds of hierarchical codebooks are useful when most of the traffic is low-rate and high-rate is the exception.

The other approach, bottom up, is the exact opposite. We start with a set of leaf nodes that partition the space and then we recursively merge nodes to form a tree. Here, the performance of the high-rate code is more efficient compared to that of the low-rate code.

[13] also shows an approach where different weights are given to the rate and distortion at different levels and then the algorithm tries to minimize the sum of weighted distortions and rates. This is a generalization of which the top down and bottom up approaches are special cases.

In this thesis, we will be concerned with a kind of top down approach called *greedy growing*. Greedy algorithms are one of the most commonly used methods in hierarchical codebook design. They are simple to implement and provide sufficiently good performance in many cases. The next subsection gives the details of the greedy growing algorithm.

2.3.4 Definitions

Clusters in a space can be regarded as a *partition* of the space. In the rest of this thesis, we will assume that each partition of a set $\mathcal{D} \subseteq \mathbb{R}^d$ is characterized by a set of centroids $\theta_1, \theta_2, \ldots, \theta_k \in \mathbb{R}^d$. The partition itself is created as the union of k *cells*, $\mathcal{V}_i, i = 1, 2, \ldots, k$. Cell $\mathcal{V}_i$ corresponds to centroid θ_i. To derive the cells from the centroids, we need to fix a distance criterion $\rho(x, y)$ between any two points $x, y \in \mathbb{R}^d$. In the remainder of this thesis, we will fix $\rho(x, y)$ as the squared Euclidian distance $\rho(x, y) = ||x - y||^2$.

Given this distance, we derive the cells $\mathcal{V}_i$ as follows

$$\mathcal{V}_i = \{x : \rho(x, \theta_i) < \rho(x, \theta_j), j = 1, 2, \ldots, k, j \neq i\}$$

Note that each cell is associated with a cluster and all points belonging to that cell belongs to the corresponding cluster. Thus, given a new vector x, we assign it the cluster corresponding to the cell it belongs to. This is done by finding the centroids θ_i among $\theta_1, \ldots, \theta_k$ that is closest to it in the distance metric $\rho(x, \theta)$. Denote by θ_x this particular centroid that is closest to x and $\mathcal{V}_x$ the corresponding cell, then x is said to belong to the cluster associated with cell $\mathcal{V}_x$.

For the case of unsupervised clustering, the distance measure also doubles as a distortion measure. $\rho(x, y)$ not only measures the distance between x and y, but also gives the error made when x is represented by y. A valid criterion for choosing one particular partition out of the infinite possibilities is to minimize the expected distortion $E\{\rho(X, \theta_X)\}$. Here the expectation is computed with respect to the probability density $p(x)$ on X. There are other ways of defining criteria for unsupervised clustering [37].

Let $\theta = \theta^* = c(\mathcal{V}, p)$ attain the minimum for $D(\mathcal{V}, \theta) = E\{\rho(X, \theta)|X \in \mathcal{V}\}$ given the probability density p. If $\rho(.,.)$ is the Euclidian distance metric, this point is unique and belongs to the convex hull of $\mathcal{V}$. We call this θ^* as the *generalized centroid* of $\mathcal{V}$ and denote the minimum value by $D^*(\mathcal{V}) = D(\mathcal{V}, \theta^*)$. In what follows we will not explicitly mention the probability density p unless there is any ambiguity.

Now assume that we have two centroids θ_1, θ_2 and correspondingly, two cells $\mathcal{V}_1, \mathcal{V}_2$ such that $\mathcal{V}_1 \cup \mathcal{V}_2 = \mathcal{V}$. The resultant distortion due to this partition of $\mathcal{V}$ is $D(\mathcal{V}_1, \theta_1) + D(\mathcal{V}_2, \theta_2)$. For fixed $\mathcal{V}_1, \mathcal{V}_2$, this quantity will be the minimum and equal to $D^*(\mathcal{V}_1) + D^*(\mathcal{V}_2)$ if θ_1 and θ_2 are equal to the generalized centroids of $\mathcal{V}_1$ and $\mathcal{V}_2$ respectively. We denote the decrease in distortion

$$\triangle D(\mathcal{V}, \mathcal{V}_1, \mathcal{V}_2) = D^*(\mathcal{V}) - (D^*(\mathcal{V}_1) + D^*(\mathcal{V}_2))$$

There will be at-least one partition $\{\mathcal{V}_1^*, \mathcal{V}_2^*\}$ which is optimal in the sense that for any other partition the decrease in distortion is equal to or greater than that achieved by $\{\mathcal{V}_1^*, \mathcal{V}_2^*\}$.

2.3.5 Greedy tree growing

The greedy tree growing algorithm starts out with a root node that is associated with a cell $\mathcal{D} \subseteq \mathbb{R}^d$. Then, recursively, all leaf-nodes are examined to find the one that will give the biggest reduction in distortion when split. This incrementally optimal node is split and the process is repeated until a desired rate is reached.

We have a splitting algorithm ψ that splits the cell corresponding to any given node on the tree. For a node on the tree and the corresponding cell $\mathcal{V}$, ψ will compute a partition $\{\mathcal{V}_1, \mathcal{V}_2\}$ of $\mathcal{V}$ that gives a decrease in distortion $\triangle D(\mathcal{V}, \mathcal{V}_1, \mathcal{V}_2)$ which is close to the optimal.

Given this the algorithm is as follows:

1. Fix a splitting algorithm ψ

2. Initialize the tree, $T = c(\mathcal{D}, p)$

3. For any leaf node $\tilde{t}_j$ belonging to the set of leaf nodes $\tilde{T}$ of the tree with corresponding cell $\mathcal{U}$, use ψ to find a partition $\{\mathcal{U}_1, \mathcal{U}_2\}$.

4. Compute $\triangle D(\mathcal{U}, \mathcal{U}_1, \mathcal{U}_2)$

5. Find the leaf node $\tilde{t}_{j^*}$ with the maximum value for $\triangle D(\mathcal{U}, \mathcal{U}_1, \mathcal{U}_2)$

6. Implement the split for $\tilde{t}_{j^*}$

7. If stopping criterion is reached, stop. Else, repeat 3-6

The stopping criterion can be of several types. Most common ones are constraint on rate, number of leaf nodes, maximum value of distortion, number of iterations and so on.

In [34] it is shown that if the growth of the tree is continued indefinitely, the distortion of the tree goes to zero while [35] gives conditions under which the algorithm for a tree grown with a stopping criterion on rate eventually terminates.

2.3.6 Tree structured classifiers

Tree structured classifiers are similar to the "if-then-else" rules in case-based reasoning. We have an observation vector consisting of features that are indicative of the class of a source. These features can be real-valued, discrete or categorical. Classification takes place in a step by step manner where at each node in the tree one or a combination of more than one features are used to classify the observation vector into one of the child nodes. This is repeated until a leaf node is reached and a class label is assigned to the observation depending on the node.

Another way of interpreting tree structured classifiers is that each node divides the input vector space into as many disjoint regions as it has child nodes. Thus the input space is hierarchically partitioned into disjoint sets; each set has a class label associated with it and any observation vector falling in it is classified with that label.

Construction of tree structured classifiers can be done in the greedy fashion that we presented for tree structured quantizers. We start with a root node that contains all of the input space. Then, recursively, each leaf node is examined to find the one that gives the biggest decrease in classification cost if split. This optimal node is then split to produce the improved tree. This process is repeated until the stopping criterion is satisfied.

Some classification costs that have been used for comparing classifiers are the Bayes risk, the entropy, the Gini index and the generalized Neyman-Pearson cost which is a weighting of the probabilities of false positives and false negatives [9],[26]. Some common stopping criterion are expected depth of search and average classification error.

In [8], the authors present several algorithms for classification of data that are either numerical or categorical. They also tackle problems where the dimensionality of the observation vectors is not fixed.

The problem with creating classifiers by successive partitioning is that at each node we have to find a split that is optimal in reducing the classification error. With N elements in a node there are a possible 2^N possible partitions. Searching through all possible partitions becomes impossible even for small amounts of data. In [9], the authors derive optimal partitioning rules for creating tree structured classifiers. They show that under a wide variety of conditions, it is possible to find the optimal partition in $O(N)$ time. For any "impurity" measure for the node, they show that minimization of the impurity is equivalent to the nearest neighbor condition for a specific distance.

Chapter 3

Signal space and multi-resolution analysis

A multi-resolution analysis (MRA) is a way of obtaining representations of a signal in increasing levels of resolution. MRAs can be obtained in various ways. A truncated Fourier series is one simple example. *Affine wavelet transforms* and *wavelet packets* offer powerful and flexible procedures for obtaining MRAs. An interesting circumstance where signals are analyzed in multiple resolutions is in biological systems ([1], [47]), as mentioned before. Before we give a definition of an MRA and provide examples, we must define the space of signals that we are considering.

3.1 Signal space

Consider functions $x(t) : [0, 1) \to \mathbb{R}$ defined on the interval $\tau = [0, 1)$ with a norm defined as

$$||x||^2 = \int_0^1 x^2(t)\, dt$$

We will be dealing with the space of all functions of finite norm

$$\mathcal{L}_2(\tau) = \{x(t), t \in [0,1) : ||x||^2 = \int_0^1 x^2(t)dt < \infty\}$$

Since our distance metric between two elements of $\mathcal{L}_2(\tau)$ is the norm of the difference $||x_1 - x_2||^2$, we will actually be working in the equivalence class of elements that are equal in the mean square norm sense. This means that we will not discriminate between two elements x_1 and x_2 if $||x_1 - x_2||^2 = 0$. For example, elements that are equal almost everywhere, but not point-wise equal, will fall into the same equivalence class.

3.2 Definition of MRA

Given this signal space, we define an MRA as a sequence of sets $\mathcal{S}_i \subseteq \mathcal{L}_2(\tau)$ such that

1. $\mathcal{S}_0 \subseteq \mathcal{S}_2 \subseteq \mathcal{S}_3 \subseteq \dots$
2. $\bigcup_{i=0}^{\infty} \mathcal{S}_i = \mathcal{L}_2(\tau)$
3. $x(t) \equiv 0 \in \mathcal{S}_0$
4. Each $\mathcal{S}_i$ is spanned by a set of d_i basis functions $\phi_k^i(t), k = 1, 2, \dots, d_i$ such that any $x(t) \in \mathcal{S}_i$ can be written as

   $$x(t) = \sum_{k=1}^{d_i} c_k(x)\phi_k^i(t)$$

   for a unique set of weights $c_k(x)$.

For any $x(t) \in \mathcal{L}_2(\tau)$ we denote the projection onto the space $\mathcal{S}_i$ by $\mathscr{S}_i x(t) \in \mathcal{S}_i$ where

$$\mathscr{S}_i x(t) = \arg \min_{y(t) \in \mathcal{S}_i} ||x(t) - y(t)||^2$$

Since $\mathscr{S}_i x(t) \in \mathcal{S}_i$, we can find weights $c_k(x)$ such that

$$\mathscr{S}_i x(t) = \sum_{k=1}^{d_i} c_k(x)\phi_k^i$$

Note that if $x(t) \in \mathcal{S}_i$ then $\mathscr{S}_i x(t) = x(t)$ so the notation for the weights $c_k(x)$ is consistent.

In what follows, we will assume that x and ϕ_k^i are functions of $t \in \mathcal{L}_2(\tau)$ and not make it explicit. If we denote $\overline{c}(x) = [c_1(x), c_2(x), \ldots, c_{d_i}(x)]^T$ and $\Phi^i = [\phi_1^i, \ldots, \phi_{d_i}^i]^T$ we can write the above as

$$\mathscr{S}_i x = \overline{c}^T(x)\Phi^i$$

Note that $\mathscr{S}_i x$ is a linear projection and Φ^i is not necessarily an orthogonal basis.

For a given MRA, denote by $\mathcal{C}_i \subseteq \mathbb{R}^{d_i}$ the set of all possible weight vectors; i.e. $\mathcal{C}_i = \{\overline{c}(x) : x \in \mathcal{S}_i\}$. Then $\mathscr{S}_i$ is a one-to-one mapping from the set of functions $\mathcal{S}_i$ to the set of weights $\mathcal{C}_i$. For the norm $||x||$ for any $x \in \mathcal{S}_i$ we can write

$$||x||^2 = \overline{c}^T(x)R_i\overline{c}(x)$$

where R_i is a $d_i \times d_i$ matrix whose $(m, n)^{th}$ element is given by

$$R_i^{m,n} = \int_0^1 \phi_m^i(t)\phi_n^i(t)\,dt = \langle \phi_m^i, \phi_n^i \rangle$$

Since R_i is a symmetric, positive-definite matrix we can find a non-singular matrix W_i such that $R_i = W_i^T W_i$. This implies that

$$||x||^2 = ||W_i\overline{c}(x)||^2$$

where the latter norm is the ordinary squared norm in a d_i dimensional space. Similarly, the distance between the projections onto $\mathcal{S}_i$ of any two elements $x_1(t)$

and $x_2(t)$ belonging to $\mathcal{L}_2(\tau)$ can be computed as $||\mathscr{S}_i x_1 - \mathscr{S}_i x_2||^2 = ||W_i \overline{c}(x_1) - W_i \overline{c}(x_2)||^2$ We will denote

$$\hat{x}_i = W_i \overline{c}(x)$$

Thus for all $x \in \mathcal{S}_i$, $||x||^2 = ||\hat{x}_i||_2^2$.

3.3 Wavelets

Since Fourier first showed that signals could be decomposed into projections along orthogonal basis functions, harmonic analysis has come a long way. Windowed Fourier transforms were studied by Gabor [20] while the first wavelet finds mention in the thesis of Haar [22]. Mallat discovered several relationships between quadrature mirror filters, pyramid algorithms and orthonormal wavelet bases. Meyer [33] constructed the first continuously differentiable wavelets while Daubechies [12] constructed a class of wavelets with compact support with arbitrarily high regularity.

In wavelet analysis, a signal $x(t)$ at resolution i is decomposed into a weighted sum of a series of functions $\phi_k^i(t)$

$$x(t) = \sum_{k=-\infty}^{\infty} c_{i,k} \phi_k^i(t)$$

The most important fact is that for all i, k, $\phi_k^i(t)$ is derived from the same *scaling function* $\phi(t)$ by dilation-s and translations.

$$\phi_k^i(t) = \phi(2^i t - k)$$

$\phi_k^i(t)$ might not be orthogonal across scale i or translation k. We will discuss the cases of orthogonal, bi-orthogonal and semi-orthogonal basis functions later in this section.

The set of all basis functions ϕ_k^i, $i = \ldots, -2, -1, 0, 1, 2, \ldots$ for a given i span a space denoted by $\mathcal{S}_i$. We have

$$\ldots \subseteq \mathcal{S}_{-2} \subseteq \mathcal{S}_{-1} \subseteq \mathcal{S}_0 \subseteq \mathcal{S}_1 \subseteq \ldots$$

The fact that $\mathcal{S}_i \subseteq \mathcal{S}_{i+1}$ implies that there exists weights a_j such that

$$\phi_k^i(t) = \sum_{j=-\infty}^{\infty} a_j \phi_j^{i+1}(t) \tag{3.1}$$

This is true of any MRA (not necessarily wavelet generated) and gives a recursion from which it is frequently possible to reconstruct $\phi(t)$, for a given set of coefficients a_j.

Denote by $\mathcal{W}_i$ the space of all signals that complement $\mathcal{S}_i$ to make up $\mathcal{S}_{i+1}$ so that

$$\mathcal{S}_i \oplus \mathcal{W}_i = \mathcal{S}_{i+1}$$

where for any two sets $\mathcal{U}, \mathcal{W}$, we denote $\mathcal{U} \oplus \mathcal{V} = \{u + v : u \in \mathcal{U}, v \in \mathcal{V}\}$. In addition to the scaling function $\phi(t)$ we also have a function $\psi(t)$ called the *wavelet* whose dilations and translations form a basis on the space $\mathcal{W}_i$. Since any function belonging to $\mathcal{W}_i$ also belongs to $\mathcal{S}_{i+1}$, we have weights w_j such that

$$\psi_k^i(t) = \sum_{j=-\infty}^{\infty} w_j \phi_j^{i+1}(t) \tag{3.2}$$

Note the similarity to 3.1.

Much work has been done on constructing wavelet bases that have orthogonality, compact support and symmetry. Orthogonality makes it easy to decompose a signal into a weighted sum of basis functions. Compact support is desirable because it results in finite impulse response (FIR) filters for the decomposition rather than infinite impulse response (IIR) filters. FIR filters are easier to implement using common signal processing circuits than IIR filters. Finally, symmetry of the basis functions results in linear phase filters that are necessary to avoid undesirable phase distortions.

Unfortunately, these three attributes are mutually antagonistic. Compactly supported, orthogonal wavelet functions are not symmetric. Bi-orthogonal and semi-orthogonal wavelets are a compromise that offer compactness and symmetry in exchange for some complexity in the decomposition procedure.

3.3.1 Semi-orthogonal wavelets

Semi-orthogonal wavelets differ from orthogonal wavelets in that there are two scaling functions ϕ and $\tilde{phi}$. ϕ_j^i is not orthogonal across translation j but we have

$$\langle \phi_k^i, \tilde{\phi}_j^i \rangle = \delta(j,k)$$

i.e. $\tilde{\phi}_j^i$ is orthogonal to all $\phi_k^i, k \neq j$. Thus it is possible to calculate the coefficients for the decomposition

$$x(t) = \sum_{k=-\infty}^{\infty} c_{i,k} \phi_k^i(t)$$

for $x \in \mathcal{S}_i$ as

$$c_{i,k} = \langle x, \tilde{\phi}_k^i \rangle$$

Another property of semi-orthogonal wavelets is that the space spanned by $\tilde{\phi}_j^i, j = 1, 2, \ldots$ is the same as the space spanned by $\phi_j^i, j = 1, 2, \ldots$ i.e. $\mathcal{S}_i$. $\tilde{\phi}_j^i$ is called the *dual scaling function.* We also have a dual wavelet $\tilde{\psi}_j^i$ such that

$$\langle \psi_k^i, \tilde{\psi}_j^i \rangle = \delta(j,k)$$

which spans the space $\mathcal{W}_i$. ψ_j^i and $\tilde{\psi}_j^i$ are called the *synthesis* and *analysis* wavelet respectively.

Semi-orthogonal MRAs offer compactly supported, symmetric synthesis wavelets but the analysis wavelets are not compactly supported.

3.3.2 Bi-orthogonal wavelets

Like semi-orthogonal MRAs, bi-orthogonal wavelets have dual wavelets and scaling functions. But the difference here is that the dual scaling functions span a subspace $\tilde{\mathcal{S}}_i \neq \mathcal{S}_i$ and the dual wavelets span a subspace $\tilde{\mathcal{W}}_i \neq \mathcal{W}$. Furthermore we do not have $\mathcal{S}_i$ orthogonal to $\mathcal{W}_i$ as we had in the case of orthogonal and semi-orthogonal MRAs.

Bi-orthogonal MRAs offer compactly supported, symmetric analysis and synthesis wavelets and scaling functions; something that neither orthogonal nor semi-orthogonal wavelets can provide.

For more details on the construction and use of orthogonal, semi-orthogonal and bi-orthogonal wavelets see [21].

3.4 Biological filters

Hierarchical processing of signals have been observed in the primary auditory cortex (A1) of the mammalian brain [47]. The auditory cortex receives the input from the inner ear which computes a spectrogram of the sound that impinges on the ear.

In the A1, the neurons are arranged in a 2D map. Neurons are arranged in order of selectivity to increasing frequencies along one axis of the 2D map. This is the so called *tonotopic axis.* Thus sounds of a particular frequency will excite neurons around a particular region on the tonotopic axis.

Much research has gone into determining what features are presented along the other axis of this 2D map. Researchers have identified three characteristics that vary along the second axis. They are

1. symmetry of the spectrogram,

2. bandwidth of the spectrogram and

3. direction of FM sweep.

Along the second axis, neurons exhibit a continuous gradation in the kind of symmetry they are attuned to. Starting with neurons that are selective to spectrograms with higher energy in frequencies above the Base Frequency (BF), the selectivity grades to neurons that are selective to symmetric spectrograms up to neurons selective to spectrograms with higher energy in frequencies below BF (See Fig. 3.1).

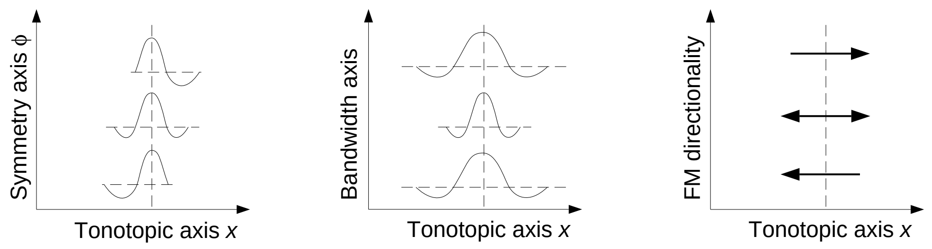


Figure 3.1: Features arranged along the second axis in A1

The bandwidth of the spectra that the neurons are selective to, also changes from narrow bandwidth in the center of the axis to broad bandwidths at the ends.

Thirdly, neurons at one end are selectively attuned to chirps with downward moving frequency and neurons at the other end are attuned to upward moving FM chirps, while neurons in the middle are equally responsive to chirps in both directions.

Wang and Shamma [47] propose a multi-resolution signal processing scheme to account for these observations. A seed function $h(x)$ is used to model the sensitivity profile of a neuron. Given any function $h(x)$, we can find symmetric and anti-symmetric functions h_e and h_o such that

$$h = h_e + h_o$$

The unique h_e and h_o are given by

$$\begin{aligned} h_e(x) &= \frac{h(x) + h(-x)}{2} \\ h_o(x) &= \frac{h(x) - h(-x)}{2} \end{aligned}$$

Then we can produce a function

$$w_s(x; \phi_c) = h_e(x) \cos \phi_c - h_o(x) \sin \phi_c$$

that continuously grades from antisymmetric in one direction to symmetric to antisymmetric in another direction when ϕ_c goes from $-\pi/2$ to $\pi/2$. This form for w_s was chosen so that the magnitude of the Fourier transform of $w_s(x : \phi_c)$ is a constant independent of ϕ_c. The authors also show that this is effective in accounting for FM selectivity.

This takes care of the variation in symmetry and FM selectivity of the response. To model the variation in bandwidth, $h(x)$ is dilated according to $h_s(x) = h(\alpha^s x)$ for a fixed parameter α (Usually $\alpha = 2$). This gives us two variables; ϕ_c which models the symmetry and s which models the scale. Thus, in this model of the primary auditory cortex, the spectrum of the input sound is analyzed along three axes, the center frequency x, the symmetry ϕ_c and scale (or bandwidth) s. Fig 3.2 shows this pictorially.

In Chapter 6 we will present an application of this model when we use it to extract multi-resolution features that will help us estimate the wear on a milling tool from its acoustic emissions.

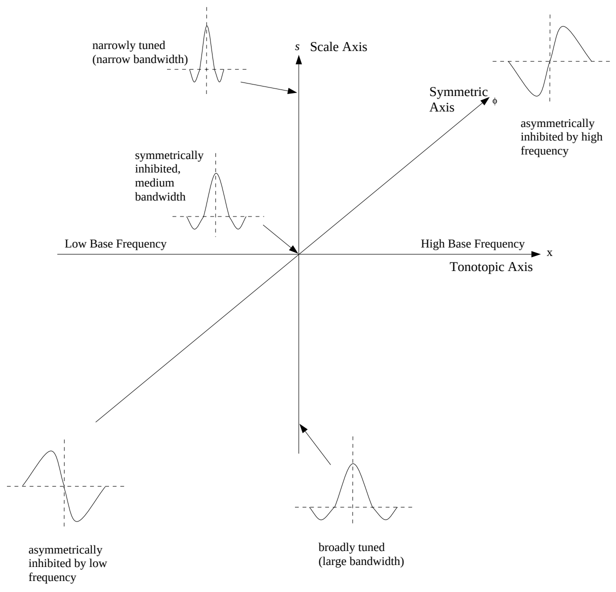


Figure 3.2: Axes of analysis of spectrum in A1

3.5 Probability densities on $\mathcal{L}_2(\tau)$

We assume that we have a probability density P on $\mathcal{L}_2(\tau)$. One interpretation of a probability density on an infinite dimensional space is that the signals $x(t)$ are the outputs of a stochastic process. Given this density, we can also find the density in the d_i dimensional space of i^{th} resolution representations of all $x \in \mathcal{L}_2(\tau)$, i.e. $\mathcal{S}_i$.

Another assumption for the remainder of this thesis is that $M_\infty(P) = E_P(||x||^2) < \infty$. In other words, the expected power of the signal must be finite. The fact that $\mathscr{S}_i x$ is the projection of x onto the space $\mathcal{S}_i$ implies that $\mathscr{S}_i x$ is orthogonal to the error $(\mathscr{S}_i x - x)$ which gives

$$\begin{aligned}
M_i(P) &= E\{||\mathscr{S}_i x||^2\} \\
&\leq E\{||x - \mathscr{S}_i x||^2\} + E\{||\mathscr{S}_i x||^2\} \\
&= E\{||x - \mathscr{S}_i x||^2\} + E\{||\mathscr{S}_i x||^2\} + \langle (x - \mathscr{S}_i x), \mathscr{S}_i x \rangle \\
&= E\{||x - \mathscr{S}_i x + \mathscr{S}_i x||^2\} \\
&= E\{||x||^2\} = M_\infty(P) < \infty
\end{aligned}$$

$M_i(P) < \infty$ for all i means that for any $\mathcal{V} \subset \mathcal{S}_i$, the problem of finding the centroid that minimizes the distortion is well defined and has at least one solution.

The actual multi-resolution analysis done on the signals plays a very important role in good classifier and quantizer performance. It must be capable of picking out *features* that are most significant to the performance, in the coarse level. Some algorithms for selecting an MRA for a specific problem are given in [10], [38] and [49].

3.6 Splitting to reduce distortion

Given a probability density p on $\mathcal{L}_2(\tau)$, we can calculate the distortion due to all the points in a cell $\mathcal{V} \subseteq \mathcal{L}_2(\tau)$ represented by point $\theta \in \mathcal{V}$ as

$$D(\mathcal{V}, \theta) = \int_{\mathcal{V}} ||x - \theta||^2 dP \tag{3.3}$$

If $\theta \in \mathcal{S}_k$, i.e. θ can be expressed as a linear combination of the d_k basis functions of $\mathcal{S}_k$, then we can express 3.3 as

$$\begin{aligned} D(\mathcal{V},\theta) &= \int_{\mathcal{V}} ||x - \mathscr{S}_k x + \mathscr{S}_k x - \theta||^2 dP \\ &= \int_{\mathcal{V}} ||x - \mathscr{S}_k x||^2 + ||\mathscr{S}_k x - \theta||^2 - 2\langle (x - \mathscr{S}_k x), (\mathscr{S}_k x - \theta)\rangle dP \\ &= \int_{\mathcal{V}} ||x - \mathscr{S}_k x||^2 dP + \int_{\mathcal{V}} ||\mathscr{S}_k x - \theta||^2 dP \qquad (3.4) \end{aligned}$$

The last equality follows since $(\mathscr{S}_k x - \theta)$ is a linear combination of the orthonormal basis functions of S_k to which $(x - \mathscr{S}_k x)$ is orthogonal.

Denote $D_k(\mathcal{V},\theta) = \int_{\mathcal{V}} ||\mathscr{S}_k x - \theta||^2 dP$ and $\hat{D}_k(\mathcal{V}) = \int_{\mathcal{V}} ||\mathscr{S}_k x - x||^2 dP = E_p\{||\mathscr{S}_k X - X||^2\}$. $D_k(\mathcal{V},\theta)$ is the component of the distortion that changes with θ. $\hat{D}_k(\mathcal{V})$ is the orthogonal component of the distortion that is independent of θ and depends only on the resolution k. For any $\mathcal{V}$, $\hat{D}_k(\mathcal{V}) \to 0$ as $k \to \infty$.

The above shows that to find a k-resolution centroid θ to minimize distortion, we need only look at the d_k dimensional space of k-resolution representations of all signals $x \in \mathcal{V}$. In other words,

$$\arg\min_{\theta \in \mathscr{S}_k} D(\mathcal{V},\theta) = \arg\min_{\theta \in \mathscr{S}_k} D_k(\mathcal{V},\theta) \qquad (3.5)$$

The minimum value that $D(\mathcal{V},\theta)$ can take for any θ is $\hat{D}_k(\mathcal{V})$, when $D_k(\mathcal{V},\theta) = 0$.

3.6.1 Projections of cells in multiple resolutions

Let us define how cells in one resolution are carried over to another resolution. Let $\mathscr{S}_i$ and $\mathscr{S}_j$ be the representations at two resolutions. Assume that we have a cell $\mathcal{V}_i$ in resolution i. Then the corresponding cell $\mathcal{V}_j$ in resolution j is

$$\mathcal{V}_j = \{\mathscr{S}_j x : \mathscr{S}_i x \in \mathcal{V}_i, x \in \mathcal{L}_2(\tau)\}$$

Note that if we take a cell in high-res space and project it to a low-res space and then project it back to the high-res space we might not necessarily end up with the original cell. This happens because more than one point in the high resolution space is projected onto the same point in the low resolution space.

Any convex cell $\mathcal{V}_k \subseteq \mathscr{S}_k$ in the k^{th} resolution is projected onto a convex cell at all resolutions. To see this, consider the projection of $\mathcal{V}_k$ onto $\mathcal{L}_2(\tau)$ defined as $\mathcal{V}_\infty = \{x \in \mathcal{L}_2(\tau) : \mathscr{S}_k x \in \mathcal{V}_k\}$. For any $x, y \in \mathcal{V}_\infty$ we have $\mathscr{S}_k x, \mathscr{S}_k y \in \mathcal{V}_x$. Then the convexity of $\mathcal{V}_k$ gives

$$\mathscr{S}_k(\alpha x + (1-\alpha)y) = \alpha \mathscr{S}_k x + (1-\alpha)\mathscr{S}_k y \in \mathcal{V}_k, \text{ for any } \alpha \in [0,1]$$

This implies that $\alpha x + (1-\alpha)y \in \mathcal{V}_\infty$ which shows that $\mathcal{V}_\infty$ is convex.

For any i, $\mathcal{V}_i = \{\mathscr{S}_i x : \mathscr{S}_k x \in \mathcal{V}_k\} = \{\mathscr{S}_i x : x \in \mathcal{V}_\infty\}$. For any $\mathscr{S}_i x, \mathscr{S}_i y \in \mathcal{V}_i$ we have $x, y \in \mathcal{V}_\infty$ and $\alpha x + (1-\alpha)y \in \mathcal{V}_\infty$ which implies that

$$\mathscr{S}_i(\alpha x + (1-\alpha)y) = \alpha \mathscr{S}_i x + (1-\alpha)\mathscr{S}_i y \in \mathcal{V}_i$$

which implies that $\mathcal{V}_i$ is convex. Thus convex cells in one resolution are projected onto convex cells in any other resolution.

Chapter 4

Compression using Multi-resolution TSVQ

As we mentioned earlier, MRTSVQ offers several advantages over unembellished TSVQ. To do the coarse quantization in the top layers of the tree structured codebook, it is frequently adequate to use a coarse representation of the signal. This in turn enables us to do distance calculations with far less computations than if the vectors were at the highest resolution. This computational advantage is very important if the number of letters in the codebook is very large. Furthermore, storing the centroids corresponding to the nodes of the tree requires less memory.

In this chapter we will present the greedy growing algorithm for MRTSVQs and establish some useful properties of this algorithm.

4.1 Preliminaries

As briefly mentioned in earlier chapters, the goal of compression is to quantize a real valued, random signal into a finite number of indexed values. Then these indices can be transmitted or stored instead of the original signal. Let $X \in \mathcal{L}_2(\tau)$ be a random variable that is distributed according to some density $p(x)$ as detailed in the previous chapter. Note that X is not a scalar or a vector, but a function defined on $[0, 1)$.

We want to find a mapping $Q : \mathcal{L}_2(\tau) \to \{\theta_1, \theta_2, \ldots, \theta_k\}$ where $\theta_i \in \mathcal{L}_2(\tau)$. Thus, given an $x \in \mathcal{L}_2(\tau)$, $Q(x)$ will be a quantized version of x. The error between x and its quantization $Q(x)$ is given by a *distortion function* $\rho(x, Q(x))$. $\rho(.,.)$ is non-negative everywhere and convex. Some examples of ρ were given in previous chapters.

In what follows, we will assume that $\rho(x, y) = ||x - y||^2 = \int_0^1 (x - y)^2 \, dt$. For $x, y \in \mathcal{S}_i$ denote

$$x = \sum_{j=1}^{d_i} c_j^i(x)\phi_j^i(t) = \overline{c}^i(x)^T \Phi(t)$$
$$y = \sum_{j=1}^{d_i} c_j^i(y)\phi_j^i(t) = \overline{c}^i(y)^T \Phi(t)$$

Then we can write $||x - y||^2 = ||W_i\overline{c}^i(x) - W_i\overline{c}^i(y)||_2^2$ where W_i is a non-singular matrix that depends on the basis functions at resolution i and $||.||_2$ is the ordinary squared norm of a vector. Thus, the distortion between two functions in $\mathcal{S}_i$ is equivalent to the squared error between linear transforms of their MRA coefficients. If we denote

$$\begin{aligned} \hat{x}^i &= W_i\overline{c}^i(x) \\ \hat{y}^i &= W_i\overline{c}^i(y) \end{aligned} \tag{4.1}$$

then the distortion between x and y in the $\mathcal{L}_2(\tau)$ space is equal to the squared error between $\hat{x}^i$ and $\hat{y}^i$ in the $\mathbb{R}^{d_i}$ space. The fact that it is easier to calculate the latter distortion compared to the former is what makes MRTSVQ much faster than TSVQ. This also allows us to extend, to $\mathcal{L}_2(\tau)$ results and algorithms that apply to $\mathbb{R}^d$. For any $\mathcal{C} \subseteq \mathcal{S}_i$, we denote $\hat{\mathcal{C}} = \{\hat{x}^i : x \in \mathcal{C}\} \subseteq \mathbb{R}^{d_i}$. In what follows we will denote $\hat{x} = \hat{x}^i$ whenever it is not necessary to make explicit the resolution of the decomposition.

If we represent all $x \in \mathcal{C} \subseteq \mathcal{S}_i$ with one representative $\theta \in \mathcal{S}_i$, the expected distortion is given by

$$D(\mathcal{C}, \theta) = E_p\{||X - \theta||^2 : X \in \mathcal{C}\} = \int_{\hat{\mathcal{C}}} ||\hat{x} - \hat{\theta}||_2^2 \, d\hat{x}$$

Denote by θ^* the representative that minimizes this distortion. As mentioned in the previous chapter, we call this the generalized centroid of $\mathcal{C}$. With the squared error distortion, the centroid is unique and lies within the convex hull of $\mathcal{C}$. With other distortion measures, these properties might not hold.

Now assume that we split $\mathcal{C}$ using two representatives θ_1 and θ_2. The LBG algorithm presented in Chapter 2 shows that the partition with the least distortion must be two cells separated by a hyper-plane given by

$$\{x \in \mathcal{C} : ||x - \theta_1||^2 = ||x - \theta_2||^2\}$$

and the partition is $\{\mathcal{C}_1, \mathcal{C}_2\} = \{\mathcal{C} \cap H, \mathcal{C} \cap H^c\}$ where

$$H = \{x : ||x - \theta_1||^2 < ||x - \theta_2||^2\}.$$

Denote by $\mathcal{C}_1$ the cell corresponding to θ_1 and $\mathcal{C}_2$ the cell corresponding to θ_2. Then $\mathcal{C}_1 \cup \mathcal{C}_2 = \mathcal{C}$. Let (θ_1^*, θ_2^*) be a set of two centroids that minimize the distortion

$$D(\mathcal{C}_1, \theta_1 + D(\mathcal{C}_2, \theta_2) = \int_{\hat{\mathcal{C}}_1} ||\hat{x} - \hat{\theta}_1||_2^2 \, d\hat{x} + \int_{\hat{\mathcal{C}}_2} ||\hat{x} - \hat{\theta}_2||_2^2 \, d\hat{x}$$

where all $\hat{x}$ is the transformed projection of the signal in the current resolution and $\hat{\mathcal{C}} = \{\hat{x} : x \in \mathcal{C}\}$.

In contrast to the case where we only had one centroid, here there can be more than one set of centroids (θ_1^*, θ_2^*) that minimize the above distortion. For example, take a probability distribution in $\hat{\mathcal{C}}$ that is radially symmetric. Then any rotation of an optimal centroid-couple will given another optimal centroid-couple.

The decrease in distortion when going from one centroid to two is given by

$$\triangle D(\mathcal{C}, \mathcal{C}_1, \mathcal{C}_2) = \int_{\hat{\mathcal{C}}} ||\hat{x} - \hat{\theta}^*||_2^2 \, d\hat{x} - \int_{\hat{\mathcal{C}}_1} ||\hat{x} - \hat{\theta}_1^*||_2^2 \, d\hat{x} - \int_{\hat{\mathcal{C}}_2} ||\hat{x} - \hat{\theta}_2^*||_2^2 \, d\hat{x}$$

This quantity plays an important role in deciding which leaf of the MRTSVQ to split.

Before proceeding, we need to present some supporting results that will be used later in this chapter. For these lemmas, assume that the cell $\mathcal{U} \in \mathcal{S}_i$ so that we can consider the space $\hat{\mathcal{U}} \subseteq \mathbb{R}^{d_i}$ of transformed coeffients as in 4.1.

Lemma 1 *For any cell $\mathcal{U} \in \mathcal{S}_i$ and hyper-plane H, we have $\triangle D(\hat{\mathcal{U}}, H) \geq 0$.*

Proof:

$$\begin{aligned}
D^*(\hat{\mathcal{U}}) &= \min_{\hat{\theta}} \int_{\hat{\mathcal{U}}} ||\hat{x} - \hat{\theta}||^2 p(\hat{x}) d\hat{x} \\
&= \min_{\hat{\theta}} \left(\int_{\hat{\mathcal{U}} \cap H} ||\hat{x} - \hat{\theta}||_2^2 p(\hat{x}) d\hat{x} + \int_{\hat{\mathcal{U}} \cap H^c} ||\hat{x} - \hat{\theta}||_2^2 p(\hat{x}) d\hat{x} \right) \\
&\geq \min_{\hat{\theta}} \int_{\hat{\mathcal{U}} \cap H} ||\hat{x} - \hat{\theta}||^2 p(\hat{x}) d\hat{x} + \min_{\hat{\theta}} \int_{\hat{\mathcal{U}} \cap H^c} ||\hat{x} - \hat{\theta}||^2 p(\hat{x}) d\hat{x} \\
&= D^*(\hat{\mathcal{U}} \cap H) + D^*(\hat{\mathcal{U}} \cap H^c)
\end{aligned}$$

$$\Rightarrow \triangle D(\hat{\mathcal{U}}, \hat{\mathcal{U}}_1, \hat{\mathcal{U}}_2) = D^*(\hat{\mathcal{U}}) - (D^*(\hat{\mathcal{U}} \cap H) + D^*(\hat{\mathcal{U}} \cap H^c)) \geq 0$$

This shows that any split will improve or leave unchanged the total distortion of a cell.

Lemma 2 *Assume a cell $\mathcal{U} \subseteq \mathcal{S}_k$ with an absolutely continuous density and non-zero probability that is split into two cells $\mathcal{U}_1, \mathcal{U}_2$ by θ_1, θ_2 belonging to the set of* *optimal centroids. Then we have* $\triangle D(\mathcal{U},\mathcal{U}_1,\mathcal{U}_2) > 0$ *and* $P(\mathcal{U}_1) > 0$ *and* $P(\mathcal{U}_2) > 0$.

Proof: Denote by θ_p the centroid of $\mathcal{U}$. Since the probability density of $\mathscr{S}_k x$ is absolutely continuous, we can find a partition $\mathcal{V}_1, \mathcal{V}_2$ such that θ_p is not one of the centroids of $\mathcal{V}_2$. This can be done by selecting $\mathcal{V}_2$ such that θ_p lies outside the convex hull of $\mathcal{V}_2$. Let θ_2 be a centroid of $\mathcal{V}_2$. Then we have

$$\begin{aligned}\triangle D(\mathcal{U},\theta) &\geq D(\mathcal{U},\theta_p) - D(\mathcal{V}_1,\theta_p) - D(\mathcal{V}_2,\theta_2)\\ &= D(\mathcal{V}_2,\theta_p) - D(\mathcal{V}_2,\theta_2)\\ &> 0\end{aligned}$$

The last inequality holds because θ_p is not a centroid of $\mathcal{V}_2$. Since we can find at-least one θ_1 and θ_2 such that $\triangle D > 0$, the result holds.

Without loss of generality, assume $P(\mathcal{U}_1) = 0$. Then $P(\mathcal{U}_2) = P(\mathcal{U})$. This implies that the distortion contributed by the elements in $\mathcal{U}_1$ is zero, which shows that the minimum distortion $D^*(\mathcal{U}_2) = D^*(\mathcal{U})$. This gives $\triangle D = 0$ which is contradicted by the previous result.

4.2 Greedy growing for MRTSVQ

The greedy growing algorithm for MRTSVQ is very similar to that for the ordinary TSVQ. The only difference is that here we need a rule that tells us whether to split a given node at the current resolution or to go to a higher resolution. One rule used in [3] is to go to the next higher resolution when the decrease in distortion obtained by splitting at the current level is lesser than some fixed fraction of the total distortion of the node. There can be many other possibilities. $d(t_i)$ will denote the dimension of the signal representation at node t_i. We will denote by ϕ the rule used to determine the resolution in which a node at any given depth is split.

This rule takes into account our tradeoff between computational ease at the lower resolution and higher potential decrease in distortion at a higher resolution. The computational burden increases linearly with the dimension of the vectors while the potential decrease in distortion on going from resolution k to $k+1$ decreases with increasing k for large k. Thus a reasonable rule would try to utilize as much of the information in the lower resolution as possible before going onto a higher resolution. Also if there is zero potential decrease in the current resolution, this rule will try to find at least one higher resolution k where the decrease in distortion is greater than zero.

We also have a splitting algorithm ψ that splits a given node to maximize the decrease in distortion. A reasonable splitting algorithm would start with a good initial position for the centroids and then use the LBG algorithm to converge to a local optimum.

We have used the notation ϕ, ψ to denote the rules that tell us when to go up in resolution and how to split. These should not be confused with the wavelet and scaling functions we presented in the last chapter that were denoted by the same letters. We will denote trees by the letter T. $T' \preceq T$ denotes that T' is a subtree of T. This means that T' and T have the same root node and all nodes belonging to T' also also belong to T. $T' \prec T$ implies that there is at-least one node in T that does not belong to T'. We will denote nodes by the letter t, leaves by $\tilde{t}$ and the set of all leaves of a tree T by $\tilde{T}$.

Given this, the algorithm for greedy growing is as follows:

Algorithm I:

1. Fix a splitting algorithm ψ and a rule ϕ that determines when to split at a higher resolution.

2. Initialize tree, $T = c(p_0, \mathcal{S}_0)$, where $c(p_0, \mathcal{S}_0)$ denotes the generalized centroid of $\mathcal{S}_0 \subseteq \mathbb{R}^{d_0}$, the zeroth resolution space under the corresponding probability density p_0. Initialize iteration count $i = 0$.

3. For any leaf node $\tilde{t}_j$ belonging to the set of leaf nodes $\tilde{T}$ of the tree with corresponding cell $U_j \subset \mathcal{S}_{\tilde{t}_j}$ at resolution $d(\tilde{t}_j)$ decide whether to go to the next higher resolution or to split at the same resolution.

4. Calculate $\triangle D(\psi, U_j)$ in the corresponding resolution.

5. Find the leaf node $\tilde{t}_{j^*}$ with the maximum value for $\triangle D(\psi, U_j)$

6. Implement the split for $\tilde{t}_{j^*}$.

7. If stopping criterion is reached, stop. Else, increase i by 1 and repeat steps 3-7

4.3 Properties of the algorithm

4.4 Property 1: Vanishing distortion

The following theorem shows that under a condition on the rule that decides which resolution to split in, the distortion of the tree can be made arbitrarily small by implementing the greedy growing algorithm.

Theorem 1 *Let Alg(p, k_n) denote the set of all trees produced by the greedy growing algorithm in k_n iterations and $T_n \in$ Alg(p, k_n). Then, if*

$$\frac{d(t_m)}{\log(depth(t_m))} \to 0$$

for the sequence of nodes $\{t_m\}$ on any branch, we have $D(T_n) \to 0$

Before we prove this theorem, we will need several preliminary results.

4.5 Preliminaries

Denote by $D(T)$ the average distortion of a tree structured VQ. We assume that the probability density p is fixed and will not explicitly mention it.

Lemma 3 *For any two trees T, T' such that $T' \prec T$*

$$D(T') \geq D(T)$$

Proof: The proof follows easily from Lemma 1 and the fact that T can be obtained from T' by splitting at-least one node

Lemma 4 *For any fixed splitting rule ψ,*

$$\sum_{t \in T} \triangle D(\mathcal{U}_t, \mathcal{U}_t^1, \mathcal{U}_t^2) \leq M_\infty(p)$$

where $\mathcal{U}_t^1, \mathcal{U}_t^2$ is the partition of of $\mathcal{U}_t$ given by ψ.

Proof: If we start with a tree of a single node at resolution 0 with a distortion $D^*(\mathcal{S}_0)$ and successively split leaf nodes to expand the tree, each split of a node t will decrease the original distortion by $\triangle D(\mathcal{U}_t, \mathcal{U}_t^1, \mathcal{U}_t^2)$. Since the total distortion of the tree always remains non-negative we get

$$D^*(\mathcal{S}_0) - \sum_{t \in T} \triangle D(\mathcal{U}_t, \mathcal{U}_t^1, \mathcal{U}_t^2) \geq 0$$

$$\Rightarrow \sum_{t \in T} \triangle D(\mathcal{U}_t, \mathcal{U}_t^1, \mathcal{U}_t^2) \leq D^*(\mathcal{S}_0) \leq M_\infty(p) \tag{4.2}$$

The last inequality comes from the optimality of $D^*(\mathcal{S}_0)$ and the fact that the function $x(t) = 0 : t \in [0, 1)$ belongs to $\mathcal{S}_0$.

When the greedy growing algorithm is applied n times to vectors from a fixed probability density p, we get a sequence of trees $T_0 \prec T_1 \prec \ldots \prec T_{n'}$. We call this a *trajectory.* If $n' = n$ we call it a complete trajectory. Incomplete trajectories can form if $\triangle D(\mathcal{U}_{\tilde{t}}, \mathcal{U}_{\tilde{t}}^1, \mathcal{U}_{\tilde{t}}^2) = 0$ for all $\tilde{t} \in \tilde{T}_n$ in all resolutions.

Lemma 5 *[34] If T has a complete trajectory of the form $T_0 \prec T_1 \prec \ldots \prec T_n = T$, then*

$$\min_{1 \leq j \leq n} \sum_{t \in \tilde{T}_{j-1}} \triangle D(\mathcal{U}_t, \mathcal{U}_t^1, \mathcal{U}_t^2) \leq \frac{M_\infty(p)}{w(n)}$$

where $w(n) = \sum_{j=1}^{n} j^{-1}$

Proof: For $j = 1, \ldots, n$ let $t_{j-1}^* \in \tilde{T}_{j-1}$ be the terminal node that is split to form T_j. Since T_{j-1} has j such nodes, the greedy node selection criterion ensures that

$$j^{-1} \sum_{t \in \tilde{T}_{j-1}} \triangle D(\mathcal{U}_t, \mathcal{U}_t^1, \mathcal{U}_t^2) \leq \triangle D(\mathcal{U}_{t_{j-1}^*}, \mathcal{U}_{t_{j-1}^*}^1, \mathcal{U}_{t_{j-1}^*}^2)$$

Therefore

$$w(n) \min_{1 \leq j \leq n} \sum_{t \in \tilde{T}_{j-1}} \triangle D(\mathcal{U}_t, \mathcal{U}_t^1, \mathcal{U}_t^2) \leq \triangle D(\mathcal{U}_{t_{j-1}^*}, \mathcal{U}_{t_{j-1}^*}^1, \mathcal{U}_{t_{j-1}^*}^2)$$

$$\leq \sum_{t \in T} \triangle D(\mathcal{U}_t, \mathcal{U}_t^1, \mathcal{U}_t^2) \leq M_\infty(p)$$

which gives the necessary result.

The following lemma is a geometric result that will be used later.

Lemma 6 *Consider two concentric balls $B_1 = B(0,r)$ and $B_2 = B(0, r\sqrt{d})$ in a d-dimensional space. Now take a hyper-plane in this space that is tangent to the inner ball B_1 and intersects the outer ball in a hyper-circle, thus dividing the surface of the outer ball into two parts, $\mathcal{V}_1$ and $\mathcal{V}_2$ with $Vol(\mathcal{V}_1) < Vol(\mathcal{V}_2)$ (See figure in Appendix). Then, we have*

$$\frac{Vol(\mathcal{V}_1)}{Vol(\mathcal{V}_1) + Vol(\mathcal{V}_2)} > \frac{1}{2\sqrt{2\pi e}} \forall d$$

Proof: Given in the Appendix.

Lemma 7 *If ψ is an optimal splitting rule, then for every set $\mathcal{U} \subseteq \mathcal{S}_k$ for some k, every distribution p and every number $\beta > 0$*

$$\triangle\triangle D(\mathcal{U}, \mathcal{U}^1, \mathcal{U}^2) \geq \frac{\beta^2}{2\sqrt{2\pi e}d} P(x \in \mathcal{U} : ||\mathscr{S}_k x - c(\mathcal{U}, k, p)|| > \beta) \tag{4.3}$$

Proof: Denote by $\hat{x}$ the k-th resolution representation of x in $\mathbb{R}^{d_k}$.We will not make the resolution k explicit as long as there is no confusion. As mentioned in the previous chapter, $||\mathscr{S}_k x - \mathscr{S}_k y||^2 = ||\hat{x} - \hat{y}||^2$ for any $x, y \in \mathcal{L}_2(\tau)$. Also denote by $\hat{c}$ the k-th resolution representation of $c(\mathcal{U}, k, p)$. Then

$$P(x \in \mathcal{U} : ||\mathscr{S}_k x - c(\mathcal{U}, k, p)|| > \beta) = P(x \in \mathcal{U} : ||\hat{x} - \hat{c}|| > \beta)$$

Consider two balls $B_1 = B(\hat{c}, \beta/\sqrt{d_k}), B_2 = B(\hat{c}, \beta)$ around $\hat{c}$ with radius $\beta/\sqrt{d_k}$ and β. Then, from the previous lemma, any hyper-plane H tangential to B_1 will have at-least $1/(2\sqrt{2\pi e})$ fraction of the surface area of B_2 on the side not containing the common center. This, in turn, means that there will be a hyper-plane H^* such that

$$P(H \cap {B_2}^c) > \frac{1}{2\sqrt{2\pi e}} P(x \in \mathcal{U} : ||\hat{x} - \hat{c}|| > \beta)$$

Let v^* be the vector in H that is closest to $\hat{c}$. Then $(\hat{c}-v^*)^t(\hat{x}-v^*) \leq 0$ holds for every $\hat{x} \in H$, which gives

$$||\hat{x}-\hat{c}||^2 - ||\hat{x}-v^*||^2 \geq ||\hat{c}-v^*||^2 = \frac{\beta^2}{d_k}$$

If $\hat{c}^1$ is the centroid of $\mathcal{V} = \mathcal{U} \bigcap H$ and $\hat{c}^2$ is the centroid of $W = \mathcal{U} \bigcap H^c$, then

$$\begin{aligned}
\triangle D(\mathcal{U},\mathcal{V},\mathcal{W}) &= D(\mathcal{U},\hat{c}) - D(\mathcal{V},\hat{c}^1) - D(W,\hat{c}^2) \\
&= D_k(\mathcal{U},\hat{c}) + R(\mathcal{U},k) - D_k(\mathcal{V},\hat{c}^1) - R(\mathcal{V},k) - D_k(W,\hat{c}^2) \\
&\quad -R(W,k) \\
&= D_k(\mathcal{U},\hat{c}) - D_k(\mathcal{V},\hat{c}^1) - D_k(W,\hat{c}^2)
\end{aligned}$$

$$\begin{aligned}
&= (D_k(\mathcal{V},\hat{c}) - D_k(\mathcal{V},\hat{c}^1)) + (D_k(W,\hat{c}) - D_k(W,\hat{c}^2)) \\
&\geq D_k(\mathcal{V},\hat{c}) - D_k(\mathcal{V},\hat{c}^1) \\
&\geq D_k(\mathcal{V},\hat{c}) - D_k(\mathcal{V},v^*) \\
&= \int_{\mathcal{V}} (\|x-c\|^2 - \|x-v^*\|^2)dp \\
&\geq \frac{\beta^2}{d} p(\mathcal{V}) \\
&\geq \frac{\beta^2}{2\sqrt{2\pi e}d} p(x \in \mathcal{U} : ||\hat{x}-\hat{c}|| > \beta)
\end{aligned}$$

which gives the required result.

Denote by $T(x)$ the function that maps $x \in \mathcal{U}_t$ to $c(\mathcal{U}_t, k_t, p)$ where $\mathcal{U}_t \subseteq L^2(\tau)$ is the cell associated with a leaf node $t \in T$ and $c(\mathcal{U}_t, k_t, p)$ is the centroid of $\mathcal{U}_t$ when it is split in the k_t-th resolution. Further denote by $k(x)$ the resolution k_t of the cell to which x belongs to and by $c(x)$ the centroid $c(\mathcal{U}_t, k_t, p)$. Then we have the following lemma

Lemma 8 *There exists constants $\beta, \gamma > 0$ depending only on δ and p such that for any tree T with $D(T) > \delta$, created after a sufficiently large number of iterations* *of the algorithm,*

$$P(x : \|\mathscr{S}_{k(x)} - c(x)\| \geq \beta) > \gamma$$

Proof: Assume that there is a number $K < \infty$ such that $P\{x : \|x\| \leq K\} = 1$. Then, for any x, y, $\|x - y\|^2 < 4K^2$. If $D(T) > \delta$ for any T, we have

$$\begin{aligned}\delta &\leq \int \|x - T(x)\|^2 dp \\ &\leq \frac{\delta}{2} + 4K^2 P\{x : \|x - T(x)\| \geq \sqrt{\frac{\delta}{2}}\}\end{aligned}$$

which gives

$$P\{x : \|x - T(x)\| \geq \sqrt{\frac{\delta}{2}}\} > \frac{\delta}{8K^2}$$

Now for any $x \in L^2(\tau)$ integer k and $c \in S_k$, $\|x-c\|^2 = \|\mathscr{S}_k x - c\|^2 + \|x - \mathscr{S}_k x\|^2$. Lemma 9 shows that the depth of the leaf node with the minimum depth keeps on increasing as the number of iterations increases. This implies that the resolution of the lowest resolution leaf node increases indefinitely and we can always find an iteration number such that for all leaf nodes of the tree

$$\|x - \mathscr{S}_{k(x)} x\|^2 < \epsilon^2 \; \forall x$$

for some $\epsilon > 0$ such that $\epsilon < \beta$.

This implies that for all leaf nodes in such a tree

$$\|x - c(x)\|^2 > \beta^2 \Rightarrow \|\mathscr{S}_{k(x)} x - c(x)\|^2 > \beta^2 - \epsilon^2$$

which gives

$$P\{x : \|S_{k(x)} - c(x)\|^2 > \beta^2 - \epsilon^2\} > P\{x : \|x - c(x)\| > \beta\} > \gamma$$

providing the desired result. The more general case where p does not have compact support can be derived from the result given in the appendix of [34].

Proof of Theorem 1: If the algorithm stops before all iterations have been finished, then it means that $\triangle D_k(\mathcal{U}_t, \mathcal{U}_t^1, \mathcal{U}_t^2) = 0$ for all leaf nodes $t \in \tilde{T}$ and all resolutions. This implies either that the probability $P(\mathcal{U}_t)$ is zero for all leaf nodes or that all the probability is concentrated at a point. In either case, the distortion of the tree is zero.

Now consider the case where the algorithm does not terminate until all iterations have been finished. For a large enough iteration number n, consider all trees thus formed which have a distortion $D(T) > \delta$. By Lemma 5, there exists a $T' \preceq T$ such that

$$\sum_{t' \in \tilde{T}'} \triangle D(\mathcal{U}_{t'}, \mathcal{U}_{t'}^1, \mathcal{U}_{t'}^2) \leq \frac{M_\infty(p)}{w(n)} \tag{4.4}$$

Since T' is a subtree of T, it follows from Lemma 3 that $D(T') \geq \delta$, and thus there exists constants $\beta, \gamma > 0$ depending only on δ and p such that

$$P\{x : \|\mathscr{S}_{k(x)}x - T'(x)\| \geq \beta\} > \gamma \tag{4.5}$$

when n is large. Then, the optimality of ψ and Lemma 7 imply that

$$\sum_{t' \in \tilde{T}'} \triangle D(\mathcal{U}_{t'}, \mathcal{U}_{t'}^1, \mathcal{U}_{t'}^2) \geq \frac{\beta^2}{2\sqrt{2\pi e} d_{max}} P\{x : \|x - T'(x)\| \geq \beta\} \geq \eta \tag{4.6}$$

where $\eta = \gamma\beta^2/2\sqrt{2\pi e} d_{max} > 0$ and d_{max} is the dimension of the MRA at the leaf node with the highest resolution. The above implies that

$$M_\infty(p)\frac{d_{max}}{w(n)} > \frac{\gamma\beta^2}{2\sqrt{2\pi e}} > 0 \tag{4.7}$$

If $d_{max}/w(n) \to 0$ as n increases, the left hand side goes to zero, while the right hand side is positive and independent of n. This shows that $D(T) > \delta$ for only a finite number of trees. Since δ was arbitrary, this gives the necessary result.

Note that $\ln(n) < w(n) < \ln(n) + 1$, so the condition above can also be written as $d_{max}/\ln(n) \to 0$.

4.6 Property 2: Termination with rate constraint

In the previous chapter we mentioned a stopping criterion for the greedy growing algorithm. The growth of the tree is stopped when this criterion is satisfied. For quantizers, a very common stopping criterion is in the form of a rate constraint. We keep on growing the tree as long as the expected bit-length of the codebook is lesser than a given R and stop as soon as it gets larger than R. This is reasonable when we have a channel with a fixed capacity and we want to constrain the quantizer to have a rate lesser than or equal to the channel capacity.

Here we will show that for a rate constraint, the algorithm given in the previous chapter will terminate after a finite number of iterations. This is important as it ensures that only a finite time is required to create a quantizer in practice.

Theorem 2 *For a stopping criterion of the form $r \leq R$, the algorithm Alg. 1 terminates after a finite number of iterations i*

The proof of this result uses the following lemma. By definition, a balanced binary tree is a binary tree where all leaf nodes are at the same depth and all nodes that are not leaf nodes have both their children present.

Lemma 9 *Let $T_1 \prec T_2 \prec \ldots$ be the sequence of trees created by each iteration of Alg. 1. Then there is a sequence of balanced trees $\breve{T}_1, \breve{T}_2, \ldots$ such that $\breve{T}_k \preceq T_k$ and $Depth(\breve{T}_k) \to \infty$*

Proof: Let $\breve{T}_i$ be the largest balanced tree that is a subtree of T_i. Then $Depth(\breve{T}_i)$ is non-decreasing. If $Depth(\breve{T}_i)$ does not go to infinity, then we have some K such that $Depth(\breve{T}_i) \to K$. This implies that there is at least one leaf node t and integer $n > 0$ such that t is a leaf node for all $T_i, i > n$. Then for all such T_i, any subtree $T \preceq T_i$ contains a leaf node that is either t or one of its ancestors. Lemma 2 gives us that none of t or its ancestors have probability zero, and that for t and its ancestors, $\triangle D(.) > 0$. Denoting by $Anc(t)$ the set composed of t and its ancestors, let $\delta = \min_{\hat{t} \in Anc(t)} \triangle D(\hat{t}) > 0$.

Since t or one of its ancestors are present in the set of leaf nodes of all T_i, we have

$$0 < \delta \leq \min_{1 \leq j \leq n} \sum_{t \in \tilde{T}_{j-1}} \triangle D(\mathcal{U}_t, k(t) : \psi) \leq \frac{M_\infty(p)}{w(k)}$$

Where the second inequality comes from Lemma 5 with $w(k) = \sum_{j=1}^{n} j^{-1}$.

Since $M_\infty / w(k) \to 0$ this fails to hold true for sufficiently large values of k. This shows that our assumption that $Depth(\breve{T}_i)$ converges to K is wrong and thus $Depth(\breve{T}_i) \to \infty$.

Proof of Theorem 2: Let $T_1 \preceq T_2$. Then $R(T_1) \leq R(T_2)$. Also the rate of a balanced tree $\breve{T}$ is equal to $Depth(\breve{T})$. These two, along with the previous lemma show that $R(T_i) \to \infty$, which shows that the algorithm Alg. 1 with a stopping criterion on the maximum rate will terminate after a finite number of iterations, providing the result we need.

4.7 Online algorithm for distortion minimization

An online algorithm for distortion minimization has been implied in [2] where the authors develop an online algorithm for combined compression and classification using VQ. Here we will present the algorithm and show how it is related to the LBG algorithm.

The algorithm presented in [2] is as follows. We assume a source of i.i.d. random vectors $x_1, x_2, x_3, \ldots$ that have a distribution $p(x)$. We start with an initial set of centroids $\Theta = \{\theta_1, \theta_2, \ldots, \theta_k\}$. Then for each observation x_n we find the centroid

θ_i that is closest to it in terms of the distortion function $\rho(x_n, \theta_i)$. This centroid is moved in the direction of x_n. Specifically

$$\theta_i(n) = \theta_i(n-1) + \alpha(n) \bigtriangledown_{\theta_i} \rho(x_n, \theta_i(n-1)) \tag{4.8}$$

As shown in [2], in the limit $\alpha(n) \to 0$, the trajectory of Θ follows the trajectory of the system of ordinary differential equations

$$\begin{aligned} \dot{\theta}_1 &= \int_{\mathcal{V}_1} p(x) \bigtriangledown_\theta \rho(x, \theta_1) dx \\ \dot{\theta}_2 &= \int_{\mathcal{V}_2} p(x) \bigtriangledown_\theta \rho(x, \theta_2) dx \\ &\vdots \\ \dot{\theta}_k &= \int_{\mathcal{V}_k} p(x) \bigtriangledown_\theta \rho(x, \theta_k) dx \end{aligned}$$

where $\mathcal{V}_i$ is the cell in the Nearest Neighbor partition that corresponds to centroid θ_i. Note that each $\mathcal{V}_i$ can be a function of *all* centroids $\theta_1, \theta_2, \ldots, \theta_k$.

In [2] the authors claim that this is a gradient descent on the cost

$$J(\Theta) = \sum_{i=1}^{k} \int_{\mathcal{V}_i} p(x)\rho(x, \theta_i) dx \tag{4.9}$$

Here we will prove this claim. To simplify the proof we will consider the case of only two centroids. The general k-centroid case can be similarly proved but we will only present pointers on how to extend this proof for that case.

The ODE that characterizes the behavior of the centroids in the 2-centroid algorithm is

$$\begin{aligned} \dot{\theta}_1 &= \int_{\mathcal{V}_1} p(x) \bigtriangledown_\theta \rho(x, \theta_1) dx \\ \dot{\theta}_2 &= \int_{\mathcal{V}_2} p(x) \bigtriangledown_\theta \rho(x, \theta_2) dx \end{aligned} \tag{4.10}$$

where $\mathcal{V}_1 = \{x \in \mathbb{R} : \rho(x,\theta_1) \leq \rho(x,\theta_2)\}$ and $\mathcal{V}_2 = \{x \in \mathbb{R} : \rho(x,\theta_2) < \rho(x,\theta_1)\}$. The cost 4.9 is

$$J(\Theta) = \sum_{i=1}^{2} \int_{\mathcal{V}_i} p(x)\rho(x,\theta_i)dx \tag{4.11}$$

Taking the gradient of the cost with respect to θ_1 or θ_2 is complicated by the fact that the regions of integration $\mathcal{V}_1$ and $\mathcal{V}_2$ are themselves functions of θ_1 and θ_2. To simplify notation, let us denote

$$g(x,\theta) = \rho(x,\theta)p(x)$$

Then the expected distortion $D(\theta_1,\theta_2)$ can be written as

$$D(\theta_1,\theta_2) = \int_{\mathcal{V}_1(\theta_1,\theta_2)} g(x,\theta_1)\,dx + \int_{\mathcal{V}_2(\theta_1,\theta_2)} g(x,\theta_2)\,dx \tag{4.12}$$

where we have made explicit, the dependence of $\mathcal{V}_1$ and $\mathcal{V}_2$ on θ_1 and θ_2.

To find the gradient of 4.12 with respect to θ_1, let us compute

$$\begin{aligned}
D(\theta_1+\triangle\theta,\theta_2) \;-\; & D(\theta_1,\theta_2) \\
= & \int_{\mathcal{V}_1(\theta_1+\triangle\theta,\theta_2)} g(x,\theta_1+\triangle\theta)\,dx \\
& + \int_{\mathcal{V}_2(\theta_1+\triangle\theta,\theta_2)} g(x,\theta_2)\,dx - \int_{\mathcal{V}_1(\theta_1,\theta_2)} g(x,\theta_1)\,dx + \int_{\mathcal{V}_2(\theta_1,\theta_2)} g(x,\theta_2)\,dx \\
= & \int_{\mathcal{V}_1(\theta_1+\triangle\theta,\theta_2)} g(x,\theta_1+\triangle\theta)\,dx - \int_{\mathcal{V}_1(\theta_1,\theta_2)} g(x,\theta_1+\triangle\theta)\,dx \\
& + \int_{\mathcal{V}_1(\theta_1,\theta_2)} g(x,\theta_1+\triangle\theta)\,dx - \int_{\mathcal{V}_1(\theta_1,\theta_2)} g(x,\theta_1)\,dx \\
& + \int_{\mathcal{V}_2(\theta_1+\triangle\theta,\theta_2)} g(x,\theta_2)\,dx - \int_{\mathcal{V}_2(\theta_1,\theta_2)} g(x,\theta_2)\,dx
\end{aligned}$$

Denote $\mathcal{W}_1 = \mathcal{V}_1(\theta_1+\triangle\theta,\theta_2) \cap \mathcal{V}_2(\theta_1,\theta_2)$ and $\mathcal{W}_2 = \mathcal{V}_2(\theta_1+\triangle\theta,\theta_2) \cap \mathcal{V}_1(\theta_1,\theta_2)$ (see Fig. 4.1). Then we can write

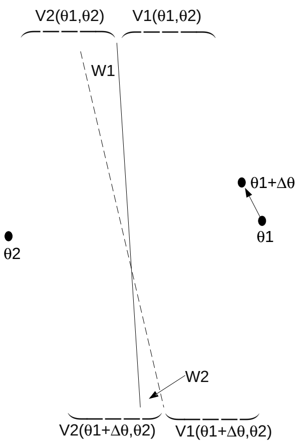


Figure 4.1: Illustration of $\mathcal{V}_1, \mathcal{V}_2, \mathcal{W}_1$ and $\mathcal{W}_2$ in 2 dimensions

$$
\begin{aligned}
D(\theta_1 + \triangle\theta, \theta_2) \quad - \quad & D(\theta_1, \theta_2) \\
= \quad & \int_{\mathcal{V}_1(\theta_1,\theta_2)} [g(\theta_1 + \triangle\theta, x) - g(\theta_1)]\, dx \\
& + \int_{W_1} g(\theta_1 + \triangle\theta, x)\, dx - \int_{W_2} g(\theta_1 + \triangle\theta, x)\, dx \\
& + \int_{W_2} g(\theta_2, x)\, dx - \int_{W_1} g(\theta_2, x)\, dx \\
= \quad & \triangle\theta^T \int_{\mathcal{V}_1(\theta_1,\theta_2)} \bigtriangledown g(\theta_1, x)\, dx + \text{ (higher order terms in } \triangle\theta) \\
& + \int_{W_1} g(\theta_1 + \triangle\theta, x)\, dx - \int_{W_2} g(\theta_1 + \triangle\theta, x)\, dx \\
& + \int_{W_2} g(\theta_2, x)\, dx - \int_{W_1} g(\theta_2, x)\, dx
\end{aligned}
$$

The sets W_1 and W_2 are infinitesimally thin sets along the hyper-plane that separates $\mathcal{V}_1$ and $\mathcal{V}_2$. Since the definition of this hyper-plane is such that $H = \{x : \rho(x, \theta_1) < \rho(x, \theta_2)\}$ we have $g(\theta_1 + \triangle\theta, x) - g(\theta_2, x) = O(\triangle\theta)$. Thus

$$
\begin{aligned}
\int_{W_1} g(\theta_1 + \triangle\theta, x)\, dx - \int_{W_2} g(\theta_1 + \triangle\theta, x)\, dx \quad + \quad & \int_{W_2} g(\theta_2, x)\, dx - \int_{W_1} g(\theta_2, x)\, dx \\
= \quad & 0 + \text{ (higher order terms in } \triangle\theta)
\end{aligned}
$$

Thus

$$D(\theta_1+\triangle\theta,\theta_2)-D(\theta_1,\theta_2) = \triangle\theta^T \int_{\mathcal{V}_1(\theta_1,\theta_2)} \bigtriangledown g(\theta_1,x)\,dx + \text{ (higher order terms in } \triangle\theta)$$

which implies that

$$\begin{aligned}\bigtriangledown_{\theta_1} D(\theta_1,\theta_2) &= \int_{\mathcal{V}_1} \bigtriangledown_{\theta_1} g(\theta_1,x)\,dx \\ &= \int_{\mathcal{V}_1} \bigtriangledown_{\theta_1} \rho(\theta_1,x)p(x)\,dx\end{aligned}$$

Similarly

$$\bigtriangledown_{\theta_2} D(\theta_1,\theta_2) = \int_{\mathcal{V}_2} \bigtriangledown_{\theta_2} \rho(\theta_2,x)p(x)\,dx$$

which is the result we wanted.

Note that the method of proof depended on the fact that the infinitesimal change in total distortion is zero when a perturbation in θ_1 causes a point close to the separating hyper-plane to move from $\mathcal{V}_1$ to $\mathcal{V}_2$. This method cannot be used to establish the cost function that is minimized by the Learning Vector Quantization algorithm.

The extension of this proof to more than two centroids is messy but straightforward. Instead of looking at the hyper-plane between only two centroids we have to look at the hyper-planes between a given centroid and all other centroids. An argument similar to the one above shows that perturbation in the position of one centroid does not change the distortion contribution from points near the separating hyper-planes.

4.8 Practical implementation of the MRTSVQ

Practical implementation of the tree structured quantizer brings up several issues that have not been addressed above. Two of them are initialization of centroids and choosing ϕ, the function that tells whether to split in the current resolution or the next. Initialization of centroids is accomplished by randomly selecting data vectors that falls into the cell. This usually gives good results. If we have more than two centroids at each split, we can initially split with just two centroids and then introduce additional centroids one after another at random positions.

One way of deciding whether to go up a resolution is by comparing the change in distortion on splitting at the current resolution to the total distortion. If the change is less than a certain fixed fraction (call it δ_D), then we split at the next higher resolution. Increasing δ_D will force the algorithm to go to higher resolutions early on, but will result in lower distortion for a given rate. If δ_D is chosen small the algorithm exploits more of the information in the lower resolution before trying out the higher resolutions. The tradeoff is that one ends up with a higher rate for the same total distortion level.

Another way of choosing the resolution to split at is to look at a weighted sum of the computational complexity and the decrease in distortion for a split at a given resolution. The resolution that minimizes this function will be chosen for the splitting. Different tradeoffs between computational complexity and tree efficiency can be obtained with different weights.

Chapter 5

Multi-resolution classifiers

In this chapter we will explore the use of Multi-Resolution TSVQ as a classifier. We assume that a source can be in any one of two states or *classes* when it outputs a signal $x(t), t \in [0,1)$. Labeling the classes by 1 or 2, we have a probability density defined on $\mathcal{L}^2(\tau)$ that depends on whether the source is in class 1 or 2. Let $p_1(x)$ denote the probability density at x given that the source was in class 1 and $p_2(x)$ denote the density given that the source was in class 2. We assume that either class occurs independently with probabilities π_1 and $\pi_2 = 1 - \pi_1$.

Our problem is to estimate the class of the source given only the output x. As mentioned in Chapter 3, if we know the prior probabilities and the marginal probability densities, the *Bayes optimal classifier* achieves the least possible misclassification error. If all we have is a training set consisting of pairs $\{x_n, c_n\}$ of observations and classes, we need algorithms that find a classifier that comes as close to the Bayes optimal classifier as possible.

5.1 Multi-resolution vector classification

Now consider the above problem of constructing a classifier from training samples with the added feature that the signals output by the source can be observed in various levels of detail. The first question is: How is this useful? The answer is that processing each signal to determine the class might be easier if less detailed versions of the signal are used. Most classifiers compare the signal to a set of exemplars and this comparison is usually computationally simpler if the signal is of lower dimension.

Additionally, in some cases, most of the high level detail of the signal is swamped in noise and only the low resolution features offer usable information about the class of the signal. In such a case using the high resolution details might make the classifier perform worse compared to using low resolution representation since the algorithm tries to model the noise.

Multi-resolution analysis is used to find out good features for other kinds of classifiers too. Typically, the multiple resolutions of the signal are examined to find those resolutions that have the most potential for the classification problem. In a TSVQ, this selection is done automatically. At each step of the tree growing process, we not only decide which node to split, but also which resolution to split it in. A resolution that offers the best trade-off between dimensionality and error reduction is chosen.

The second question is: How can we construct hierarchical classifiers using training sets in multiple resolutions? Here we will provide one solution based on an extension of the *Learning Vector Quantization* we presented in Chapter 2.

5.2 Multi-level LVQ

Taking the algorithm used to create the MRTSVQ for compression, we can try to extend LVQ to the multi-resolution tree structured classifier in a straightforward way. We start out with a root node that contains as its cell, all the training vectors at the lowest resolution. The class of the root node is assigned according to the majority vote amongst the training vectors; if the class 1 vectors outnumber the class 2 vectors, then the class of the node is 1 and vice-versa. Then, recursively, each leaf node of the tree is examined to find the one that gives the biggest decrease in the classification error when its cell is split into two parts, each part assigned a different class. This splitting is done by the LVQ algorithm at an appropriate resolution. The node that gives the highest difference is then split to produce a new tree and this process repeats.

The only problem with this algorithm is that LVQ takes a long time to converge (learning rate $\alpha(n) = O(1/n)$). Waiting until the centroids at a higher resolution have converged before going on to the next resolution is very time consuming. Additionally, in an online learning case, we cannot use the classifier until all the levels of the tree have converged. In such a case, we would be interested in an algorithm where all levels of the tree are simultaneously updated when each data sample arrives.

Such an algorithm would start out with a structure for a tree with centroids for each node initialized to appropriate values. Then, as each training sample arrives, it will be used to update the centroids of all nodes of the tree at the appropriate resolution simultaneously. Thus the tree, as a whole, adapts to the data rather than each node adapting separately. The classifier tree can be used at any time for predicting the class, with a misclassification error that goes down as the centroids converge.

Let us now present the formal algorithm

Algorithm II:

1. Fix in advance the MRA, size of tree (maximum depth), data size and number of iterations.

2. Start with an initial tree T_0 consisting of a structure that details:

   (a) How the nodes are connected to each other; i.e. which nodes are the children of a given node.

   (b) The resolution of each node. Each node is at the same resolution as its siblings and we require that the resolution is non-decreasing down the tree along every branch.

   (c) The initial positions of the centroids at each node. The centroids at each node must be at the resolution of the node.

3. Initialize iteration number $i = 1$ and data number $n = 1$.

4. Take data vector x_n at the lowest resolution and compare it to the centroids of the children of the root node and perform the following update

$$\begin{aligned}\theta_0^1(t+1) &= \theta_0^1(t) - \alpha_0(t)(\mathscr{S}_0 x_n - \theta_0^1(t)) \\ &\qquad \text{if } \|\mathscr{S}_0 x_n - \theta_0^1(t)\|^2 < \|\mathscr{S}_0 x_n - \theta_0^2(t)\|^2 \text{ and } c_n = 1 \\ \theta_0^1(t+1) &= \theta_0^1(t) + \alpha_0(t)(\mathscr{S}_0 x_n - \theta_0^1(t)) \\ &\qquad \text{if } \|\mathscr{S}_0 x_n - \theta_0^1(t)\|^2 < \|\mathscr{S}_0 x_n - \theta_0^2(t)\|^2 \text{ and } c_n = 2\end{aligned}$$

$$\begin{aligned}
\theta_0^2(t+1) &= \theta_0^2(t) - \alpha_0(t)(\mathscr{S}_0 x_n - \theta_0^2(t)) \\
&\qquad \text{if } \|\mathscr{S}_0 x_n - \theta_0^2(t)\|^2 < \|\mathscr{S}_0 x_n - \theta_0^1(t)\|^2 \text{ and } c_n = 2 \\
\theta_0^2(t+1) &= \theta_0^2(t) + \alpha_0(t)(\mathscr{S}_0 x_n - \theta_0^2(t)) \\
&\qquad \text{if } \|\mathscr{S}_0 x_n - \theta_0^2(t)\|^2 < \|\mathscr{S}_0 x_n - \theta_0^1(t)\|^2 \text{ and } c_n = 1
\end{aligned}$$

5. Recursively, for each pair of nodes θ_i^1, θ_i^2 at resolution k, do the following update only if their parent was updated.

$$\begin{aligned}
\theta_i^1(t+1) &= \theta_i^1(t) - \alpha_k(t)(\mathscr{S}_k x_n - \theta_i^1(t)) \\
&\qquad \text{if } \|\mathscr{S}_k x_n - \theta_i^1(t)\|^2 < \|\mathscr{S}_k x_n - \theta_i^2(t)\|^2 \text{ and } c_n = 1 \\
\theta_i^1(t+1) &= \theta_i^1(t) + \alpha_k(t)(\mathscr{S}_k x_n - \theta_i^1(t)) \\
&\qquad \text{if } \|\mathscr{S}_k x_n - \theta_i^1(t)\|^2 < \|\mathscr{S}_k x_n - \theta_i^2(t)\|^2 \text{ and } c_n = 2
\end{aligned}$$

$$\begin{aligned}
\theta_i^2(t+1) &= \theta_i^2(t) - \alpha_k(t)(\mathscr{S}_k x_n - \theta_i^2(t)) \\
&\qquad \text{if } \|\mathscr{S}_k x_n - \theta_i^2(t)\|^2 < \|\mathscr{S}_k x_n - \theta_i^1(t)\|^2 \text{ and } c_n = 2 \\
\theta_i^2(t+1) &= \theta_i^2(t) + \alpha_k(t)(\mathscr{S}_k x_n - \theta_i^1(t)) \\
&\qquad \text{if } \|\mathscr{S}_k x_n - \theta_i^2(t)\|^2 < \|\mathscr{S}_k x_n - \theta_i^2(t)\|^2 \text{ and } c_n = 1
\end{aligned}$$

6. Increase data number by 1. Repeat for all data vectors.

7. Increase iteration number by 1. Repeat for all iterations.

Note that there is an implicit linkage between a node and its parent. Only the vectors that cause the parent to be updated will be considered for updating the node and its siblings. Thus, the only vectors being used to split a node will be the vectors that fall into its cell.

5.3 Convergence of multi-level LVQ

In our algorithm, the decision whether to consider a vector for the update of a node depends on whether the vector fell in the cell of the node's parent. Since the cell of the parent changes with the position of the centroid at each time step, the centroids of the node have to try and keep up with a "moving target." We can ask the question: Under what conditions will the centroids of the tree converge to a stationary value as the number of data vectors increase indefinitely?

To investigate this, we will need to present and prove a theorem that is a slightly different from the result shown by Borkar [5]. While [5] deals with stochastic approximation algorithms in two levels with different "training speeds", we need to generalize it to arbitrary number of levels K. In addition, the algorithm considered in [5] has an update function that depends on the state variables at lower levels (i.e. update for state x_i depends on x_j, $j > i$). We will only need update functions that depend only on state variables at higher levels. This adds some simplicity to the proof.

5.3.1 Preliminaries

In what follows, we will denote by $x_i(n) \in \mathbb{R}^{d_i}$ the d_i-dimensional component of the state space in the i-th level at time step n. $X_i(n) = [x_0(n), x_1(n), \ldots x_i(n)]$ denotes the set of state variables that are at or above the i-th level. $X(n) = X_{K-1}(n) = [x_0(n), x_1(n), \ldots x_{K-1}(n)]$ denotes the total state space of the algorithm.

Consider the stochastic approximation algorithm consisting of the linked difference equations

$$\begin{aligned}
x_0(n+1) &= x_0(n) + \alpha_0(n)[f_0(X_0(n)) + M_0(n)] \\
x_1(n+1) &= x_1(n) + \alpha_1(n)[f_1(X_1(n)) + M_1(n)] \\
\vdots \quad & \quad \vdots \\
x_{K-1}(n+1) &= x_{K-1}(n) + \alpha_{K-1}(n)[f_{K-1}(X_{K-1}(n)) + M_{K-1}(n)] \qquad (5.1)
\end{aligned}$$

where

- A.1 $f_i(X_i(n)) : \mathbb{R}^{d_0+d_1+d_2+\ldots d_i} \to \mathbb{R}^{d_i}$ is Lipschitz for all i
- A.2 $\sum_n \alpha_i(n) = \infty$ and $\sum_n \alpha_i(n)^2 < \infty$ for all i
- A.3 $\alpha_i(n)/\alpha_j(n) \to 0$ if $i < j$ for all i, j
- A.4 If $F_n = \sigma\{X_i(l), M_i(l) | l \leq n, \forall i\}$ denotes the σ-algebra formed by the state X_i and the noise M_i, then $(M_i(n), F_n)$ are random variables satisfying

$$\sum_n \alpha_i(n) M_i(n) < \infty \text{ a.s.}$$

In the case of the multi-level LVQ, we will show that $G_i(n) = \sum_{m=1}^{n} \alpha_i(m) M_i(m)$ is a martingale; then condition A.4 follows from a version of the martingale convergence theorem.

The usual method of analyzing the convergence of a system of stochastic approximations like 5.1 is to compare it to the associated ordinary differential equation (ODE). Denoting by the same symbols $\{x_i\}, \{X_i\}$, the state variables in continuous time, we have the system of equations

$$\begin{aligned}
\dot{x}_0(t) &= f_0(X_0(t)) \\
\epsilon \dot{x}_1(t) &= f_1(X_1(t)) \\
\epsilon^2 \dot{x}_2(t) &= f_2(X_2(t)) \\
&\vdots \\
\epsilon^{K-1} \dot{x}_{K-1}(t) &= f_{K-1}(X_{K-1}(t))
\end{aligned} \tag{5.2}$$

for the limit $\epsilon \downarrow 0$. This is a generalized *singularly perturbed system* where each variable x_i behaves as if all variables x_0 to x_{i-1} are constants. Denote by $\lambda_i(x_0, x_1, \ldots, x_{i-1})$ the function that gives the equilibrium point of $\dot{x}_i(t) = f_i(X_i(t))$ for given constant values of $x_0, x_1, \ldots, x_{i-1}$.

Now suppose that there is an asymptotically stable equilibrium $X^* = [x_0^*, x_1^*, \ldots, x_{K-1}^*]$ for the system of equations 5.2. Then we have the following theorem

Theorem 3 *If there is an N such that $X(n)$ remains in the domain of attraction of X^* for all $n > N$ and $\sup_n X(n) < \infty$, the iterates 5.1 converge to X^* a.s.*

The assumption that $X(n)$ remains in the domain of attraction of the equilibrium and that its supremum is bounded might not be always valid. In that case, we can keep projecting $X(n)$ back into the bounding set at the expense of an error term. [25] shows how to deal with the error term in the case of constrained optimization.

5.3.2 Proof of the theorem

Our proof closely parallels and borrows much of its notation from [5]. We will first show that the state variable $x_i(t)$ at any level $i > 0$ converges to $x_i^* = \lambda_i(x_0, x_1, \ldots, x_{i-1})$. Then we will show that $x_0(t)$ converges to x_0^*.

First, a definition and a lemma. Consider the ODE in $\mathbb{R}^d$ given by

$$\dot{z}(t) = h(z(t)) \tag{5.3}$$

for a Lipschitz h such that Eq. 5.3 has an asymptotically stable attractor J with a domain of attraction $D(J)$. Given $T, \delta > 0$, we call a bounded measurable $y(t) : R^+ \to R^d$ a (T, δ)-perturbation of Eq. 5.3 if there exists $0 = T_0 < T_1 < \ldots < T_n = \infty$ with $T_{i+1} - T_i \geq T$ and solutions $z^j(t)$ of Eq. 5.3 such that

$$\sup_{t \in [T_j, T_j+1]} ||z^j(t) - y(t)|| < \delta$$

Lemma 10 *Given $\epsilon, T > 0$ there exists a $\bar{\delta} > 0$ such that for $\delta \in (0, \bar{\delta})$, every (T, δ)-perturbation of Eq. 5.3 converges to the ϵ-neighborhood J^ϵ of J.*

The proof is given in the appendix of [5]

Fix a level i. We will now show that as far as this level is concerned, the ODE being followed by 5.1 is

$$\begin{aligned}
\dot{x}_0(t) &= 0 \\
\dot{x}_1(t) &= 0 \\
\vdots \quad & \quad \vdots \\
\dot{x}_{i-1}(t) &= 0 \\
\dot{x}_i(t) &= f_i(X_i(t))
\end{aligned}$$

For a given T let $t_i(0) = 0 = T_i(0)$ and define

$$\begin{aligned} t_i(n) &= \sum_{k=1}^{n} \alpha_i(k), n \geq 1 \\ m(0) &= 0, \\ m(n) &= \min \left\{ k \geq m(n-1) | \sum_{j=m(n-1)+1}^{k} \alpha_i(j) \geq T \right\} \\ T_i(n) &= t_i(m(n)) \end{aligned} \tag{5.4}$$

With this notation we have $T_i(j+1) \in [T_i(j)+T, T_i(j)+T+C_i], C_i = \sup_n \alpha_i(n)$. Define $\bar{X}_i(t) = [\bar{x}_0(t), \bar{x}_1(t), \ldots, \bar{x}_i(t)]$ as $\bar{X}_i(t_i(n)) = X_i(n)$ with a linear interpolation for the time between the instants $t_i(n)$ and $t_i(n+1)$.

Consider the system of equations 5.4. We have the lemma

Lemma 11 *For any $\delta > 0$, there exists a time $t_\delta > 0$ such that $\bar{X}_i(t + t_\delta)$ is a (T, δ) perturbation of 5.4*

Proof: Rewrite the stochastic approximation algorithm 5.1 up to level i as

$$\begin{aligned} x_0(n+1) &= x_0(n) + \alpha_i(n) \frac{\alpha_0(n)}{\alpha_i(n)} [f_0(X_0(n)) + M_0(n)] \\ x_1(n+1) &= x_1(n) + \alpha_i(n) \frac{\alpha_1(n)}{\alpha_i(n)} [f_1(X_1(n)) + M_1(n)] \\ &\vdots \qquad \vdots \\ x_i(n+1) &= x_i(n) + \alpha_i(n)[f_i(X_i(n)) + M_i(n)] \end{aligned} \tag{5.5}$$

Let $X_i^n(t)$ be the solution to 5.4 on $[T_i(n), \infty)$ with initial conditions $X_i^n(t) = X_i(n)$. Then 5.5 can be seen as a discretized version of 5.4 with step sizes $\alpha_i(n)$ and an error

$$\frac{\alpha_j(n)}{\alpha_i(n)} [f_j(X_j(n)) + M_j(n)]$$

at the j-th level, $j < i$ and $\alpha_i(n)M_i(n)$ at the i-th level. The fact that all f_j are Lipschitz and that $\alpha_j(n)/\alpha_i(n) \to 0$ for all $j < i$ makes the contribution of this error asymptotically negligible as $n \to \infty$. With the fact that $\alpha_i(n) \to 0$, we get the required result by a standard approximation argument using the Gronwall inequality.

Lemma 12 $X_i(n) \to [x_0, x_1, \ldots, x_{i-1}, \lambda_i(x_0, x_1, \ldots, x_{i-1})]$

This follows from the above two lemmas.

Now we have to show that $x_0(n)$ converges. This follows in a straightforward fashion from

Lemma 13 *Under the conditions in Theorem 3 and A.1 to A.4, $x_0(n) \to x_0^*$ where x_0^* is the equilibrium solution of*

$$\dot{x}_0(t) = f_0(x_0(t))$$

a.s.

Proof: See [25]

This completes the proof of Theorem 3

5.3.3 Multi-level LVQ

The application of Theorem 3 to the Multi-level LVQ algorithm is straightforward once we set the algorithm up in the form of 5.1 and verify that assumptions A.1 to A.4 are satisfied.

To make the link between Algorithm 1 in section 5.2 and the stochastic approximation algorithm 5.1, let us denote the set of all centroids at resolution k by $\Theta_k(n) = \{\theta_i^1(n), \theta_i^2(n)\}$ node i belongs to the k-th level of the tree. Also denote by $\tilde{\mathscr{S}_k}x = [\mathscr{S}_k x, \mathscr{S}_k x, \ldots, \mathscr{S}_k x]$ where the number of repetitions is equal to the number of centroids at resolution k. Then the LVQ algorithm can be written as

$$\begin{aligned}
\Theta_0(n+1) &= \Theta_0(n) + \alpha_0(n)\eta_0(\Theta_0(n))[\Theta_0(n) - \tilde{\mathscr{S}_0}x(n)] \\
\Theta_1(n+1) &= \Theta_1(n) + \alpha_1(n)\eta_1(\Theta_0(n), \Theta_1(n))[\Theta_1(n) - \tilde{\mathscr{S}_1}x(n)] \\
\vdots \quad &\quad \vdots \\
\Theta_{K-1}(n+1) &= \Theta_{K-1}(n) \\
&\quad + \alpha_{K-1}(n)\eta_{K-1}(\Theta_0(n), \Theta_1(n), \ldots, \Theta_{K-1}(n))[\Theta_{K-1}(n) - \tilde{\mathscr{S}_{K-1}}x]
\end{aligned}$$

where $\eta_i(\Theta_0, \Theta_1, \ldots, \Theta_i)$ is a diagonal matrix of size $l_k d_k \times l_k d_k$ where l_k is the number of centroids and d_k is the dimension of the signal vector at the resolution k. The diagonal is composed of blocks of length d_k where all elements of the j-th such block are

$$\begin{pmatrix}
0 & \text{if vector } x \text{ does not fall into each ancestor of centroid } j \\
1 & \text{if } x \text{ falls into each ancestor of } j \text{ and the class of } x \text{ is different from class of } j \\
-1 & \text{if } x \text{ falls into each ancestor of } j \text{ and the class of } x \text{ is same as class of } j
\end{pmatrix}$$

Here centroid j refers to the j-th amongst the l_k centroids at resolution k.

Now let us show that assumption A.4 is satisfied

$$f_i(\Theta_0(n), \Theta_1(n), \ldots, \Theta_i(n)) = E_x\{\eta_i(\Theta_0(n), \Theta_1(n), \ldots, \Theta_i(n))[\Theta_i(n) - \tilde{\mathscr{S}_i}x]\}$$

and

$$M_i(n) = \eta_i(\ldots)[\Theta_i - \tilde{\mathscr{S}_i}x] - f_i(\ldots)$$

Then $E_x(M(n)) = 0$. Given that x_n and x_m are uncorrelated for $n \neq m$, it is easy to verify that

$$E\{M_i(n)M_i(m)\} = 0$$

These two facts imply that $\{M_i(n)\}$ is a martingale difference sequence and that

$$G_i(n) = \sum_{m=1}^{n} \alpha_i(m) M_i(m)$$

is a martingale. Then, a martingale convergence theorem like Theorem 5.14 in [7] shows that $G_i(n)$ converges a.s.

A.2 and A.3 are design issues, so the only other assumption we have to check is A.1.; i.e. that $f_i(\ldots)$ is Lipschitz. It is not hard to verify that this is satisfied if x has a probability distribution $p(x) = \pi_1 p_1(x) + \pi_2 p_2(x)$ that is bounded.

Thus, all assumptions for Theorem 3 are satisfied and we get the result that all centroids $\theta_j(n)$ converge as $n \to \infty$.

5.4 Issues in practical implementation

Practical implementation of the above algorithm introduces several issues that are not readily apparent from the description. Here we will discuss some of them and offer heuristic solutions.

The convergence result holds only if there is an equilibrium solution to the associated ODE and if the initial conditions are close enough to it. The ODE might not have an equilibrium solution if, for a cell, there is no hyper-plane that will partition it into two parts with clear majorities in each part. This can mean two things; either there is no more improvement possible by splitting that node at the current resolution or improvement is possible only with more than two partitions. If it is the former case one might try splitting at a higher resolution. In the latter case the algorithm can be modified to include the possibility of splitting a node with more than two partitions and its convergence proved without difficulty.

Proper initialization of the centroids is very important for fast convergence.

Several heuristic rules exist for good initialization. One method that works well is randomly selecting a data vector of each class as initial values. Another method is to initialize the centroid to the mean of the data vectors that belong to its class.

It is necessary to initialize the class of each centroid according to the majority vote in its partition and update the class after a number of iterations. [26] shows an example where failure to do this will result in divergence of the centroids even though there is a stable solution. It might happen that this updating causes the class of both centroids to be the same; in such a case re-assignment of the centroids and their classes is the only recourse.

Chapter 6

Simulations and applications of MRTSVQ

In this chapter we will present two simulation examples for compression and classification. Working on synthetic data, these simulations will illustrate the implementation of the algorithms we presented in earlier chapters and their typical behavior.

We will also present several applications of clustering and classification using MRTSVQ. Tree structured VQ will be used to cluster yeast genes according to their expression profile and for classification of cells into tumerous and non-tumerous classes. Then we will present a parallel tree method for predicting wear on a milling tool. Computational advantages of the MRTSVQ method will be illustrated by an application in fingerprint identification.

6.1 Simulation for compression

In this section we will show the results obtained by using the greedy algorithm for quantization of signals. We have used the algorithm described in Chapter 4. Splitting was done using the LBG algorithm with random initial values for the centroids. The decision to go up in resolution was taken on the basis of the decrease in distortion on splitting a leaf. If the decrease was less than a fixed fraction of the total distortion of the leaf, we proceeded to the higher resolution.

The quantized signals were created as the output of a linear filter of 2nd order with an i.i.d, Gaussian noise input. These signals were then analyzed in 7 resolutions using a Bi-orthogonal wavelet of order 2 for both analysis and reconstruction. The resulting coefficient vectors were transformed to have the same norm as the function they approximate. Fig. 6.1 shows an example of a signal at 7 resolutions.

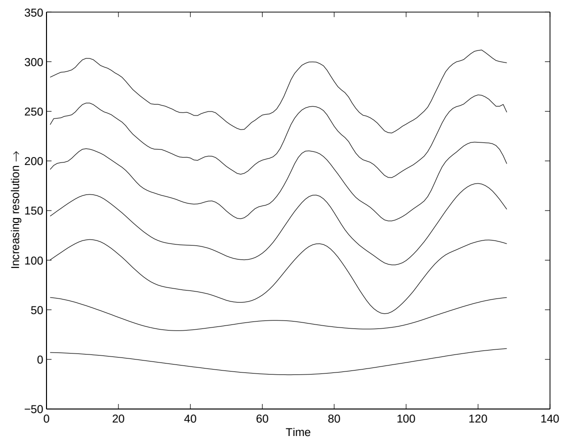
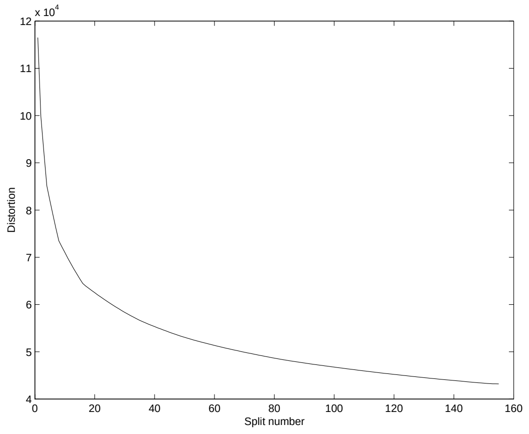
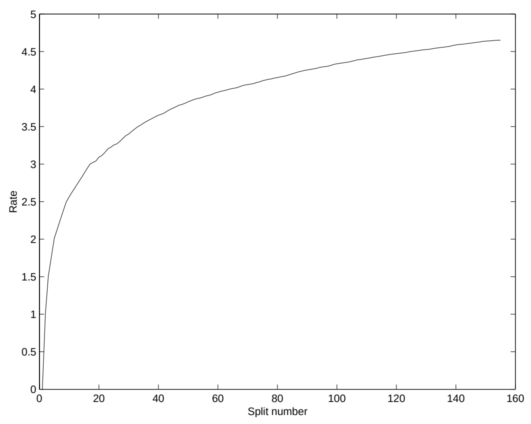


Figure 6.1: Signal in 7 resolutions

Fig. 6.2(a) shows the decrease in distortion as the number of splits increase while Fig. 6.2(b) shows the increase in rate as the tree grows. Note that distortion decreases to zero in the asymptotic case while rate goes to infinity as was shown in Chapter 4.



(a) Distortion vs. split number



(b) Rate vs. split number

6.2 Simulation for classification

We used the greedy tree growing algorithm to create a tree structured classifier for multi-resolution data. Each signal can be expressed as

$$s(t) = \sum_{k=1}^{K} s_k(t)$$

where each $s_k(t)$ is re-sampled from a d_k dimension random vector x_k. The Matlab command "resample" was used for this purpose. The random vector x_k was produced from a Gaussian mixture density for either class. The distance between the mean of x_k for class 1 and 2 increases as k increases. This results in a signal $s(t)$ that has increasing information relevant to the classification as the resolution increases. The Daubechies 8th order wavelet was used to analyze the data into 5 resolutions. The tree growing algorithm used LVQ to split each node. At each iteration, the node that offered the biggest reduction in the classification error was chosen to be split. Fig. 6.2 shows the decrease in the Bayes risk as the number of splits increase.

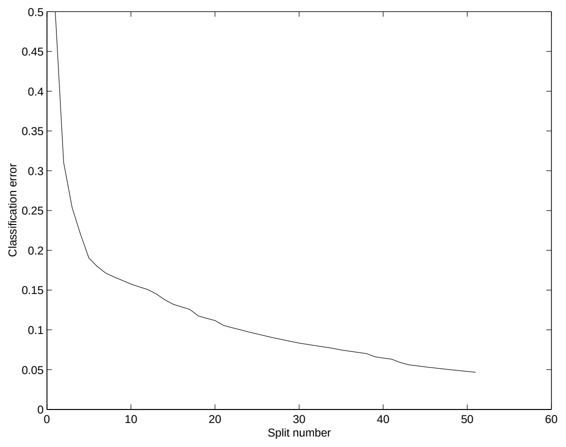


Figure 6.2: Classification error vs. split number

6.3 Yeast gene clustering

An application of un-supervised clustering for clustering yeast (*Saccharomyces Cervisiae*) genes according to their expression profiles is presented in this section. This application is motivated by the recent advances in gene analysis techniques that have the potential to measure the expressions of thousands of genes at the same time [46] [40]. Since each cellular process is the result of several hundred genes acting in concert, such techniques have expanded our ability to study the cell at an unprecedented level of detail.

The inevitable consequence of this detail is the flood of data that is produced by techniques such as micro-arrays. There is an urgent need for advanced methods for organizing and displaying this data in a manner that is useful and intuitive for biologists. One possible way of organizing is to group together the genes that are expressed similarly in similar environments.

Since most cellular processes are step-by-step processes [28] where the products of one step initiate and control the activity in the next step, one would expect that genes that regulate each step would be expressed together. Thus genes that belong to a common metabolic pathway would be highly correlated in their expression. This in turn, means that clustering genes according to their expressions in different environments would group together genes that have similar function.

The data, obtained from [14], consists of the log ratio of the expression levels of 2467 genes during 8 experiments. For each experiment, expression levels are measured at different time points (differing number of time points for each experiment) giving a 79-dimensional vector of expression data for each experiment and time point.

Eisen *et.al.* have used a pair-wise, bottom up approach to cluster yeast genes in [15]. Initially, all genes belong to individual cells. Then the two closest genes (in terms of a distance function) are chosen and their cells are merged and replaced by the average of the two genes. Then this process is applied repeatedly until all cells have been merged into the root node. The result is then displayed as a tree and genes relating to similar functions are shown to cluster together.

We will use a similar clustering method to reveal similarities of expression in genes of similar functional origin. However our method will be top down, in that we start out with a single cluster containing all the vectors which is then split successively to create the tree. In addition, a multi-resolution analysis is used to represent the expression vector for each gene in different levels of detail. The higher splits in the tree are done on the basis of lower resolution vectors and additional

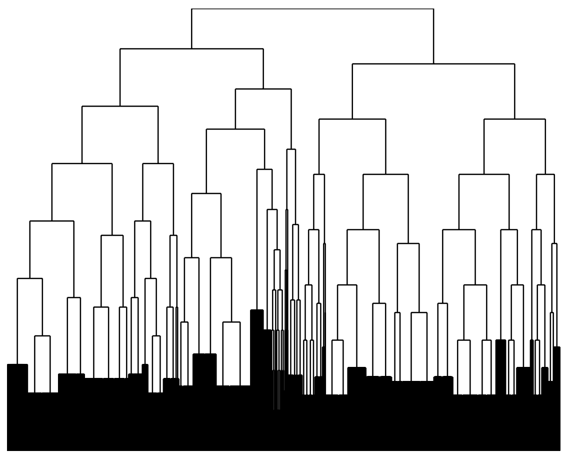


Figure 6.3: Clustering tree for yeast gene expression profiles

resolution is used when needed in the subsequent splits.

Representing the expression profile of each gene in different resolutions has the advantage that we can easily see those experiments whose expressions are crucial for clustering genes similar in certain ways. A low resolution representation of the expression profile gives a broad view of the expression of the gene while details might be needed for better discrimination between genes with similar functions.

6.3.1 Data analysis methods and results

The 79 dimensional expression vector for each gene was represented in 6 resolutions using a Daubechies 4th order wavelet filter. The tree growing algorithm is implemented as discussed in Chapter 4. A large enough rate constraint was put so that each leaf node ends up with a relatively small number (less than 10) number of vectors. Fig. 6.3 shows that structure of the resulting tree.

To study the clustering of genes according to function, we plot only those nodes in the tree that contain at-least one gene of the particular function. For example,

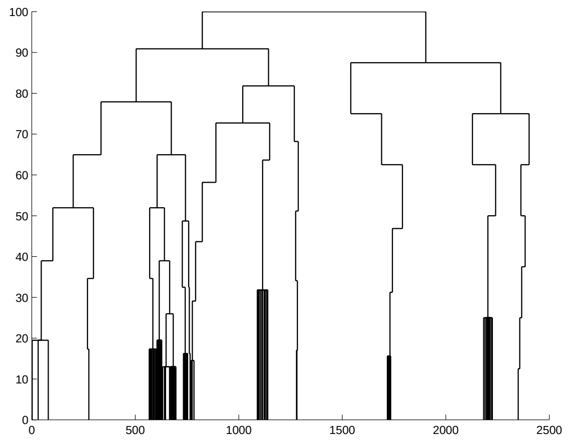


Figure 6.4: Occurences of ribosomal genes in clustering tree

Fig. 6.4 shows all genes that contribute to ribosomal function. From these plots we can notice some significant features.

Fig. 6.4 shows that ribosomal genes are very strongly correlated in their expression. Furthermore, it is apparent that there are five distinct clusters of such genes. Genes in the same cluster behave similarly to each other, but each cluster is distinctly different from another cluster.

Another kind of clustering behavior is noticable when we look at Fig. 6.5 which shows genes that have functions related to protein synthesis. Here we see that there are several strong and weak clusters along with many genes that are not part of any cluster. This is the kind of clustering that is mostly seen when looking at functions that occur under many different environmental conditions.

Of course, there are some classes of genes that do not cluster at all. Fig. 6.6 shows genes that have a Helix-Turn-Helix structure. Such structural features are not expected to show themselves in the expression profiles and are therefore hard to cluster.

In this section we have shown how un-supervised clustering can be used to

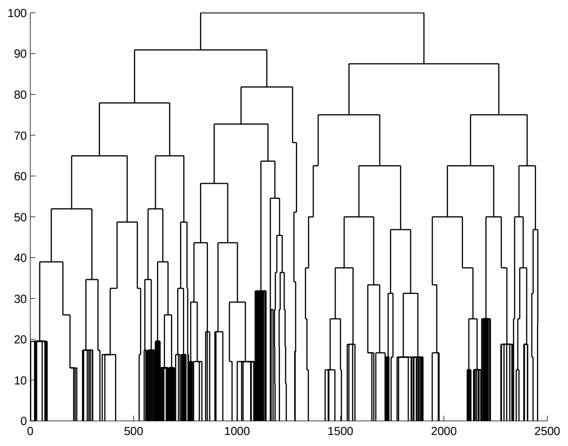


Figure 6.5: Occurrences of protein synthesis genes in clustering tree

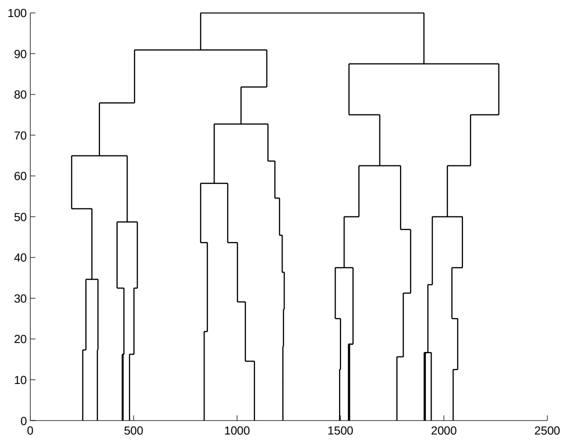


Figure 6.6: Occurrences of helix-turn-helix genes in clustering tree

analyze and present data from gene expression experiments to display how genes are organized according to function. Such methods will be of great importance in the future to deal with the flood of information expected to be available from current biological experimental tools.

6.4 Lymphoma prediction using tree classifiers

In this section we present an application of tree structured classifiers for lymphoma prediction. Alizadeh *et al* [17] have explored the use of gene expression analysis for prediction of cancerous cells. We will use the same dataset and use tree structured classifiers with multi-resolution analysis for classifying cancerous from non-cancerous cells.

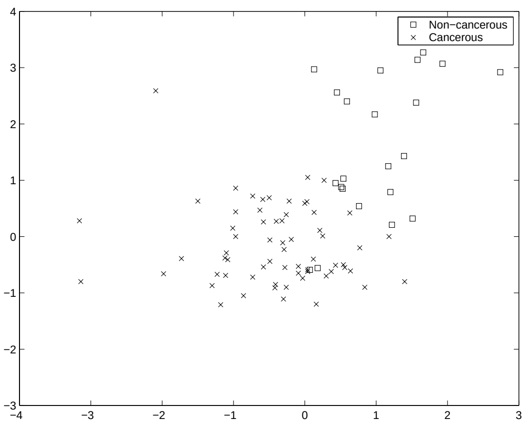
We have the expressions of 4096 genes from 98 different cell types. Of these 98, 72 are cancerous while 26 are non-cancerous. We are interested in finding out which genes are most predictive of lymphoma through their expressions.

To rank gene expressions according to their discriminative power, we use the *Fischer discriminant* [19]. For scalar observations x, Fischer proposed the following measure of separation between observations for class 1 from class 2

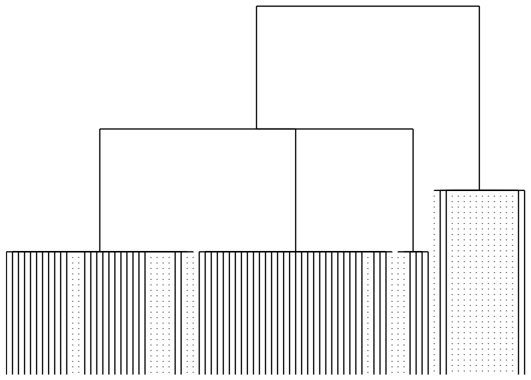
$$F = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2}$$

where μ_1, μ_2 are the means of the observations belonging to class 1 and class 2 respectively and σ_1, σ_2 the variances. The larger the value of F, the more separated are the probability densities of the two classes in that feature space.

We order the 4096 genes according to the Fischer discriminant. The best 6 genes are chosen and a k-th order multi-resolution representation is created by taking the vector composed of the expressions of the best k genes. Thus, from the list of gene expressions sorted according decreasing Fischer index, we choose the first k genes and concatenate them into a k resolution representation. As k increases, we append genes one after another from the list. Fig. 6.7(a) shows the points in the two classes in the space of the best two features. A classification tree is created by using LVQ to split each node in a greedy fashion. Since we have all the data on hand, we need not use the online algorithm as presented in Chapter 5. Fig. 6.7(b) shows an example of a tree created by using the greedy algorithm. A 10-fold cross validation was performed to find the average error. We summarize



(a) Distribution of the two classes according to best two genes



(b) Example of tree classifier from greedy growing algorithm. Solid lines are cancerous, dotted non-cancerous

the result in Table 6.1. LVQ is a flat (non-hierarchical) LVQ classifier using all the expressions from all the genes and SVM is a Support Vector Machine classifier.

Table 6.1: Average prediction error on lymphoma data

Type of classifier	Avg. error on test set
MRTSVQ	0.085
LVQ*	0.021
SVM*	0.0104

* Bhamidipati [4]

6.5 Application to wear prediction

Trying to predict the wear of a tool from the sound (equivalently, the vibrations) it makes is useful in real-time monitoring of machinery to detect faults as and when they occur, rather than wait until the next maintenance period. This way, unnecessary maintenance, as well as long runs in a faulty condition, can be avoided. In the case of a cutting tool, trying to cut with a blunt tool can lead to the breakage of the tool and degradation of the job, while pulling the tool off for frequent assessments are expensive in terms of the machinist's time.

We use two auditory filters, developed by Shamma et.al., for preprocessing [47], [48]. The first one is a model of the filter banks and nonlinear operations that take place in the inner ear [48]. The second filter mimics the analysis of the filtered signal that take place in the primary auditory cortex and has been described in detail in Chapter 3.

6.5.1 Inner Ear

This filter (Fig 6.7) describes the mechanical and neural processing in the early stages of the auditory system. In the Analysis Stage, a bank of constant-Q filters, approximates the function of the eardrum and the basilar membrane in the cochlea

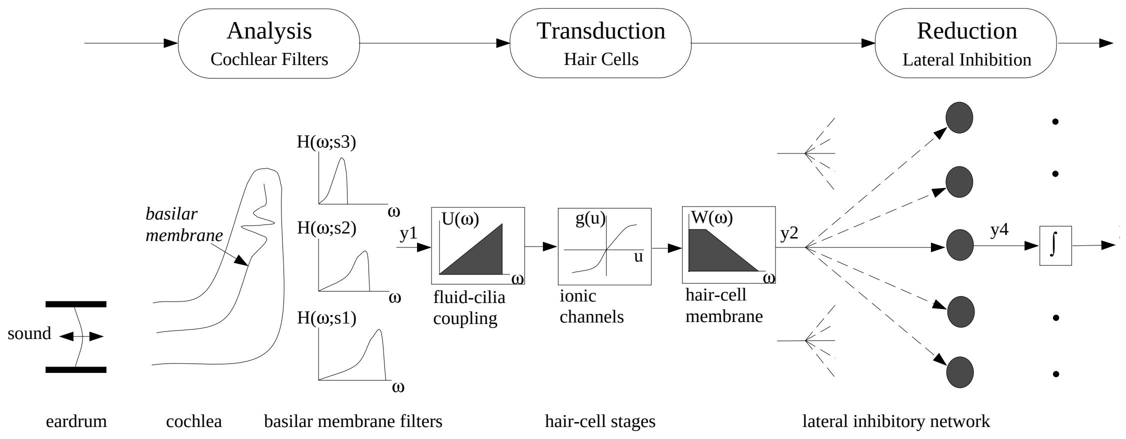


Figure 6.7: Spectral processing of sound stimuli in the inner ear

with the continuous spatial axis of the cochlea as the scale parameter. Another way to interpret the output of the cochlear filters is as an affine wavelet transform of the stimulus. The Transduction Stage models the conversion of the mechanical displacements in the basilar membrane into electrical activity along a dense, topographically ordered array of auditory nerve fibers. This conversion can be well modeled by a three-stage process consisting of

1. a velocity coupling stage (time derivative),

2. an instantaneous non-linearity describing the opening and closing of the ionic channels and

3. a low-pass filter with a relatively short time constant to describe the ionic leakage through the hair cell membranes.

The third stage called the Reduction Stage effectively computes an estimate of the spectrum of the stimulus, through a *lateral inhibitory network* (LIN). The details can be found in [48].

The output from this filter is analyzed into multiple resolutions by the filter that models the primary auditory cortex.

6.5.2 Description of the data

The data-set consists of accelerometer readings from the spindle head for three cases.

1. 0.5"dia. mill cutting a steel job,
2. 0.5"dia. mill cutting a titanium job and
3. 1.0"dia. mill cutting a steel job.

In each case, the speed of the mill is different, varying from 344 rpm for case 3 to 733 rpm for case 2. All three cases differ widely in the character of sound as well as behavior over short and long time scales. Case 1 is rather well behaved, with increase in wear leading to a gradual change in the frequency characteristics of the sound. Indeed, one can find harmonics around 8kHz that increase in power, more or less monotonically, as the tool life increases.

Cases 2 and 3 are much less easily analyzable. They show episodes of high-wear-rate when the sound character markedly changes, interspersed through periods of quiet cutting when the tool seems to behave ideally. There are no easy pointers like the 8kHz harmonic in Case 1.

The sound sample from each pass of each tool also includes a sync file, which gives the beginning of each revolution of the tool in terms of sample numbers. Also 2-3 wear measurements are given for the lifetime of each tool.

6.5.3 Preprocessing and training

The sound samples from the accelerometer were cut up into frames, each frame being the sound from one revolution of the tool. Each such frame was re-sampled to 4096 samples, normalized to zero mean and unit variance, and further subdivided into four sub-frames each. Each such sub-frame would correspond to one geometric quarter of the tool, or one flute.

All sub-frames are then passed through the inner ear and auditory cortex filter to obtain a set of multi-resolution vectors describing the timbre spectrum of the sound. We use an average of four sub-frames (each belonging to one revolution) as our observation vector. These vectors were then used as input to the tree growing algorithm.

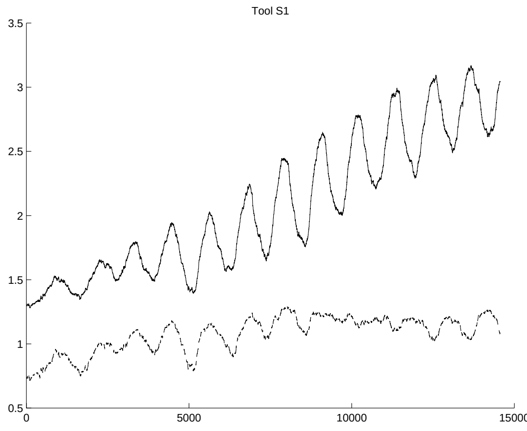
We utilize the class labels in growing the TSVQ, by building a tree for each class, using only the appropriately labeled data. This method, usually called Parallel TSVQ, gives better results than making one tree for all the classes combined. In the combined tree, an initial wrong misclassification into one particular sub-tree can end in a vector being incorrectly classified. This problem is avoided, to a great extent, in the parallel case. The Parallel TSVQ is also quicker to execute when we have a large number of classes. Testing on each tree can be done in parallel, which reduces computational time.

The tree growing algorithm we used is similar to the algorithm used in [3] that we have described earlier. This is essentially similar to the algorithm we have presented in this thesis except that the stopping criterion is a constraint on the number of leaves on the tree rather than the rate of the tree.

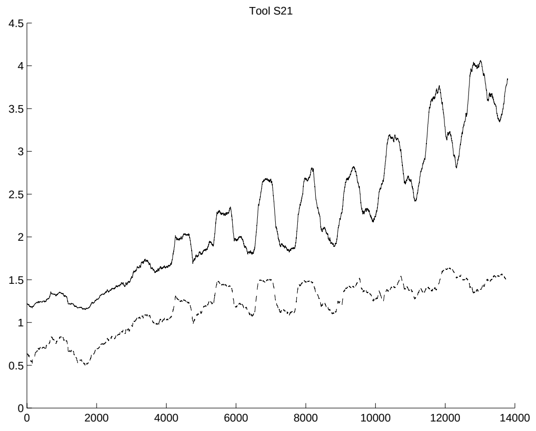
6.5.4 Testing

Testing was done on data belonging to all three cases. Tools that had not been used in training were used in the testing procedure. The preprocessing was similar to what was done for training. Each vector was dropped down all five trees and the distance to the centroids of the leaf nodes it fell into, was compared. The vector is assigned a wear-class according to the wear level of the tree that gives the least distance from the centroid to the vector. This way, we get a time series of wear-class prediction for all the frames for all the passes. Next we take a sliding window of 500 frames and find the mean wear estimate for this window. This gives an estimate of the changing wear at different times.

For the case of a 0.5" dia. tool cutting 4340 steel, the plots of mean wear estimates vs. tool life is shown in Figs 6.8(a) and 6.8(b), for different tools. It is apparent that our method has picked up features in the sound that seem to be correlated to the tool-life and the wear of the tool. The periodic variation in the



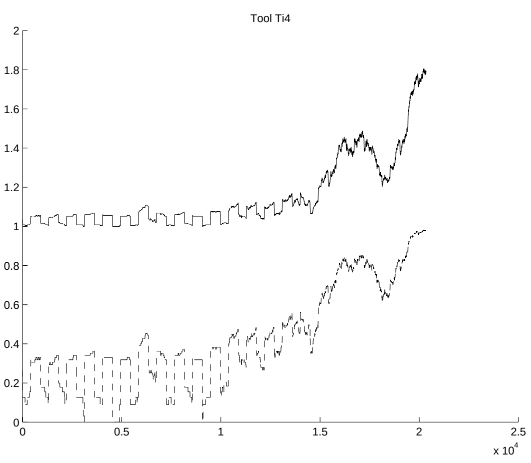
(a) Average (solid) and variance (dashed) of wear vs. tool life for S1



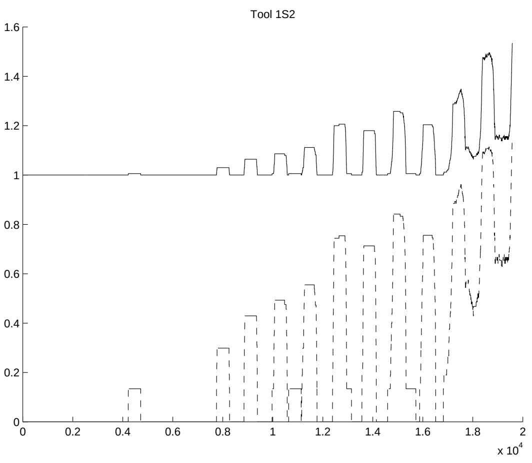
(b) Average (solid) and variance (dashed) of wear vs. tool life for S21

wear estimate is a result of the different passes. The actual sound made by the tool is not just a function of the state of the tool, but also depends on the position on the job the tool is presently at. The wear estimate at the start and end of the pass are higher than that in the middle. This could correspond to the observation that the *wear rate* at the starting and ending of the job is higher than in the middle. A tool wear model incorporating wear information from the sound is used for wear prediction in [44].

Fig 6.8(c) shows the results of testing a Case 2 (0.5" dia. tool cutting titanium job) tool data on the tree trained using Case 1 data only. Here also we see the gradual increase in the tool wear, though the way it increases is different from that in Case 2. Fig6.8(d) shows the results of the classification on a tool from Case 3 (1.0" dia. tool cutting 4340 steel).



(c) Average (solid) and variance (dashed) of wear vs. tool life for Ti4



(d) Average (solid) and variance (dashed) of wear vs. tool life for 1S2

6.6 Fingerprint identification

In this section we will investigate the problem of storing fingerprint images for fast retrieval and identification. A set of fingerprints takes about 10 megabytes of storage in the raw form. The FBI has roughly 200 million fingerprints on record and storing all of them in the raw form would require around 200 terabytes of disk-space. In addition, whenever a new fingerprint is obtained, it has to be compared with each print in the archive to find out if there is a match.

The Wavelet/Scalar Quantizer (WSQ) standard [6] was designed for lossy compression of fingerprints for archival and transmission purposes. It uses a wavelet preprocessor followed by scalar quantization and Huffman coding to reduce the data by approximately 1:12. When a new fingerprint is obtained, it is compared with all the decoded fingerprints in storage and a match is found.

Much of the time taken in identifying a new fingerprint is the processing time associated with uncompressing each fingerprint in the database and comparing it to the new print. It is our contention that this step is unnecessary and leaving the prints in the multi-resolution domain will speed up the search process in two ways

1. Computation required for uncompressing each print is saved.
2. Multi-resolution tree structured arrangement of prints will decrease search time to less than $O(\log(k))$ where k is the number of prints in the database.

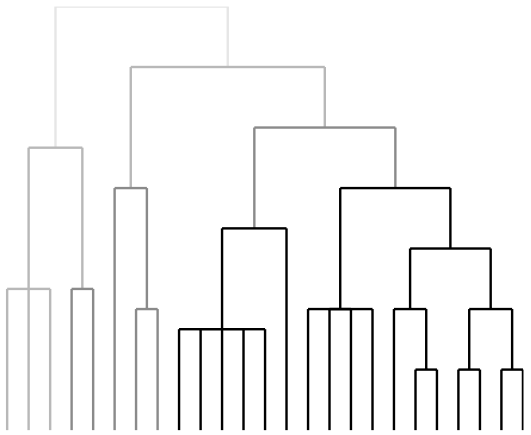
To illustrate these advantages, we store 25 fingerprints from the NIST 8-Bit Gray Scale Images of Fingerprint Image Groups (FIGS) sample set [18] using an MRTSVQ. We show that the number of computations needed to identify a given fingerprint is reduced considerably as compared to a brute force search. This process is called *authentication* since we know that we already have the test fingerprint in our database and all we want is to search for it in the fastest possible way.

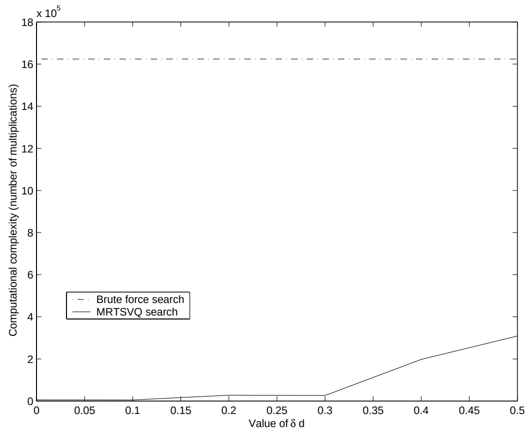
There is also the complimentary problem of *verification* where we have an unknown test fingerprint and we want to see if we have this print in our database. This problem is harder than authentication because we will need to extract features like *loops, whorls, arches* and *minutiae* from the test print to find the fingerprint in our database that it most closely corresponds to. This problem has been addressed in [23] and [27]. Since extraction of these features is, technically speaking, not related to classifier design, we will not deal with that problem here.

6.6.1 Fingerprint encoding and tree growing

We use a Bi-orthogonal wavelet filter with 3rd order reconstruction filter and 7th order decomposition filter ('`bior3.7`' in MATLAB) to represent each fingerprint in a multi-resolution form. There are 6 resolutions and at the lowest resolution, each image is represented by a set of 484 coefficients while at the highest resolution the dimension is 64961. The original size of each fingerprint is 480×512 pixels with 8-bit gray level. Tree growing is accomplished according to the algorithm given in Chapter 4. The decision to split at a higher level is taken when the fractional decrease in distortion of the cell is lesser than a value δd that is fixed before executing the algorithm. Larger values of δd force the algorithm to seek higher resolutions early on in the tree growing process, thus increasing the computational complexity of the tree search. Lower values allow the algorithm to remain in the low resolution space for a longer time, resulting in trees with lower computational complexity for search operations.

Fig. 6.8(e) shows an example of a tree generated by the greedy algorithm. Darker branches correspond to higher resolutions.


(e) Tree-structured organization of fingerprint data: Darker branches are at higher resolution



(f) Computational complexity vs. δd

Defining the computational complexity of the tree as the average number of multiplications for distance computation Fig. 6.8(f) shows the experimental computational complexity for each value of δd. The complexity increases sharply with δd but is much smaller than the complexity of the brute force search.

Chapter 7

Conclusions and further research

7.1 Conclusions

In the previous chapters we have presented algorithms for achieving compression and classification of signals represented in multiple resolutions. We have presented a greedy algorithm for quantization of multi-resolution signals and showed that under some conditions the expected distortion will go to zero asymptotically as the number of iterations increase to infinity. The condition for this is that the dimensionality of the MRA should not increase faster than the logarithm of the depth. We have also shown that under a rate constraint, the algorithm will stop in finite time.

We have derived an online algorithm for constructing quantizers and shown that it minimizes a cost that is exactly the distortion error.

For classification we have introduced an extension of LVQ for creating a multi-scale tree. This algorithm adapts the centroids of the tree as a whole to adapt to a sequence of training samples of observation vectors and class labels. We have shown that such an algorithm will converge to an equilibrium solution that is the equilibrium for an associated ODE.

Tree structured classification and clustering methods have wide applicability in most problems where the relationships between data-points have a hierarchical structure. This is more general than appears at first sight. In most problems, we implicitly assume that data-points lying near each other are more closely related than points lying far from each other. Thus, if we consider a sequence of nested neighborhoods of a point, there is a hierarchy of relationships between it and a point in a neighborhood. Thus hierarchical relationships are very common in real world data. [43] explains in more detail why it is reasonable for most data to have a hierarchical structure.

7.2 Future research

7.2.1 Combined compression and classification

Combining the compression and classification criteria is motivated by the necessary tradeoff between the contrary requirements for good compression compared to good classification performance. An example that we considered in the previous chapter is fingerprint archival and classification. Storing fingerprints in the raw form takes terabytes of storage so we would be interested in finding ways of compressing these images. But we cannot compress them so much that we lose vital information necessary for discriminating between different prints.

This example is typical of applications where we not only need to reduce the amount of storage required but also do the compression in a way that does not lose the information necessary for classifying a new observation. The first formulation of this problem and its solution by VQ was by Perlmutter *et. al.* [36]. They use the

Nearest Neighbor criterion combined with majority rule and a generalize centroid to develop an iterative algorithm that seeks to minimize a weighted combination of the classification error and the distortion.

Baras and Dey [2] present an LVQ-like algorithm for online minimization of a combined classification and compression cost. They prove that this algorithm converges to the equilibrium points of an associated ODE.

It has not yet been shown whether the distortion and classification errors go to the minimum possible as the number of centroids increase. Also there have been no results forthcoming on how changes in the relative weighting between the classification error and the compression error changes the asymptotic behavior of the algorithm.

7.2.2 Convergence of LVQ for local equilibria

In Chapter 5 we have shown how the multi-scale LVQ algorithm converges to the global equilibrium of the associated ODE. A far more realistic assumption is that there are more than one non-global equilibria. In such a case, our results do not hold.

One way to solve this would be to use a Ljung-type convergence theorem that says that if the centroids visit the domain of attraction of a local, asymptotically stable equilibrium an infinite number of times then the algorithm will converge to that equilibrium.

Appendix A

Geometric result

In Chapter 4 we used the following geometric result,

Lemma 14 *Consider two concentric balls $B_1 = B(0, r)$ and $B_2 = B(0, r\sqrt{d})$ in a d-dimensional space. Now take a hyper-plane H in this space that is tangent to the inner ball B_1 and intersects the outer ball in a hyper-circle, thus dividing the surface of the outer ball into two parts, V_1 and V_2 with $Vol(V_1) < Vol(V_2)$. Then, we have*

$$\frac{Vol(V_1)}{Vol(V_1) + Vol(V_2)} > \frac{1}{2\sqrt{2\pi e}} \forall d$$

Proof: Fix a dimension d. The surface area of a sphere in d dimensions is

$$\frac{\frac{\pi}{2}^{\lfloor \frac{d}{2} \rfloor} 2^d r^{d-1}}{(d-2)(d-4)\ldots(1 \text{ or } 2)} \tag{A.1}$$

The hyper-plane H intersects B_2 in a hyper-circle such that the surface area of the sphere enclosed within this hyper-circle is the same as that enclosed by a cone with vertex at the center of the sphere and angle $\alpha = \cos^{-1}(1/\sqrt{(}d))$ (see Fig. A.1). This surface area can be computed as (omitting tedious details)

$$\frac{(2r)^{d-1} \frac{\pi}{2}^{\lfloor \frac{d-1}{2} \rfloor}}{(d-3)(d-5)\ldots} I_{d-2}(\cos^{-1}(1/\sqrt{(}d))) \tag{A.2}$$

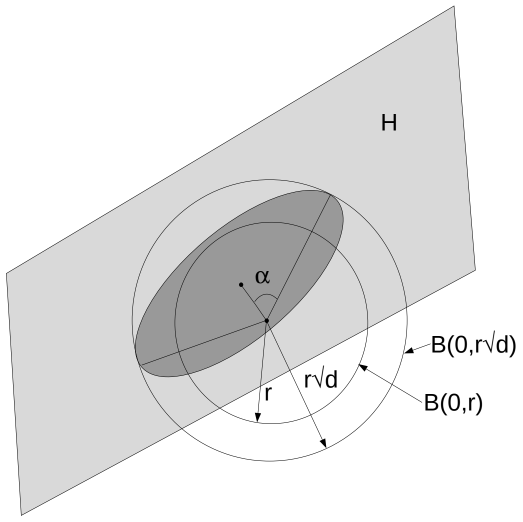


Figure A.1: Illustration of B_1, B_2 and H in 3-dimensional space

where $I_k(\alpha) = \int_0^\alpha \sin^k(x)dx$

Taking the ratio of the surface enclosed by the cone A.2 to the total area A.1 we get

$$\begin{aligned} R(d) &= \frac{1}{\pi}\cos^{-1}\left(\frac{1}{\sqrt{d}}\right) - \frac{\sqrt{d-1}}{\pi d}\left[1 + \frac{2}{3}\left(\frac{d-1}{d}\right) + \frac{2\times 4}{3\times 5}\left(\frac{d-1}{d}\right)^2 + \dots \right. \\ & \left. + \frac{2\times 4\times\dots\times(d-4)}{3\times 5\times\dots\times(d-3)}\left(\frac{d-1}{d}\right)^{\frac{d-4}{2}}\right] \end{aligned} \tag{A.3}$$

if d is even and

$$\begin{aligned} R(d) &= \frac{1}{2} - \frac{1}{2\sqrt{d}}\left[1 + \frac{1}{2}\left(\frac{d-1}{d}\right) + \frac{1\times 3}{2\times 4}\left(\frac{d-1}{d}\right)^2 + \dots \right. \\ & \left. + \frac{1\times 3\times\dots\times(d-4)}{2\times 4\times\dots\times(d-3)}\left(\frac{d-1}{d}\right)^{\frac{d-3}{2}}\right] \end{aligned} \tag{A.4}$$

if d is odd.

$R(d)$ is monotonically decreasing for increasing d, so $R(d') > \lim_{d\to\infty} R(d)$ for any d. Now we will find a lower bound on $\lim_{d\to\infty} R(d)$

It is enough to find the above limit for d taking only odd values. A.4 can be written as

$$R(d) = \frac{1}{2} - \frac{1}{2\sqrt{d}} S(d)$$

where

$$S(d) = 1 + \frac{1}{2}\left(\frac{d-1}{d}\right) + \frac{1\times 3}{2\times 4}\left(\frac{d-1}{d}\right)^2 + \ldots + \frac{1\times 3\times \ldots \times (d-4)}{2\times 4\times \ldots \times (d-3)}\left(\frac{d-1}{d}\right)^{\frac{d-3}{2}}$$

We can write $S(d) = S_1(d) - S_2(d)$ where

$$S_1(d) = 1 + \frac{1}{2}\left(\frac{d-1}{d}\right) + \frac{1\times 3}{2\times 4}\left(\frac{d-1}{d}\right)^2 + \ldots + \frac{1\times 3\times \ldots \times (i-4)}{2\times 4\times \ldots \times (i-3)}\left(\frac{d-1}{d}\right)^{\frac{i-3}{2}} + \ldots$$

for an infinite number of terms and

$$S_2(d) = \frac{1\times 3\times \ldots \times (d-2)}{2\times 4\times \ldots \times (d-1)}\left(\frac{d-1}{d}\right)^{\frac{d-1}{2}} + \frac{1\times 3\times \ldots \times (d)}{2\times 4\times \ldots \times (d+1)}\left(\frac{d-1}{d}\right)^{\frac{d+1}{2}} + \ldots$$

$S_1(d)$ can be shown to be the Taylor expansion of

$$\left(1 - \frac{d-1}{d}\right)^{-1/2} = \sqrt{d}$$

while $S_2(d)$ can be written as

$$\begin{aligned}
&\frac{(d-1)!}{((\frac{d-1}{2})!)^2 2^{d-1}}\left(\frac{d-1}{d}\right)^{\frac{d-1}{2}}\left(1 + \frac{d}{d+1}\left(\frac{d-1}{d}\right) + \frac{d(d+2)}{(d+1)(d+3)}\left(\frac{d-1}{d}\right)^2 + \ldots\right) \\
&> \frac{(d-1)!}{((\frac{d-1}{2})!)^2 d^{d-1}}\left(\frac{d-1}{d}\right)^{\frac{d-1}{2}}\left(1 + \frac{d}{d+1}\left(\frac{d-1}{d}\right) + \left(\frac{d}{d+1}\right)^2\left(\frac{d-1}{d}\right)^2 + \ldots\right) \\
&= \frac{(d-1)!}{((\frac{d-1}{2})!)^2 d^{d-1}}\left(\frac{d-1}{d}\right)^{\frac{d-1}{2}}\left(\frac{1}{1-\frac{d-1}{d+1}}\right) \\
&= \frac{(d-1)!}{((\frac{d-1}{2})!)^2 d^{d-1}}\left(\frac{d-1}{d}\right)^{\frac{d-1}{2}}\frac{(d+1)}{2}
\end{aligned}$$

where the inequality comes from the fact that $(d+i)/(d+i+1) > d/(d+1)$ for any $i, d > 0$

Using Stirling's formula

$$n! \approx \sqrt{\frac{2\pi}{n+1}} e^{-(n+1)}(n+1)^{n+1}$$

for the factorial of a sufficiently large integer n, we can simplify the lower bound on $S_2(d)$ as

$$S_2(d) > \sqrt{\frac{1}{2\pi}} \frac{e(d+1)}{\sqrt{d}} \left(\frac{d}{d+1}\right)^d \left(\frac{d-1}{d}\right)^{\frac{d-1}{2}}$$

Then

$$R(d) > \frac{1}{2} - \frac{1}{2\sqrt{d}} \left(\sqrt{d} - \sqrt{\frac{1}{2\pi}} \frac{e(d+1)}{\sqrt{d}} \left(\frac{d}{d+1}\right)^d \left(\frac{d-1}{d}\right)^{\frac{d-1}{2}}\right)$$

which simplifies to

$$R(d) > \frac{e}{2\sqrt{2\pi}} \frac{\sqrt{(1-1/d)^d}}{\sqrt{(1-1/d)}} \frac{1}{(1+1/d)^d} \left(1 + \frac{1}{d}\right)$$

Using the fact the $(1-1/d)^d \to e^{-1}$ and $(1+1/d)^d \to e$ as $d \to \infty$ we get

$$\lim_{d\to\infty} R(d) > \frac{1}{2\sqrt{2\pi e}}$$

which gives the desired result.

BIBLIOGRAPHY

[1] J. Ambrose-Ingerson, R. Granger, and G. Lynch. "Simulation of paleocortex performs hierarchical clustering". *Science*, 247:1344–1348, March 1990.

[2] J. S. Baras and S. Dey. "Combined compression and classification with Learning Vector Quantization". *IEEE Transactions on Information Theory*, 45(6):1911–1920, September 1999.

[3] J. S. Baras and S. I. Wolk. "Wavelet based hierarchical organization of large image databases: ISAR and face recognition". In *Proc. SPIE 12th International Symposium on Aerospace, Defence Sensing, Simulation and Control*, volume 3391, pages 546–558, April 1998.

[4] P. Bhamidipati. "Clustering algorithms for microarray data mining". Master's thesis, University of Maryland at College Park, 2002.

[5] V. S. Borkar. "Stochastic approximation with two time scales". *System and Control Letters*, 29:291–294, 1997.

[6] J. Bradley, C. Brislawn, and T. Hopper. "The FBI Wavelet/Scalar Quantization Standard for gray-scale fingerprint image compression". In *Proc. SPIE*, volume 1961, pages 293–304, 1993.

[7] L. Breiman. *"Probability Theory"*. Classics in applied mathematics. SIAM, Philadelphia, 1992.

[8] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *"Classification and Regression Trees"*. Wadsworth, Belmont, CA, 1984.

[9] P. A. Chou. "Optimal partitioning for classification and regression trees". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4), April 1991.

[10] R. R. Coifman and M. V. Wickerhauser. "Entropy-based algorithms for best-basis selection". *IEEE Transactions on Information Theory*, 38:713–718, 1992.

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

[12] I. Daubechies. *"Ten lectures on Wavelets"*. Number 61 in CBMS-NSF Series in Applied Mathematics. SIAM, Philadelphia, 1992.

[13] M. Effros. "Practical multi-resolution source coding: TSVQ revisited". In *Data Compression Conference*, pages 53–62, 1998.

[14] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. `http://genome-www.stanford.edu/clustering/`.

[15] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. "Cluster analysis and display of genome-wide expression patters". *Proceedings of the National Academy of Science, USA*, 95:14863–14868, Dec 1998.

[16] W. H. R. Equitz and T. M. Cover. "Successive refinement of information". *IEEE Transactions on Information Theory*, 37(2):269–275, March 1991.

[17] A. A. Alizadeh et al. "Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling". *Nature*, 403:503–511, Feb 2000.

[18] NIST fingerprint database. `http://www.nist.gov/srd/nistsd4.htm`.

[19] R. Fischer. "The case of multiple measurements in taxonomic problems". *Annals of Eugenics*, 7(II):179–188, 1936.

[20] D. Gabor. "Theory of communication". *Journal of the IEE*, 93:429–457, 1946.

[21] J. C. Goswami and A. K. Chan. *"Fundamentals of Wavelets: Theory, Algorithms and Applications"*. John Wiley and Sons, Inc, New York, 1999.

[22] A. Haar. "Zur Theorie der orthogonalen Funktionensysteme". *Mathematische Annalen*, 69:331–371, 1910.

[23] K. Karu and A. K. Jain. "Fingerprint classification". *Pattern Recognition*, 29(3):389–404, 1996.

[24] T. Kohonen. *"Self organization and associative memory"*. Springer-Verlag, Berlin, 1989.

[25] H. J. Kushner and G. G. Yin. *"Stochastic approximation algorithms and applications"*. Stochastic modelling and applied probability. Springer, New York, 1997.

[26] A. LaVigna. *"Nonparametric classification using Learning Vector Quantization"*. PhD thesis, University of Maryland at College Park, 1989.

[27] H. C. Lee and R. E. Gaensslen, editors. *"Advances in fingerprint technology"*. Elsevier, New York, 1991.

[28] B. Lewin. *"Genes"*. John Wiley and sons, New York, third edition, 1987.

[29] Y. Linde, A. Buzo, and R. M. Gray. "An algorithm for vector quantizer design". *IEEE Transactions on Communications*, COM–28:84–95, 1980.

[30] S. P. Lloyd. "Least squares quantization in PCM". *Bell Laboratories Technical Note*, 1957.

[31] J. Max. "Quantizing for minimum distortion". *IRE Transactions on Information Theory*, IT-6:7–12, March 1960.

[32] G. F. McLean. "Vector quantization for texture classification". *IEEE Transactions on Systems, Man and Cybernetics*, 23(3):637–649, May/June 1993.

[33] Y. Meyer. *"Wavelets:Algorithms and applications"*. SIAM, Philadelphia, 1993.

[34] A. B. Nobel. "Recursive partitioning to reduce distortion". *IEEE Transactions on Information Theory*, 43(4):1122–1133, July 1997.

[35] A. B. Nobel and R. A. Olshen. "Termination and continuity of greedy growing for tree-structured vector quantizers". *IEEE Transactions on Information Theory*, 42(1):191–205, January 1996.

[36] K. O. Perlmutter, S. M. Perlmutter, R. M. Gray, R. A. Olshen, and K. L. Oehler. "Bayes risk weighted vector quantization with posterior estimation for image compression and classification". *IEEE Transactions on Image Processing*, 5(2):347–360, February 1996.

[37] J. Puzicha, T. Hofmann, and J. M. Buhmann. "A theory of proximity based clustering: Structure detection by optimization". *Pattern Recognition*, 4(33):617–634, 1999.

[38] K. Ramachandran and M. Vetterli. "Best wavelet packet bases in a rate distortion sense". *IEEE Transactions in Image Processing*, 2(2):160–175, April 1993.

[39] B. Rimoldi. "Successive refinement of information: Characterization of the achievable rates". *IEEE Transactions on Information Theory*, 40(1):253–266, January 1994.

[40] M. Schena, D. Shalon, R. W. Davis, and P. O. Brown. "Quantitative monitoring of gene expression patterns with a complementary dna microarray". *Science*, 270(5235):467–470, October 1995.

[41] S. A. Shamma. "The acoustic features of speech phonemes in a model of the auditory system: Vowels and unvoiced fricatives". *Journal of Phonetics*, 16:77–91, 1988.

[42] C. E. Shannon. "A mathematical theory of communication". *Bell Sys. Tech. Journal*, 27:379–423,623–656, 1988.

[43] S. A. Starks, V. Kreinovich, and A. Meystel. "Multiresolution data processing: It is necessary, it is possible, it is fundamental". pages 22–25, Gaithersburg, MD, USA, September 1997.

[44] S. Varma and J. S. Baras. "Tool wear estimation from acoustic emissions: A model incorporating wear rate". In *International Conference on Pattern Recognition*, Quebec City, Quebec, Aug 2002.

[45] S. Varma, J. S. Baras, and S. A. Shamma. "Biologically inspired acoustic wear analysis". In *Nonlinear Signal and Image Processing Conference*, Baltimore, MD, June 2001.

[46] V. E. Velculescu, L. Zhang, B. Vogelstein, and K. W. Kinzler. "Serial analysis of gene expression". *Science*, 270(5235):484–487, October 1995.

[47] K. Wang and S. A. Shamma. "Spectral shape analysis in the central auditory system". *IEEE Transactions on Speech and Audio Processing*, 3(5):382–395, September 1995.

[48] X. Yang, K. Wang, and S. A. Shamma. "Auditory representations of acoustic signals". *IEEE Transactions on Information Theory*, 38(2):824–839, March 1992.

[49] Y. Zhuang and J. S. Baras. "Optimal wavelet basis selection for signal representation". Technical Report T.R. 94-7, Center for Sattelite and Hybrid Communication Networks, University of Maryland at College Park, 1994.