Contents

List of Algorithms

List of Figures

List of Tables

Part I

Introduction

Chapter 1

Introduction

1.1 Introduction

1.2 Contributions of this PhD Thesis

1.3 Publications during the development of this PhD Thesis

1. Graña, M.; D'Anjou, A.; González, A.; Albizuri, F. & Cottrell, M (1994) *Local Stochastic Competition for Vector Quantization of Images* in WCNN94 Proceedings, P. Werbos, H. Szu, B. Widrow (eds.), Lawrence Elbaum, 4, pp.205-210, ISBN: 0-8058-1745-X

2. Graña, M.; D'Anjou, A.; González, A. I.; Albizuri, F. & Cottrell, M. (1994) *Local Stochastic Learning Rule for Image Vector Quantization* in Adaptive Systems, Intelligent Approaches, Massively Marallel Computing and Emergent Techniques in Signal Processing and Communications Proceedings, Docampo, D. & Figueiras, A. (eds.), Universidad de Vigo, pp. 219-222, ISBN: 84-605-1547-8

3. González, A. I.; Graña, M. & D'Anjou, A. (1995) *An Analysis of the GLVQ Algorithm*, IEEE Trans. Neural Networks, 6, pp. 1012-1016

4. Graña, M.; D'Anjou, A.; González, A. I.; Albizuri, F. & Cottrell, M. (1995) *Competitive Stochastic Neural Networks For Vector Quantization Of Images* Neurocomputing, 7, pp. 187-195

5. González, A. I.; Graña, M.; D'Anjou, A. & Cottrell, M. (1996) *On the application of competitive neural networks to time-varying clustering problems* in Spatiotemporal models in biological and artificial systems, Silva, F.; Principe, J. & L.B., A.(eds.), IOS Press, pp. 49-55, ISBN: 90-5199-304-8

6. Graña, M.; D'Anjou, A.; Albizuri, F.; Lozano, J.; Larrañaga, P.; Yurramendi, Y.; Hernandez, M.; Jimenez, J.; Torrealdea, F. J.; Poza, M. & González, A. I. (1996) *Experimentos de Aprendizaje con Máquinas de Boltzmann de Alto Orden,* Informatica y Automatica, 29, pp. 42-57

7. Graña, M.; González, A. I.; D'Anjou, A.; Lozano, J.; Larrañaga, P. & Albizuri, F. (1996) *Evolutive strategies for adaptive color quantization* in Intelligent Methods for Signal Processing and Communications, Docampo, D.; Figueiras, A. & Perez, F. (eds.), Universidad de Vigo, pp. 174-178, ISBN: 84-8158-043-0

8. González, A. I. & Graña, M. (1997) *Competitive neural networks as adaptive algorithms for non-stationary clustering: experimental results on the Color Quantization of image sequences* in ICNN97, IEEE press, 3, pp. 1602-1607, ISBN: 0-7803-4122-8

9. González, A. I.; Graña, M.; D'Anjou, A. & Albizuri, F. (1997) *A near real-time evolutive strategy for adaptive Color Quantization of image sequences* in Joint Conf. Inf. Sciences 1997, Wang, P.P. (eds.), Duke University Press., NC, USA, 1, pp. 69-72 ISBN: 0-9643456-4-1

10. González, A. I.; Graña, M.; D'Anjou, A.; Albizuri, F. & Cottrell, M. (1997) *Self Organizing Map for Adaptive Non-stationary Clustering: some experimental results on Color Quantization of image sequences* in ESANN97, Verleysen, M. (eds.), dFacto press, Bruselas, pp. 199-204 ISBN: 2-9600049-7-3

11. González, A. I.; Graña, M.; D'Anjou, A.; Albizuri, F. & Cottrell, M. (1997) *A Sensitivity Analysis of the Self Organizing Map as an Adaptive Onepass Non-stationary Clustering Algorithm: The Case of Color Quantization of Image Sequences* in Neural Processing Letters, 6, pp. 77-89

12. González, A. I.; Graña, M.; D'Anjou, A. & Torrealdea, F. J. (1997) *Anticipation versus adaptation in Evolutionary Algorithms: The case of Non-Stationary Clustering* in Computing Anticipatory Systems: CASYS - First International Conference, Dubois, M.D. (eds.), Springer Verlag, AIP Conference Proceedings., 437, pp. 517-527, ISBN: 1-56396-827-4

13. González, A. I.; Graña, M.; Lozano, J. & Larrañaga, P. (1997) *Experimental results of a Michigan-like Evolutionary Strategy for nonstationary Clustering* in Artificial Neural Nets and Genetic Algorithms, Smith, G. D.; Steele, N. C. & Albrecht, R. F. (eds.), Springer Verlag, pp. 555-559, ISBN: 3-211-83087-1

14. Graña, M.; D'Anjou, A.; Albizuri, F. X.; Hernández, M.; Torrealdea, F. J.; de la Hera, A. & González, A. I. (1997) *Experiments of Fast Learning with High Order Boltzmann Machines* in Applied Intelligence, Kluwer Academic Publishers, 7, pp. 287-303

15. González, A. I. & Graña, M. (1998) *Una estrategia evolutiva para clustering no estacionario* in Inteligencia Artificial, 98, pp. 68-73

16. González, A. I.; Graña, M.; D'Anjou, A.; Albizuri, F. & Torrealdea, F. J. (1998) *A comparison of experimental results with a Evolution Strategy and Competitive Neural Networks for near real-time Color Quantization of image sequences* in Applied Intelligence special issue "Evolutionary Learning", 8, pp. 43-51

17. Graña, M.; González, A. I.; Raducanu, B. & Echave, I. (1998) *Fast face localization for mobile robots: signature analysis and color processing* in Intelligent robots and computer vision XVII: Algorithms, Techniques and Active Vision, Casasent, D.P. (eds.), SPIE Conference Proceedings., pp. 387-398, ISBN: 0-8194-2983-X

18. González, A. I. & Graña, M. (1999) *Diseño mediante una estrategia evolutiva de filtros para imágenes basados en agrupamiento* CAEPIA99, Mira, J. & Sánchez-Andrés, J.V. (eds.), Comité Organizador CAEPIA-TTIA'99, pp. 106-113, ISBN: 931170-2-1

19. González, A. I.; Graña, M. & Cottrell, M. (1999) *Basic competitive neural networks as adaptive mechanisms for nonstationary color quantization* in Neural Computing and Applications, 8, pp. 347-367

20. González, A. I.; Graña, M.; Echave, I. & Ruiz-Cabello, J. (1999) *Bayesian VQ image filtering design with fast adaptation competitive neural networks* in Engineering Applications of Bio-inspired Artificial Neural Networks, Mira, J. & Sánchez-Andrés, J.V. (eds.), Springer-Verlag, LNCS 1607, pp. 341-350, ISBN: 3-540-66068-2

21. Graña, M.; González, A. I.; Echave, I. & Ruiz-Cabello, J. (1999) V*Q based image Filtering* in Pattern Recognition and Image Analysis, Torres, M. & Sanfeliu, A. (ed.), Ediciones GNB. , pp. 471-478, ISBN: 84-95120-80-1

22. Albizuri, F.; González, A. I.; Graña, M. & D'Anjou, A. (2000) *Neural learning for distributions on categorical data* in Neurocomputing, 34, pp. 1-9

23. González, A. I. & Graña, M. (2000) *Colour Quantization of Image Sequences using competitive neural networks* in Image Processing II: Mathematical Methods, Algorithms & Applications, Blackledge, J. & Turner, M. (eds.), Horwood publishing, pp. 313-338, ISBN: 1-898563-61-6

24. González, A. I. & Graña, M. (2000) *Experimental results of an Evolution Strategy for VQ Image Filtering* in Proc 5th JCIS, Wang, P.P. (eds.), Association for Intelligent Machinery, Inc., pp. 1066-1069, ISBN 0-9643456-9-2

25. González, A. I.; Graña, M.; Albizuri, F.; D'Anjou, A. & Torrealdea, F. J. (2000) *A Near Real-Time Evolution-based Adaptation Strategy for Dynamic Color Quantization of Image Sequences* in Information Sciences, 122, pp.161-183

26. González, A. I.; Graña, M.; Cabello, J. R.; D'Anjou, A. & Albizuri, F. X. (2001) *Experimental resuls of an evolution-based adaptation strategy for VQ image filtering* in Inf. Sci. Inf. Comput. Sci., Elsevier Science Inc., 133, pp. 249-266

27. González, A. I. & Graña, M. (2005) *Controversial empirical results on batch versus one pass online algorithms* in WSOM'05

28. González, A. I.; D'Anjou, A.; García-Sebastián, M. & Graña, M. (2006) *SOM and Neural Gas as Graduated Nonconvexity Algorithms* in International Conference on Computational Science and its Applications M. Gavrilova and alt. (eds.), Springer Verlag, Berlin Heidelberg, LNCS 3982, pp. 1143-1152, ISBN-13 978-3-540-34075-1

29. González, A. I.; D'Anjou, A. & Graña, M. (2006) *Convergence of SOM and NG as GNC algorithms* in WCCI – IJCNN 2006, IEEE Press (eds.), Omnipress, pp. 6578-6584, ISBN 0-7803-9490-9

30. García-Sebastián, M.; González, A. I. & Graña, M. (2007) *Derivation of SOM-like rules for intensity inhomogeneity correction in MRI* in Proceedings of the 9th international work conference on Artificial neural networks, Sandoval, F.; Prieto, A.; Cabestany, J.; Graña, M. (eds.) Springer-Verlag, LNCS 4507, pp. 676-683, ISBN: 978-3-540-73006-4

31. González, A. I. (2007) *Algoritmos de graduación de la no-convexidad en el estudio de la convergencia de las redes neuronales artificiales competitivas* in Actas de las I Jornadas de Inteligencia Computacional, JIC'07, Moreno, M.; Veganzones, M. & García, M. (eds.), Basque Country University Editorial, pp.16-26, ISBN: 978-84-9860-019-3

32. García-Sebastián, M.; González, A. I. & Graña, M. (2009) *An adaptive field rule for non-parametric MRI intensity inhomogeneity estimation algorithm* in Neurocomputing, Elsevier Science Publishers B. V., 72, pp. 3556-3569

33. Beristain, A.; Graña, M. & González, A. I. (2011) *A pruning algorithm for stable Voronoi skeletons* in Journal of Mathematical Imaging and Vision, (*-first online-*)

34. Graña, M. & González, A.I. (2011) *Towards Relevant Dendritic Computing* in Proceedings of Third World Congress on Nature and Biologically Inspired Computing (NaBIC'11).

1.4 Structure of the PhD Thesis

Chapter 2

Unsupervised learning: Definitions and notations

Unsupervised learning concerns the problem of finding a function from unlabeled data.

Cluster analysis and the related Vector Quantization design problem are important techniques in many engineering and scientific disciplines. They have applications in signal processing, pattern recognition, machine learning and data analysis [97, 138, 73, 76, 148, 89]. A vast number of approaches have been proposed to solve these problems, among them competitive neural networks have been proposed as a kind of adaptive partitional methods [192, 3, 142, 172]. In the most usual formulations of the Clustering or Vector Quantization problems the assumption is that the underlying stochastic process is stationary and that a given set of sample vectors properly characterizes this process. We will also consider here a time-varying formulation for these problems.

Clustering [76, 89, 148, 268] is the task of grouping a set of data vectors into groups or clusters, so that data vectors in the same cluster are more similar to each other than data vectors belonging to differents clusters. The quality of the solution is measured by a given clustering criterium function, based on a similarity measure. Vector Quantization [97, 130, 131] tries to find a set of representatives or codevectors that minimize the expected quantization error, based on some appropriate distance, when encodes the set of data vectors. This procedure divides the data set into a number of subregions called Voronoi regions, where all data vectors collected in them are represented by the corresponding codevector.

Both problems can be formulated as nonconvex and nonlinear optimization problems. Competitive Neural Networks are a class of procedures for stochastic gradient minimization that have been applied to these problems. In Competitive Neural Networks, neurons compite for activate (win) upon presentation of a subset of the input data and modify their weights in response to this input set. In neural networks, like Perceptrons or Adalines, the neurons adapted their weights

in response to desired outputs (known) performing a supervised learning. In competitive learning, input data with which it works lack the information about the class it belongs, so this learning is referred as unsupervised learning. To avoid falling into local minima during the learning process, a common approach is to modify the basic adaptation rule so that, upon the presentation of an input vector, not only are updated the winning neuron but also other neurons based on their proximity to that input vector.

2.1 Clustering

Clustering methods may be divided into two main groups: Partitional Clustering and Hierarchichal Clustering methods. Partitional Clustering methods generate a single partition of the data in an attempt to recover natural groups present in the data; they are antagonist of Hierarchichal Clustering tecniques which organize the data into a nested sequence of groups [148].

Partitional Clustering problem can be formally described as follows: *Given N data vectors of $d-$dimensional metric space, stablish a partition of the data vectors into M groups, or clusters, such that the data vector in a cluster are more similar to each other than to data vector in different clusters.* The value of M may or may not be specified *a priori*. A local or global clustering criterion must be selected. A global criterion represents each cluster by a representative vector and assigns the data vectors to clusters according to most similar representatives. A local criterion forms clusters by utilizing local structure in the data. The steps to find a solution:

1. Choose a criterion,

2. Compute all possible partitions with M clusters , and,

3. Select the partition that optimizes the criterion.

This is a combinatorial problem. An alternative to optimize the criterion function is using an iterative technique that only is examined a small number of partitions, thus starting with an initial partition, data vectors are assigned to clusters in order to reduce the value of the criterion function. These methods are computationally efficient but often converge to local minima of the criterion function.

The criterion function commonly used is based on the square-error criterion or within-cluster distortion for a fixed number of clusters. Starting from an initial partition of the set of N data vectors into M clusters $\{C_1, C_2, \cdots, C_M\}$ such that cluster C_i has n_i data vectors and each data vector is in exactly one cluster, so that $\sum_{i=1}^{M} n_i = N$. The center of cluster C_i is defined as the centroid of the cluster: $\mathbf{y}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} \mathbf{x}_j^{(i)}$ where $\mathbf{x}_j^{(i)}$ is the jth data vector belonging to cluster C_i. The square-error for cluster C_i is the sum of the squared Euclidean distances between each pattern in C_i and its cluster center $\mathbf{y}_i$ and the total

square-error or within-cluster distortion is computed as:

$$\xi^2 = \sum_{i=1}^{M} \sum_{j=1}^{n_i} \left(\mathbf{x}_j^{(i)} - \mathbf{y}_i \right)^{\mathrm{T}} \left(\mathbf{x}_j^{(i)} - \mathbf{y}_i \right)$$

Partitions are updated by reordering data vectors to clusters in an attempt to reduce this square-error, for example, allocating every data vector to its closest cluster center according to a specific metric, such as Euclidean distance or Mahalanobis distance. The objective is to find a partition containing M clusters that minimizes ξ^2 for fixed M. Different initial partitions can lead to different final clustering solutions because algorithms based on square-error can converge to local minima.

2.2 Vector Quantization

Let us assume a set of sample vectors, $\mathbf{X} = \{\mathbf{x}_1, .., \mathbf{x}_N\}$, each $x_i \in \mathbb{R}^d$. A vector quantizer $\mathcal{Q}$ is a map from each $\mathbf{x}_i$ into a finite set $\mathbf{Y} = \{\mathbf{y}_1, ..., \mathbf{y}_M\}$ of representative vectors or codevectors, usually called the codebook:

$$Q : \mathbb{R}^d \rightarrow \mathbf{Y}$$

where each $\mathbf{y}_i \in \mathbb{R}^d$ with and M is the size of the codebook.

A vector quantizer can be decomposed into two operations: vectorial encoder C and vectorial decoder D:

$$C : \mathbb{R}^d \rightarrow I$$

$$D : I \rightarrow \mathbb{R}^d$$

where $I = \{1, \ldots, M\}$ is the number of different codes. The vector quantization map is usually defined by the nearest neighbor rule as vectorial encoder:

$$C_{NN}(\mathbf{x}, \mathbf{Y}) = i \; s.t. \; \|\mathbf{x} - \mathbf{y}_i\| = \min_{j=1..M} \{\varepsilon(\mathbf{x}, \mathbf{y}_j)\}$$

where $\varepsilon(.)$ is a similarity measure. And vectorial decoder, that allows to recover the codified image, is, thus, defined as:

$$D(i, \mathbf{Y}) = \mathbf{y}_i$$

Associated with each codebook there is a partition of the input space:

$$R_i = \left\{ \mathbf{x} \in \mathbb{R}^d \,|\, C(\mathbf{x}, \mathbf{Y}) = i \right\}$$

$$\bigcup_{i=1}^{M} R_i = \mathbb{R}^d; \; \bigcap_{i=1}^{M} R_i = \emptyset$$

Given a measure of the reproduction error $\varepsilon(\mathbf{x},\widehat{\mathbf{x}}) = \varepsilon(\mathbf{x},\mathcal{Q}(\mathbf{x}))$. The Vector Quantizer performance is measured by the expectation of this reproduction error

$$\xi = E_{\mathbf{x}}[\varepsilon(\mathbf{x},\widehat{\mathbf{x}})] = \int \varepsilon(\mathbf{x},\widehat{\mathbf{x}})\, p(\mathbf{x})\, d\mathbf{x} \tag{2.1}$$

where $p(\mathbf{x})$ is the probability density function (p.d.f.) of the input vectors. When the source that generates the input vectors is stationary this p.d.f. remains unchanged over time. Given a sample $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ the performance (2.1) can be estimated computing the mean error over the sample

$$\widehat{\xi} = \frac{1}{N}\sum_{j=1}^{N} \varepsilon(\mathbf{x}_j,\widehat{\mathbf{x}_j}) \tag{2.2}$$

Based on the similarity measure adopted, this partition of input space is different. The most widely used error measure, and the one that we will effectively use in our experimental work, is the squared Euclidean distance

$$\varepsilon_E^2(\mathbf{x},\mathbf{y}) = \|\mathbf{x}-\mathbf{y}\|^2 = (\mathbf{x}-\mathbf{y})^{\mathrm{T}}(\mathbf{x}-\mathbf{y})$$

which is the base for the computation of the Euclidean distortion

$$\xi_E^2 = \sum_{i=1}^{M} P_i E_{\mathbf{x}}\left[\|\mathbf{x}-\mathbf{y}_i\|^2 \,|\mathbf{x} \in R_i\right]$$

A special class of vector quantizers is the Nearest Neighbor quantizers in which the partition of the input space is defined as follows

$$R_i = \{\mathbf{x}\,|i = \arg\min\{\varepsilon(\mathbf{x},\mathbf{y}_i)\,; i = 1, ..., M\}\}$$

when the distortion measure is the squared Euclidean distance, the Nearest Neighbor quantizer becomes a Voronoi quantizer, based on a Voronoi tesselation of the input space. It must be noted that the partition of the input space induced by the Nearest Neighbor quantizer does strongly depend upon the error measure considered.

The problem of the design of a Vector Quantizer is the search for the codebook that minimizes the distortion introduced by replacing the original $\mathbf{x}$ vectors by their class representative $\mathbf{y}_i$:

$$\min_{\mathbf{Y}} \xi$$

.

When the error measure of the quantization is the squared Euclidean distance and the partition is based on the Nearest Neighbor approach, both Partitional Clustering [76, 89, 148, 268] and VQ can be formally stated as the following optimization problem

$$\min_{\mathbf{Y}} \widehat{\xi_E^2} = \min_{\mathbf{Y}} \frac{1}{N}\sum_{i=1}^{M}\sum_{j=1}^{N} \|\mathbf{x}_j - \mathbf{y}_i\|^2 \delta_i(\mathbf{x}_j, \mathbf{Y}) \tag{2.3}$$

where δ_i is the membership coefficient

$$\delta_i(\mathbf{x}, \mathbf{Y}) = \begin{cases} 1 & i = \arg \min_{k=1,..,M} \left\{ \|\mathbf{x} - \mathbf{y}_k\|^2 \right\} \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

2.3 Competitive Neural Networks

Competitive Neural Network (CNN) algorithms are derived to solve the Clustering and Vector Quantization problems as adaptive algorithms that perform stochastic gradient descent on a distortion like criterium function [268, 140, 142, 168, 172, 157], which varies from one scheme to another scheme network. When the energy function is known, the learning rule associated with it can be formally derived. Thus, the basic competitive learning rule or Simple Competitive Learning (SCL) can be deduced as a minimization algorithm of the within-cluster distortion in terminology of clustering or, in terminology of vector quantization, the quantization distortion by the method of stochastic gradient descent. However, most of the learning rules of CNN's have been proposed on intuitive reasoning and, in some cases, extremely difficult or impossible to derive analytically the formulation of the objective function would be minimized by these rules.

We start recalling the definition of the general competitive learning rule. After that we give the formulations of the SCL, the Self-Organizing Map (SOM), Fuzzy Learning VQ (FLVQ), Neural Gas (NG) and Soft Competition Scheme (SCS) as instances of the general competitive learning rule. We discuss the objective function minimized by each of these learning rules, and their relation to the Euclidean distortion. The argument is that, as discussed in [43], the SOM and other competitive neural networks can be taken as robust initialization procedures for the SCL, when the goal is the minimization of the Euclidean distortion. Similar arguments appear in the literature [30, 236, 270, 288, 292] supporting diverse attempts to define robust Clustering/VQ algorithms.

2.3.1 The general Competitive Learning rule

The general expression of the learning rule for Competitive Neural Networks has the following shape: $\mathbf{x}(t) \in \mathbf{X}$; $1 \leq i \leq M$

$$\mathbf{y}_i(t+1) = \mathbf{y}_i(t) + \alpha_i(t)\, \Phi_i(\mathbf{x}(t), \mathbf{Y}(t), \phi(t))\, (\mathbf{x}(t) - \mathbf{y}_i(t)) \tag{2.5}$$

where M is the number of competitive units, $\mathbf{y}_i(t)$ and $\mathbf{Y}(t)$ are the i-th codevector and the codebook at adaptation time t. The neighboring function $\Phi(.)$ defines the set of units that are adapted upon presentation of the input vector $\mathbf{x}(t)$, by defining how the input vector affects the codevector $\mathbf{y}_i$, being ϕ the neighborhood control parameter for each particular neighboring function. The $\alpha_i(t)$ denotes the (local) learning rate, in order to guarantee theoretical

convergence the learning rate must cope with the Robins-Monro conditions [89, 168, 174, 231, 268]:

- $\lim_{t\to\infty} \alpha(t)=0$,
- $\sum_{t=0}^{\infty} \alpha(t) = \infty$, and
- $\sum_{t=0}^{\infty} \alpha^2(t) < \infty$.

These conditions imply very lengthy adaptation processes, so that in practice they are often overlooked.

Let us consider that this function remains fixed during adaptation

$$\Phi_i(\mathbf{x}(t), \mathbf{Y}(t), \phi(t)) = \Phi_i(\mathbf{x}, \mathbf{Y}, \phi); \forall t \tag{2.6}$$

We can hypothesize that the learning rule is the stochastic gradient minimization of an objective function. That means that we are assuming that the neighboring function is the opposite of the instantaneous gradient of the hypothetical objective function:

$$\Phi_i(\mathbf{x}, \mathbf{Y}, \phi) = -\left.\frac{\partial}{\partial \mathbf{y}_i}\xi_\Phi\right|_{\mathbf{x}}$$

and that, under the appropriate conditions, this function can be deduced from the neighboring function as follows:

$$\xi_\Phi = E_{\mathbf{x}}\left[\sum_{i=1}^{M}\int -\Phi_i(\mathbf{x}, \mathbf{Y}, \phi)(\mathbf{x}-\mathbf{y}_i)\, d\mathbf{y}_i\right] \tag{2.7}$$

Usually, the neighboring functions vary during the adaptation or learning process. The analytical form of the minimized function then becomes much more complex, most of the times it remains unknown. A convenient approach is to view the learning rule (2.5) as performing a cascade of minimizations of a sequence of objective functions

$$\xi_\Phi(t) = \sum_{i=1}^{M}\int\int -\Phi_i(\mathbf{x}, \mathbf{Y}, \phi(t))(\mathbf{x}-\mathbf{y}_i)\, p(\mathbf{x})\, d\mathbf{y}_i d\mathbf{x}$$

The limit of this sequence of objective functions will be determined by the limit of their respective neighboring functions:

$$\lim_{t\to\infty} \Phi_i(\mathbf{x}, \mathbf{Y}, \phi(t)) = \Phi_i^*(\mathbf{x}, \mathbf{Y}, \phi) \Longrightarrow \lim_{t\to\infty} \xi_\Phi(t) = \xi_{\Phi^*}$$

The application of (2.5) is, then, a minimization procedure of the limit objective function. It is expected that the application of the general competitive rule (2.5) will produce a global minimization of the limit objective function ξ_{Φ^*}. The process is similar to an annealing [1] defined over the width and shape of the neighborhood function. In the most common application of (2.5) the

neighboring function will shrink to approach the Nearest Neighbor membership function:

$$\lim_{t\to\infty} \Phi_i\left(\mathbf{x}, \mathbf{Y}, \phi\left(t\right)\right) = \delta_i\left(\mathbf{x}, \mathbf{Y}\right)$$

Where the Nearest Neighbor membership function is (2.4) Therefore, the sequence of objective functions usually converges to the Euclidean distortion

$$\lim_{t\to\infty} \xi_\Phi\left(t\right) \approx \xi_E^2 \tag{2.8}$$

or to some equivalent function that has its minima in the same places. This convergence is controlled by the specific parameter ϕ of the neighboring function. Under this view, SOM, NG, FLVQ and SCS are taken as robust initialization procedures for the SCL. Therefore, they are expected to improve the performance of SCL in the global minimization of the Euclidean distortion. This interpretation is in contrast with the interpretation of the neighboring functions as modifications of the learning rate $\alpha_i\left(t\right)$, such as discussed in [30, 288]. We consider that the learning rate must have the same scheduling in all cases, and that the remaining coefficients characterize the sequence of objective functions being minimized.

2.3.2 The Simple Competitive Learning

The Simple Competitive Learning (SCL) is the straighforward minimization of the Euclidean distortion 2.3 [89, 140, 142, 168, 172, 268]. SCL algorithm is the simplest adaptive algorithm for Clustering or VQ. It corresponds to the (2.5) when the neighboring function is defined as the Nearest Neighbor function:

$$\Phi_i\left(\mathbf{x}, \mathbf{Y}, \phi\right) = \delta_i\left(\mathbf{x}, \mathbf{Y}\right) = \begin{cases} 1 & i = \arg\min\limits_{k=1,\ldots,M} \left\{\left\|\mathbf{x} - \mathbf{y}_k\right\|^2\right\} \\ 0 & \text{otherwise} \end{cases} \tag{2.9}$$

There is not any neighborhood control parameter, thus $\phi = \emptyset$. No neighborhood is considered only the winning codevector is adjusted. The convergence of the SCL to the optimal codebook will depend upon the initial conditions, because of the local nature of gradient descent. The accuracy of the results will also depend upon the learning rate parameter $\alpha_i\left(t\right)$, whose scheduling will be discussed at the end of this section. The SCL is the stochastic equivalent to the Isodata algorithm or K-means.

2.3.3 The Self Organizing Map

The fundamental idea of self-organizing feature maps was originally introduced by Malsburg [191] and Grossberg [133] to explain the formation of neural topological maps. Based on this work Kohonen [167, 168] proposed the model known as Self-Organizing Maps (SOM) which has been successfully applied to a large number of engineering applications up to date. A number of algorithms have

been proposed in the literature based on the original idea, among them growing structures like [87].

The SOM consists of the updating of the winner (in the Euclidean distance) and his neighbors under a topological structure of the codevector indices. This topological structure is specified by a distance defined over the set of indices. Let us denote the winning unit as

$$w(\mathbf{x}, \mathbf{Y}) = \arg \min_{k=1,..,M} \left\{ \|\mathbf{x} - \mathbf{y}_k\|^2 \right\} \tag{2.10}$$

The most general formulations of the SOM are given by neighboring functions defined over the distance from the unit to the winning unit:

$$\Phi_i^{\psi}(\mathbf{x}, \mathbf{Y}) = \psi(|w(\mathbf{x}, \mathbf{Y}) - i|); 1 \leq i \leq c$$

where $\psi(.)$ is a decreasing function and $|.|$ is the distance defined over the index space. We have chosen for our works the simplest definition of the SOM, in which the index distance is the absolute distance and all the codevectors with indices inside a neighborhood radius v are equally allowed to adapt. The time invariant neighborhood is, thus, defined as:

$$\Phi_i^{(SOM)}(\mathbf{x}, \mathbf{Y}, v) = \begin{cases} 1 & |w(\mathbf{x}, \mathbf{Y}) - i| \leq v \\ 0 & otherwise \end{cases}; 1 \leq i \leq M \tag{2.11}$$

It is trivial to verify

$$\lim_{v \to \infty} \Phi_i^{(SOM)}(\mathbf{x}, \mathbf{Y}, v) = 1; \quad \lim_{v \to 0} \Phi_i^{(SOM)}(\mathbf{x}, \mathbf{Y}, v) = \delta_i(\mathbf{x}, \mathbf{Y})$$

so that (2.11) becomes the crisp membership for null v, and it is perfectly fuzzy for large v.

The function minimized by (2.5) and (2.11) is an extension of the Euclidean distortion [43, 142, 161, 168] in which each codevector contributes the distortion due to the quantization of its input Voronoi tesselation and those of its neighboring codevectors relative to itself. The time dependent neighboring function is based on a time dependent neighboring radius $v(t)$:

$$\Phi_i^{(SOM)}(\mathbf{x}, \mathbf{Y}, v(t)) = \begin{cases} 1 & |w(\mathbf{x}, \mathbf{Y}) - i| \leq v(t) \\ 0 & otherwise \end{cases}; 1 \leq i \leq M$$

The neighboring radius is decreased gradually so that

$$\lim_{t \to \infty} v(t) = 0 \Longrightarrow \lim_{t \to \infty} \Phi_i^{(SOM)}(\mathbf{x}, \mathbf{Y}, v(t)) = \delta_i(\mathbf{x}, \mathbf{Y})$$

and therefore

$$\lim_{t \to \infty} \xi_{\Phi^{(SOM)}}(t) \approx \xi_E^2$$

The functional convergence will depend on the parameters controlling the neighborhood radius. The formal analysis of the convergence of the SOM is a research topic by itself [66, 83, 84, 85]. Much of the study is devoted to its convergence to organized states. From our perspective, the convergence to organized states is only significative if the organized states ensure better initial conditions for the minimization Euclidean distortion performed by the SCL rule.

2.3.4 The Fuzzy Learning Vector Quantization

The Fuzzy Learning Vector Quantization (FLVQ) [30, 61, 158, 159, 211, 270] has its roots are in the minimization of the Fuzzy Clustering criterium [26]. In its definition FLVQ was proposed as a batch algorithm, howewer, we consider it as an on-line algorithm in the class characterized by (2.5). The neighborhood control parameter ϕ is the exponent m and the neighboring function has the form $(1 \leq i \leq M)$:

$$\Phi_i^{(FLVQ)}(\mathbf{x}, \mathbf{Y}, m) = (u_i(\mathbf{x}, \mathbf{Y}))^m = \left(\sum_{k=1}^{M} \left(\frac{\|\mathbf{x} - \mathbf{y}_i\|^2}{\|\mathbf{x} - \mathbf{y}_k\|^2} \right)^{\frac{1}{m-1}} \right)^{-m} \tag{2.12}$$

where $u_i(\mathbf{x}, \mathbf{Y})$ is the fuzzy membership function. This neighboring function is well defined when $\mathbf{x} \neq \mathbf{y}_i$. It can be shown [30] that this neighboring function becomes the Nearest Neighbor function when m approaches 1 from above.

$$\lim_{m \to 1^+} u_i(\mathbf{x}, \mathbf{Y}) = \delta_i(\mathbf{x}, \mathbf{Y}) \Longrightarrow \lim_{m \to 1^+} \Phi_i^{(FLVQ)}(\mathbf{x}, \mathbf{Y}, m) = \delta_i(\mathbf{x}, \mathbf{Y})$$

When m grows, the neighboring function becomes more fuzzy, but it is also exponentially damped

$$\lim_{m \to \infty} u_i(\mathbf{x}, \mathbf{Y}) = \frac{1}{M} \Longrightarrow \lim_{m \to \infty} \Phi_i^{(FLVQ)}(\mathbf{x}, \mathbf{Y}, m) = 0$$

so that, the learning rule (2.5) with the neighboring function given by (2.12) becomes the SCL when m approaches 1 from above. When m grows to infinity, all the codevectors are equally affected by the input, but this influence is exponentially damped. Very big values of m give no adaptation at all. The minimized function at constant m is related to the fuzzy clustering criterium, although FLVQ can not be derived as an stochastic gradient descent of it. The time dependent neighboring function is defined by changing the neighborhood control parameter in time:

$$\Phi_i^{(FLVQ)}(\mathbf{x}, \mathbf{Y}, m(t)) = \left(\sum_{k=1}^{M} \left(\frac{\|\mathbf{x} - \mathbf{y}_i\|^2}{\|\mathbf{x} - \mathbf{y}_k\|^2} \right)^{\frac{1}{m(t)-1}} \right)^{-m(t)} ; 1 \leq i \leq M \tag{2.13}$$

The schedule starts from large values of $m(t)$ and decrease to 1.

$$\lim_{t \to \infty} m(t) = 1^+ \Longrightarrow \lim_{t \to \infty} \Phi_i^{(FLVQ)}(\mathbf{x}, \mathbf{Y}, m(t)) = \delta_i(\mathbf{x}, \mathbf{Y}) \tag{2.14}$$

$$\Longrightarrow \lim_{t \to \infty} \xi_{\Phi^{(FLVQ)}}(t) \approx \xi_E^2 \tag{2.15}$$

The initial fuzziness due to the large values of $m(\tau)$ would move all the codevectors to the region of the space occupied by the sample data. The proper scheduling of $m(\tau)$ is critical. Too long fuzzy periods will produce a collapse of the codevectors to the sample mean. Too fast convergence to SCL will not give any improvement upon it.

2.3.5 The Soft Competition Scheme

The first reference found in the literature to the Soft Competition Scheme is [288], where the SCS is proposed as a combination of simulated annealing and SCL. In [30] the algorithm is revisited giving an statistical interpretation of the neighboring coefficients. They are shown to be the a posteriori probabilities of the input assuming a mixture of Gaussians as the model of the data distribution. Let us consider $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ as a sample of a random vector $\mathcal{X}$, and let us consider the hypothesis of its probability density function being a mixture of Gaussian densities: $P_{\mathcal{X}}(\mathbf{x}) = \sum_{i=1}^{M} p(\omega_i)\, \Psi(\mathbf{x}\,|\mu_i, \Sigma_i)$ where $p(\omega_i)$ are the *a priori* probabilities of the classes (clusters) and $\Psi(\mathbf{x}\,|\mu_i, \Sigma_i)$ denotes a Gaussian density with mean μ_i and variance-covariance matrix Σ_i that models the conditional density $p(\mathbf{x}\,|\omega_i)$. Then, the clustering problem becomes a search for the parameters (μ_i, Σ_i) that provide the best fit to the empirical distribution $\{P^*_{\mathcal{X}}(\mathbf{x}_i)\}$ computed from the sample. This search can be performed as an stochastic gradient minimization of the Kullback-Leiber cross-entropy: $C_{KL} = \sum_{i=1}^{n} P^*_{\mathcal{X}}(\mathbf{x}_i) \log \frac{P^*_{\mathcal{X}}(\mathbf{x}_i)}{P_{\mathcal{X}}(\mathbf{x})}$. In the case of diagonal convariance matrices, $\sum_i = \sigma_i^2 \mathbf{I}$, the adaptation rule derived as the stochastic gradient descent of this measure has the shape of a Competitive Neural Network (2.5) with a normalized Gaussian as the neighboring function. Thus, for the SCS algorithm, the neighborhood control parameter is the variance σ^2 assumed around the cluster representatives, and the neighboring function has the form $1 \leq i \leq M$:

$$\begin{aligned} \Phi_i^{(SCS)}(\mathbf{x}, \mathbf{Y}, \sigma) &= \Psi\left(\mathbf{x}\,\middle|\mathbf{y}_i, \sigma_i^2\mathbf{I}\right)\left(\sum_{k=1}^{M} \Psi\left(\mathbf{x}\,\middle|\mathbf{y}_k, \sigma_k^2\mathbf{I}\right)\right)^{-1} \\ &= e^{-\frac{1}{2}\|\mathbf{x}-\mathbf{y}_i\|^2\sigma^{-2}}\left(\sum\nolimits_{k=1}^{M} e^{-\frac{1}{2}\|\mathbf{x}-\mathbf{y}_k\|^2\sigma^{-2}}\right)^{-1} \end{aligned} \tag{2.16}$$

As with FLVQ, it is possible to show that:

$$\lim_{\sigma\to\infty} \Phi_i^{(SCS)}(\mathbf{x}, \mathbf{Y}, \sigma) = \tfrac{1}{M}; \quad \lim_{\sigma\to 0} \Phi_i^{(SCS)}(\mathbf{x}, \mathbf{Y}, \sigma) = \delta_i(\mathbf{x}, \mathbf{Y})$$

and, therefore, that SCS converges to SCL for specific values of the neighboring parameter. We have assumed the simplest model. This model can be made more complex by allowing for each component of the mixture different standard deviations σ_i , different standard deviations σ_{ij} for each axis or even non-diagonal covariance matrices Σ_i. The standard deviation is decreased to perform the desired functional convergence. Therefore, we define the time dependent neighborhood as $1 \leq i \leq M$:

$$\Phi_i^{(SCS)}(\mathbf{x}, \mathbf{Y}, \sigma(t)) = e^{-\frac{1}{2}\|\mathbf{x}-\mathbf{y}_i\|^2\sigma(t)^{-2}}\left(\sum\nolimits_{k=1}^{M} e^{-\frac{1}{2}\|\mathbf{x}-\mathbf{y}_k\|^2\sigma(t)^{-2}}\right)^{-1} \tag{2.17}$$

and the process starts from large values of the assumed standard deviation and

decrease it to zero, so that

$$\begin{aligned}\lim_{t\to\infty}\sigma(t) &= 0 \Longrightarrow \lim_{t\to\infty}\Phi_i^{(SCS)}(\mathbf{x},\mathbf{Y},\sigma(t)) = \delta_i(\mathbf{x},\mathbf{Y}) \\ &\Longrightarrow \lim_{t\to\infty}\xi_{\Phi^{(SCS)}}(t) \approx \xi_E^2\end{aligned}$$

This process is very sensitive to the scheduling of the variances. Too long adaptations at big values of the standard deviation will collapse the codevectors to the sample mean. Too fast convergence to SCL will not improve over SCL.

2.3.6 The Neural Gas

This Competitive Neural Network introduced in [193] shares the structure shown in equation (2.5) characterized by the following time invariant neighboring function:

$$\Phi_i^{NG}(\mathbf{x},Y,\lambda) = e^{(-k_i(\mathbf{x},\mathbf{Y})/\lambda)} \tag{2.18}$$

where $k_i(\mathbf{x},\mathbf{Y})$ is the rank function that returns the position $\{0,\ldots,M-1\}$ of the codevector $\mathbf{y}_i$ in the set of codevectors ordered by their distances to the input $\mathbf{x}$: $k_i(\mathbf{x},\mathbf{Y}) = \{\mathbf{y}_j \,|\|\mathbf{y}_j-\mathbf{x}\| \le \|\mathbf{y}_i-\mathbf{x}\|\}$. The neighborhood parameter control ϕ is the temperature λ. All codevectors are updated, there are not properly defined neighbours, but the temperature paremeter λ controls the span of the effect that the presentation of an input vector has on the unit weights. It must be noted that the neighborhood function in equation (2.18) is equal to 1 for the winning unit w regardless of the temperature parameter. As the temperature goes to zero, the neighboring function goes also to zero for the non-winning units, converging to SCL:

$$\lim_{\lambda\to\infty}\Phi_i^{(NG)}(\mathbf{x},\mathbf{Y},\lambda) = 1; \quad \lim_{\lambda\to 0}\Phi_i^{(NG)}(\mathbf{x},\mathbf{Y},\lambda) = \delta_i(\mathbf{x},\mathbf{Y})$$

The time dependent neighboring function is based on a time dependent temperature $\lambda(t)$:

$$\Phi_i^{NG}(\mathbf{x},Y,\lambda(t)) = e^{(-k_i(\mathbf{x},\mathbf{Y})/\lambda(t))} \tag{2.19}$$

The temperature is decreased gradually so that

$$\lim_{t\to\infty}\lambda(t) = 0 \Longrightarrow \lim_{t\to\infty}\Phi_i^{(NG)}(\mathbf{x},\mathbf{Y},\lambda(t)) = \delta_i(\mathbf{x},\mathbf{Y})$$

and therefore

$$\lim_{t\to\infty}\xi_{\Phi^{(NG)}}(t) \approx \xi_E^2$$

2.3.7 The setting of the learning control parameters

In this subsection we will discuss the setting of the critical parameters of the competitive neural networks discussed above. We first discuss the learning rate scheduling, because it is common to all of them. Later we discuss the scheduling of each neighborhood parameter that controls the desired functional convergence.

Here we assume the competitive neural networks as stochastic gradient descent of a given objective function. In order to guarantee theoretical convergence to a (local) minimum of the objective function, the learning rate varies as the adaptation proceeds and it must comply with the Robins-Monro conditions([89, 168, 174, 231, 268]): $\lim_{\tau\to\infty}\alpha(\tau)=0$, $\sum_{\tau=0}^{\infty}\alpha(\tau)=\infty$ and $\sum_{\tau=0}^{\infty}\alpha^2(\tau)<\infty$.

Contrary to [30, 288] we consider the learning rate as a control parameter of the stochastic gradient approach and the neighboring function as a characteristic of the objective function being minimized, therefore we do not mix them in a single parameter. The learning rate schedule must be equally applicable to all the algorithms. The one used in our work has the following features:

1. We assume a "one-pass" adaptation process. The sample is presented only once to perform the adaptation.

2. Local learning rate for each unit $\alpha_i(t)$ $i=1,..,M$.

3. Linear scheduling of the learning rate

$$\alpha_i(t)=0.1\,(1-t_i/N) \tag{2.20}$$

   where N is the sample size, and t_i is the local adaptation time whose exact computation depends on the learning rule.

Regarding the neighboring functions $\Phi_i(\mathbf{x},\mathbf{Y},\phi(t))$, we have chosen an exponential decreasing of the neighborhoods. In general, the rate of convergence to the null neighborhood is denoted r. Therefore we have assumed for the SOM, NG, FLVQ and SCS

$$\Phi_i^{(\cdot)}(\mathbf{x},\mathbf{Y},\phi(t))=\delta_i(\mathbf{x},\mathbf{Y}) \quad t\geq\frac{N}{r}$$

where N is the size of the sample, and t is the adaptation time. When $r=1$ the neighborhood does not become the Nearest Neighbor in all the training. There is not an SCL phase, and the convergence to SCL has no effect. As r grows the convergence to SCL is faster and the SCL phase longer. We do not consider $r<1$ because it does not have any meaning in a one-pass adaptation framework. The scheduling of the neighborhood sizes for the SOM, FLVQ and SCS are given by the following expressions:

$$v(t)=\left\lceil(v_0+1)^{\left(1-\frac{r}{N}t\right)}\right\rceil-1 \quad t\geq\frac{N}{r} \tag{2.21}$$

$$m(t) = m_0 \left(\frac{1.1}{m_0}\right)^{\frac{r}{N}t} \quad t \geq \frac{N}{r} \tag{2.22}$$

$$\sigma(t) = (\sigma_0 + 1)^{\left(1-\frac{r}{N}t\right)} - 1 \quad t \geq \frac{N}{r} \tag{2.23}$$

2.4 Frame Based Adaptive Vector Quantization

In this section we will review the definition of Adaptive Vector Quantization and we will introduce the notion of Frame-Based Adaptive Vector Quantization. Finally we will discuss the proper application of competitive neural networks FBAVQ problems.

2.4.1 Adaptive Vector Quantization

In section 2 we have reviewed the stationary definitions for VQ and Clustering. The stochastic process $\{X(t_j)\, j = 0, 1, 2, ...\}$ modelling the data source was a sequence of i.i.d. random vectors whose joint probability density function (j.p.d.f.) remains time invariant

$$p(\mathbf{x}(t_j), ..., \mathbf{x}(t_{j-m})) = p(\mathbf{x}(t_i), ..., \mathbf{x}(t_{i-m}))\,; t_i \neq t_j \tag{2.24}$$

where m is the memory span of the process. If the stationary process is Gaussian, the mean and the second order moments are enough to characterize it.

In the non-stationary case, the j.p.d.f. is no longer time invariant. That is, (2.24) does not hold for all time. The simplest non-stationary process is the random walk or Wiener process, an unstable autorregresive process of unbounded variance, that is sometimes used as a benchmark [86]. We can characterize an stochastic process as locally stationary when it satisfies

$$p(\mathbf{x}(t_j), ..., \mathbf{x}(t_{j-m})) = p(\mathbf{x}(t_i), ..., \mathbf{x}(t_{i-m}))\,; t_i < t_{j+Nm}; N >> 0 \tag{2.25}$$

that means that (2.24) holds for enough long finite periods of time. The adaptive approach computes on-line the modification of the Vector Quantizer. The assumption of local stationarity is important for the adaptive approach, because it implies that enough data is available for adaptation. Adaptive VQ is discussed in [97]. Basic approaches are the mean reduction, gain adaptation and several strategies for codebook replenishment [58, 86, 295]. Competitive neural networks have been proposed for Adaptive VQ. In [97] they were declared to be of little use because of their long convergence times and lack of robustness. In [176] they were retried for image sequences with some success due to the care in the choosing of the initial conditions. In fact, most of the applications of competitive neural networks to VQ have been reported on still images and image sequences of small variability (i.e.: talking faces).The adaptive approach produces a time varying VQ

$$\widehat{\mathbf{x}}(t) = \mathcal{Q}_t(\mathbf{x}) \tag{2.26}$$

whose quality measure is a time varying error expectation

$$\xi(t) = E_{\mathbf{x},t}\left[\varepsilon\left(\mathbf{x},\widehat{\mathbf{x}}(t)\right)\right] = \int \varepsilon\left(\mathbf{x},\widehat{\mathbf{x}}(t)\right) p(\mathbf{x},t)\, d\mathbf{x} \tag{2.27}$$

2.4.2 Sampling and local stationarity

The vector stochastic process $\{\mathbf{x}(t)\}$ being quantized comes from a sampling procedure that may influence its probabilistic nature. When treating with scalar processes, such as acoustic signals or financial time series, vector stochastic processes are built up by aggregation of scalar samples [97]. In the general non stationary case, the evolution of the underlying physical process is arbitrary both in the functional form of its j.p.d.f. and in the speed of the changes. If the sampling procedure is slow relative to the changes in the underlying physical process, the sample will be composed of a sequence of (independent) vectors obeying different p.d.f.'s Then, a sample obtained in a time window (t_i, t_f) must be denoted as

$$\aleph(t_i, t_f) = \{\mathbf{x}(t_1), ..., \mathbf{x}(t_n)\} \quad with \quad t_1 = t_i; t_f = t_n \tag{2.28}$$

where $\mathbf{x}(t_j)$ is a sample of the signal random vector at time t_j whose probability density is denoted $p(\mathbf{x},t_j)$. Most instances of Adaptive VQ applied to image processing assume that an still image obeys this characterization, and try to perform *intraframe* adaptation. From our point of view, it is difficult in (2.28) to asses some kind of local stationarity. Thus, the tuning of the algorithms is a delicate task, and the results reported must be taken with caution. We assume that the sampling process is fast relative to the underlying physical changes. Therefore, the samples obtained in a time window can be considered a set of i.i.d. random vectors. The sampling procedure produces a sequence of such sets, which we denote

$$\aleph(t) = \{\mathbf{x}_1(t), ..., \mathbf{x}_n(t)\} \quad t = 0, 1, 2, ... \tag{2.29}$$

where the p.d.f. $p(\mathbf{x},t)$ remains invariant for the time window in which the sample has been computed. We will assume t to be the image or frame number, and the same sample size n for all t. This approach can be called Frame-Based Adaptive VQ.

2.4.3 Frame-Based Adaptive VQ

Adaptive VQ for lossy signal compression is analyzed in the framework of rate-distortion theory [25]. However, AVQ can be of use for other tasks, such as image segmentation [156, 158, 192, 210, 272, 276] or non linear projection methods[161]. We define the Frame- Based Adaptive VQ (FBAVQ) as a minimization of the accumulated quality measure

$$\min_{\mathcal{Q}_t} \sum_{t \geq 0} \xi(t)$$

Where $\{\mathcal{Q}_t\}$ is the sequence of coding functions produced by adaptation. For Nearest Neighbor quantizers, the FBAVQ is defined as the search for the sequence of codebooks

$$\mathbf{Y}(t) = \{\mathbf{y}_1(t) \ , .., \mathbf{y}_c(t)\}; t = 0, 1, 2, ... \tag{2.30}$$

that minimize the distortion of the sequence of quantizations performed using them at each time instant. When the sampling procedure produces a sequence of i.i.d. samples as in (2.29), and the distortion measure is the squared Euclidean distance, the FBAVQ can be stated as the following minimization problem:

$$\min_{\mathcal{Q}_t} \sum_{t\geq 0} \widehat{\xi_E^2}(t) = \min_{\mathcal{Q}_t} \sum_{t\geq 0} \sum_{j=1}^{n} \sum_{i=1}^{c} \|-\mathbf{y}_i(t)\|^2 \delta_i(\mathbf{x}_j(t), \mathbf{Y}(t)) \tag{2.31}$$

2.4.4 Application of Competitive Neural Networks to Frame-Based Adaptive VQ

Frame-Based Adaptive VQ as stated in (2.31) is a rather complex infinite time dynamic programming problem. This problem can be made more tractable assuming

$$\min_{\mathcal{Q}_t} \sum_{t\geq 0} \xi(t) = \sum_{t\geq 0} \min_{\mathcal{Q}_t} \xi(t)$$

which becomes for a given sample, and the Nearest Neighbor Euclidean quantization

$$\min_{\{\mathbf{Y}(t)\}} \sum_{t\geq 0} \widehat{\xi_E^2}(t) = \sum_{t\geq 0} \min_{\{\mathbf{Y}(t)\}} \sum_{j=1}^{n} \sum_{i=1}^{c} \|\mathbf{x}_j(t) - \mathbf{y}_i(t)\|^2 \delta_i(\mathbf{x}_j(t), \mathbf{Y}(t))$$

The minimization of the sequence of time dependent error function can be done independently at each time step. The adaptive application of the neural network algorithms is done as follows: At time t the initial cluster representatives are the ones computed from the sample of the process at time $t-1$. The vectors $\aleph(t) = \{\mathbf{x}_1(t), .., \mathbf{x}_n(t)\}$ in the sample at time t are presented sequentially and randomly as inputs to compute the adaptation equations and to obtain a new set of cluster representatives. A distinctive feature of the experimental work reported below is that we impose a one-pass adaptation at each time step. That means that the sample vectors will be presented only once and that the scheduling of the learning rate and other learning control parameters are adjusted to that time constraint.

2.5 Stationary clustering

Stationary cluster analysis assume that the data is a sample $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ of an stationary stochastic process, whose statistical characteristics will not change

in time. Non-stationary Clustering assume that the data come from a non-stationary stochastic process sampled at diverse time instants. That is, the population can be modelled by a discrete time stochastic process $\{X_t; t = 1, 2, \ldots\}$ of unknown joint probability distribution. We do not assume any knowledge of the time dependencies that could allow a predictive approach [3].

2.6 Non-stationary clustering

A working definition of the Non-stationary Clustering problem could read as follows: Given a sequence of samples $\mathbf{X}(t) = \{\mathbf{x}_1(t), \ldots, \mathbf{x}_N(t)\}; t = 0, 1, \ldots$ of the population obtain a corresponding sequence of partitions of each sample that consists of a sequence of sets of disjoint clusters $P(\mathbf{X}(t)) = \{\mathcal{X}_1(t), \ldots, \mathcal{X}_c(t)\}$. This sequence of partitions minimizes a criterium function along time $C = \sum_{t\geq 0} C(t)$. In the general statement of the problem the difinition of the criterium function is based on the definition of an appropriate distance. We follow the conventional approach of using the Euclidean distance. We consider a sequence of codevectors or cluster representatives $\mathbf{Y}(t) = \{\mathbf{y}_1(t), \ldots, \mathbf{y}_M(t)\}$ such that the desired partitions are defined by the nearest (Euclidean) representative.

$$\mathbf{x}_j(t) \in \mathcal{X}_i(t) \Leftrightarrow i = \arg \min_{k=1,\ldots,M} \left\{ \|\mathbf{x}_j(t) - \mathbf{y}_k(t)\|^2 \right\}$$

The criterium function that we will consider at each time step is, therefore, the distortion (or whithin cluster variance)

$$C(t) = \sum_{j=1}^{N} \sum_{i=1}^{M} \|\mathbf{x}_j(t) - \mathbf{y}_i(t)\|^2 \delta_i(\mathbf{x}_j(t), \mathbf{Y}(t)),$$

$$\delta_i(\mathbf{x}_j(t), \mathbf{Y}(t)) = \begin{cases} 1 & i = \arg \min_{k=1,\ldots,M} \left\{ \|\mathbf{x}_j(t) - \mathbf{y}_k(t)\|^2 \right\} \\ 0 & \text{otherwise} \end{cases}.$$

The Non-stationary Vector Quantization design problem can be stated as the search for a sequence of representatives $\mathbf{Y}(t) = \{\mathbf{y}_1(t), \ldots, \mathbf{y}_c(t)\}$ that minimizes the error function, aka reconstruction distortion, $E = \sum_{t\geq 0} E(t)$. The squared Euclidean distance is the dissimilarity measure most widely used to formulate the criterium/error functions. The Non-stationary Clustering/VQ problem can be stated as an stochastic minimization problem:

$$\min_{\{\mathbf{Y}(t)\}} \sum_{t\geq 0} \sum_{j=1}^{N} \sum_{i=1}^{M} \|\mathbf{x}_j(t) - \mathbf{y}_i(t)\|^2 \delta_{ij}(t), \tag{2.32}$$

$$\delta_{ij}(t) = \begin{cases} 1 & i = \arg \min_{k=1,\ldots,M} \left\{ \|\mathbf{x}_j(t) - \mathbf{y}_k(t)\|^2 \right\} \\ 0 & \text{otherwise} \end{cases}.$$

The proposition of adaptive algorithms to solve this stochastic minimization problem is based in two simplifying assumptions:

1. The minimization of the sequence of time dependent error function can be done independently at each time step.

2. Smooth (bounded) variation of optimal set of representatives at successive time steps. Then the set of representatives obtained after adaptation in a time step can be used as the initial conditions for the next time step.

The first condition implies that the problem can be decomposed into a sequence of isolated problems. Therefore, bad solutions in a given time instant do not degrade the overall response of the adaptive algorithm along time. Besides, if we can compute an optimal solution for the stationary case, we can obtain an optimal solution of the non-stationary case, through the computation of the optimal solutions at each time instant.

The second condition implies that adaptive algorithms can be formulated as independent local minimization procedures. The solution computed for the previous time step can be assumed as a good initial condition for the next time step. Therefore, the local minimization performed by the adaptive algorithm can lead to (near) optimal results.

The adaptive computation of the cluster representatives along time can be stated as follows:

- At time t take as initial cluster representatives the ones already computed from the sample of the process at time $t-1$.

- Use the sample vectors $\mathbf{X}(t) = \{\mathbf{x}_1(t), \ldots, \mathbf{x}_N(t)\}$ to perform an adaptive computation leading to the new estimates of the cluster representatives.

2.7 One-pass versus online algorithm versions.

Each algorithm described before has some control parameters, like the learning ratio, the neighbourhood size and shape, or the temperature. The online realizations usually modify their values following each input data presentation and adaptation of the codebook. The batch realizations modify their values after each presentation of the whole input data sample. Both online and batch realizations imply that the input data set is presented several times. On the contrary, the One Pass realizations imply that each input data is presented at most once for adaptation, and that the control parameters are modified after each presentation.

In the case of conventional online realizations of the training process, t either corresponds to the number of iterations performed over the whole sample, and the learning rate and neighbor size is fixed during iteration, or corresponds to the input vector presentation number, and the learning rate and neighbor size is updated at each presentation. The first scheduling is compatible with the batch realization, where the whole sample is used to estimate the network parameters and the computational time is naturally the number of sample iterations.

In the special case of one-pass realization of the training process, t is the input presentation number and its maximum value is the sample size. These

imposes strong schedules of the algorithm control parameters. In the experiments, we use the following expression for the learning rate updating [60]:

$\alpha(t) = \alpha_0 \left(\frac{\alpha_N}{\alpha_0}\right)^{\frac{t}{N}}$

where α_0 and α_n are the initial and final value of the learning rate, respectively. Therefore the learning rate reaches its final value after N input vector presentations.

In the neighboring function SOM (**??**), the neighboring radius $h(t)$ is defined as $h(t) = \left\lceil h_0 \left(\frac{h_N}{h_0}\right)^{t/\tau N} \right\rceil - 1$ where h_0 y h_N are the initial and final neighbourhood radius, respectively. The parameter τ determines that the neighbouring function reduces to the SCL case (null neighbourhood) after the presentation of the first $1/\tau$ sample vectors. We have used $\tau = 8$ in most of our computational experiments, and 1D unit index topologies. With this neighborhood reduction rate, we can get after a quick initial unit ordering, a slow local fine-tuning minimizing the distortion. We proposed this scheduling in [110] to approach real-time constraints and other authors have worked with this idea [42] in the context of conventional online realizations.

2.8 Initial codebooks

We have worked with three possible options to initialize codebooks:

- in Sample: random selection of samples of the (first) image.
- in RGB box: is an arbitrary codebook randomly generated in the input space.
- Threshold algorithm: corresponds to a threshold guided selection of elements in the sample of the (first) image.

2.8.1 A simple threshold algorithm

In order to obtain initial estimates of the codevectors, a simple algorithm based on a threshold applied on the Euclidean distance can provide good results.

The threshold algorithm starts by assigning the first sample vector as the first codevector: $\mathbf{y}_1 = \mathbf{x}_1$. It then goes over the sample until it finds a vector $\mathbf{x}_k$ such that its Euclidean distance to each of the already found codevectors $\{\mathbf{y}_1, .., \mathbf{y}_c; c < M\}$ is greater than a threshold value θ:

$$\forall i; \|\mathbf{y}_i - \mathbf{x}_k\| > \theta$$

Then, this input vector becomes a new codevector $\mathbf{y}_c + 1 = \mathbf{x}_k$. When the whole sample has been examined without achieving the completion of the codebook obtaining M codevectors, the threshold is halved and the search restarted until M codevectors are found.

In our experiments the threshold value is given by the formula: $d * \theta^2$, with values for $\theta = \{8, 32, \ldots\}$.

Chapter 3

Applications

This chapter describes the main applications that have been considered in the work of this thesis. Some of them have been developed in collaboration with other researchers in the group. We will acknowledge this collaboration and define the contribution of the PhD candidate to the global system.

3.1 Face localization

Most of the systems that dealt with some form of face information processing assume the face localization problem as solved, and work upon very restricted face images, most of them are like mugshots, with little background and constant scale. They impose [55, 254, 275, 279] very severe illumination and position conditions in order to abstract from the tasks of face localization and registration. In essence, face localization is a two class classification problem. The complexity of the face localization task is determined by several conditions: the dynamic aspects of the scene, the relative size of the face on the scene, the background and foreground (clothes) clutter, the variations in scale, pose, and orientation, etc. Face localization remains an active area of research despite the apparent global success reported in some papers. Some of the most successful attempts involve the application of Artificial Neural Networks [184, 237, 238, 256, 275, 279]. The main problem of the application of Neural Networks to this problem is the determination of the train and test datasets, and ensuring its generality. Artificial train patterns are generated to cover the range of transformations against which invariance is desired (translation, rotation, tilt). Non trivial bootstrapping methods [237, 238] are used to cover the non-face instances. Another general concern is that of scale invariance. To obtain some scale invariance the image must be resampled and processed at the expected scales. The conventional solution is the construction of a pyramid of diverse resolution images over which the Neural Network is applied [237, 238]. Other methods proposed for face localization range from the famous eigenfaces [271], Belief Networks [290], Hidden Markov Models [243], Genetic Algorithms [289] , to natural language

parsing of image captions [253] combined with wavelet analysis. Recent works [178, 52, 255, 262] also use the analysis of color for face localization.

The approach presented in [127] does not rely on data or on face models. It is decomposed in two steps, the first tries to localize the head region based on the analysis of the signatures of motion images. The second tries to provide confirmation of the head hypothesis through the color analysis of the head subimage. The color analysis is performed as a color quantization process. The color representatives are computed through an adaptive neural network technique, in fact a supervised version of the well known Kohonen Self Organizing Map. The contribution of this PhD candidate to this work was the development and application of the ideas about using SOM in a supervised way.

3.2 Image vector Quantization

3.3 Color Quantization

Color Quantization [141, 207, 185, 272], is an instance of the more general technique of Vector Quantization (VQ) [97] in the space of colors. Color Quantization has application in visualization, color image segmentation, data compression and image retrieval [156]. The number of color representatives searched is tightly related to the application. In visualization and compression applications the typical size of the color palette (codebook, color representatives) is 256, whereas for segmentation and retrieval tasks the size of the color palette is smaller. We have chosen 16 as a typical number of color representatives for these latter kind of applications. We do not deal with the problem of finding the natural number of colors, which is a much more involved problem. Testing the ability of the proposed algorithms to perform adaptive clustering into a fixed number of clusters is preliminar to find out adaptively the natural number of clusters. Besides, the definition of a numerical measure for the natural clustering problem is highly dependent of the application, and the subject of strong discussion inside the Clustering community.

Color Quantization is a mapping of a multispectral image

$$f(x,y) = [f_R(x,y), f_G(x,y), f_B(x,y)] \in [0,1]$$

into an indexed image $f^c(x,y) \in \{1,\ldots,c\}$ where c is the number of color representatives, which we will denote $\mathbf{Y} = \{\mathbf{y}_1,\ldots,\mathbf{y}_c\}$ with $\mathbf{y}_i \in [0,1]^3$. The visualization of the color quantized image is done through the inversion of the quantization $\hat{f}(x,y) = \mathbf{y}_i \Leftrightarrow f^c(x,y) = i$. Visual comparison with the original image gives the perceptual evaluation of the color quantization. The numerical evaluation of the quality of the color quantization can be done computing the distortion

$$E = \sum_{x,y} \left\| \hat{f}(x,y) - f(x,y) \right\|^2$$

From this description of the Color Quantization process it is obvious that it belongs to the class of Clustering and VQ problems, the sample is given by all or some of the image pixels, the vectors are defined in a 3D space, and the criterion function corresponds to the quantization distortion. The design of Color Quantizers can, therefore, use any of the tools developed for Clustering and VQ, including Competitive Neural Networks. The Heckbert algorithm [141] is a Clustering algorithm that performs a greedy search for the partition of the image colors. In its original formulation it performs a recursive splitting of the RGB space based on the histograms of the projections on the color axis. The axis and the splitting point are selected according to the variance of the histogram and its median. An improvement that ensures minimization of the distortion (although not globally optimal because of its greedy nature) is to consider the variances of the partitions [283] of the cube. This version of the algorithm will be referred onwards as the minimum variance Heckbert algorithm. It gives near optimal results, but its complexity is proportional to the dimension of the space and the discretization of the space axes. Most practical implementations reduce the number of values in each color axis from 2^8 to 2^5 by a direct truncation, before applying the Heckbert algorithm.

3.3.1 Some applications of Color Quantization

Color Quantization has applications in visualization [141, 207, 185, 152], color image segmentation [272], data compression [101, 59] and image retrieval [156]. Early applications of Color Quantization were addressed to visualization tasks [141, 207]. The problem was to render color images for display in low color resolution monitors. This can be of interest for games that require fast visualization or that involve network communication. Nowadays monitors and visualization devices do not require this reduction of the color space. A recent application for CQ is the content-based retrieval of information in multimedia databases that include color images [156, 258, 16, 150]. Usually the color space is partitioned in regular intervals and the color histogram of the image is used as the feature for the search. Color representatives are sometimes used to index the images in the database [156]. These color representatives are computed using clustering based techniques. There are also instances [297] that use the codebooks obtained from Adaptive Vector Quantization for the search in image databases, where the input vectors are subimage blocks. One of the recent applications of color processing is the segmentation of video sequences [297, 57]. The variations in the color histograms are used to identify the units (shots) in the decomposition of the sequence. These units are then used for fast access into the sequence or to construct an index for the organization of video databases. The non-stationary clustering approach could be of use for this task providing that color representatives are used instead of color histograms. This is a stationary application because past images can't be quantized at each database increment. However, given a suitable initial sample of the images to be stored in the database, the techniques discussed in this paper can be applied.

The last class of applications of CQ address the segmentation of images.

This segmentation is of interest for sensitive interfaces: the color detection of faces, hands and other human features can be very effective [187, 178]. The usual approach is the a priori identification in the color space of the region that corresponds to face colors. This approach is very restricted to the images taken to estimate the face color region. The adaptive techniques discussed in the paper can be of interest to allow robust detection in case of illumination changes and other sources of noise. Sensitive interfaces are of increasing interest for Personal Computers an for control applications, such as enhancing the human interface to robots. The optical flow is a very central issue in many computer vision applications including robot navigation. The image segmentation obtained applying non-stationary CQ to the preprocessing of image sequences can be used for the robust computation of the optical flow [123], and to other visión tasks, such as stereo matching [286].

3.4 Vector Quantization for image sequences

Non-stationary Clustering problems assume a time varying population sampled at discrete times. Therefore the Clustering of the data must be recomputed at each time instant. A related problem is that of Adaptive Vector Quantization [97]. The works reported here belong to the class of shifting-mean Adaptive Vector Quantization [99]. Both the Evolution Strategy and the Competitive Neural Networks are applied as adaptive algorithms for the computation of the cluster representatives given by the cluster means at each time instant. Our work fits in the original proposition of Holland [146] of evolution as a mechanism for adaptation to uncertain and varying environment conditions

Color Quantization of images within a sequence contains the essence of the paradigm of real time Non-stationary Clustering. Although sequences of images (video) lead naturally to the consideration of time varying Clustering/VQ problems, the usual approaches to the computation of codebooks for both Color Quantization and Vector Quantization of image sequences consider time invariant distributions of colors [59] or image blocks [56], and apply conventional Clustering methods. Some heuristic efforts [101, 58] have been reported that try to cope with the time varying characteristics inherent to image sequences. Our approach is to assume the problem as a Non-stationary Clustering problem that may be solved by the application of Adaptive VQ algorithms: the Color Quantization of the image sequences become the Dynamic Color Quantization problem. We propose in ensuing chapters Competitive Neural Networks and Evolution Strategies as Dynamic Color Quantization algorithms.

The process of Color Quantization is the codification of a color image into a finite and small set of color representatives. Color Quantization design is the problem of search for a set of optimal color representatives in a color image. The optimality of such a set can be measured perceptually by displaying the inverse of the image codification. It can also be measured quantitatively by computing the distortion of the codification or the Signal-to-Noise Ratio (SNR). In the present paper we present as experimental results both qualitative and numerical results

(Distortion). From a numerical point of view, the search for the optimal color representatives can be put into the general framework of Clustering based on representatives or Vector Quantization design [97, 138, 73, 76, 148, 89]. Color Quantization is then a Vector Quantization defined in the color space, usually based on the Euclidean distance in this space. One relevant question is that of the color space and the color distance employed. It is well known that the Euclidean distance in the RGB space does not preserve the perceptual distance between colors. There are several abstract color spaces [21], such as the Yuv and the Lab spaces, defined by the CIE in order to obtain better preservation of the perceptual distance, and new color spaces are being defined in order to cope with the color equivalencies needed to support color processing in complex distributed environments (i.e. Internet). It can be argued that performing Clustering based on minimizing the Euclidean Distortion in the RGB color space is doomed to give perceptually suboptimal results. However, there is a growing body of evidence [221, 141, 152, 285, 187] showing that this perceptual suboptimality is of no consequence for most practical applications. We assume this framework to support our experiments in the RGB space.

3.4.1 Color quantization of image sequences

Given an image sequence $\{f_t(x,y); t = 1, 2, , \ldots\}$, the Color Quantization of this sequence is obviously a Non-Stationary Clustering problem, in which the searched partitions are the color quantizations of the images in the sequence $\{f_t^c(x,y); t = 1, 2, , \ldots\}$ and the infinite horizon criterion function is the accumulative color quantization distortion

$$E = \sum_{t \geq 0} E(t) = \sum_{t \geq 0} \sum_{x,y} \left\| \hat{f}_t(x,y) - f_t(x,y) \right\|^2 .$$

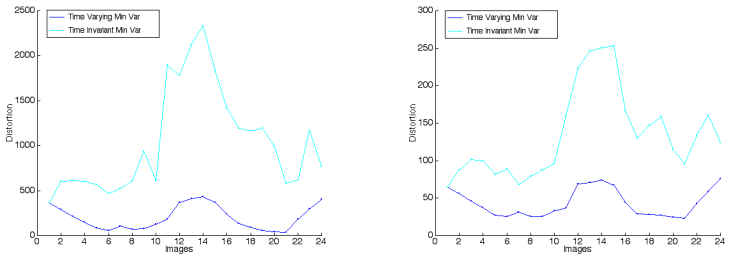
The Color Quantization of the image sequence applies a sequence of color palettes $\mathbf{Y}(t) = \{\mathbf{y}_1(t), \ldots, \mathbf{y}_c(t)\}$, and it becomes Non-Stationary Color Quantization when no predictive scheme can be applied in the computation of the color palettes. The search for these color representatives varying in time becomes an Adaptive Vector Quantization problem .

In a rather general statement the Non-Stationary Clustering problem is, thus, an infinite time stochastic dynamic programming problem

$$\min_{\{\mathbf{Y}(t)\}} \sum_{t \geq 0} E(t) .$$

To alleviate its complexity the adaptive assumption states that the problem can be decomposed in time, so the global minimization can be achieved through instantaneous minimization: .

$$\min_{\{\mathbf{Y}(t)\}} \sum_{t \geq 0} E(t) = \sum_{t \geq 0} \min_{\{\mathbf{Y}(t)\}} E(t) .$$



(a) (b)

Figure 3.1: Benchmark distortion values obtained with the application of the Matlab implementation of the Heckbert algorithm to compute the color quantizers of 16 (a) and 256 (b) colors of the images in the experimental sequence.

The straightforward approach, under this assumption, is to perform independent optimizations at each time instant. Independent optimization would, then, give an optimal result if the optimization method is globally optimal. We have applied this approach as the benchmark optimal results in our experiments applying the minimum variance Heckbert algorithm to each image independently. However, time independent optimization discards any use of time dependencies that could improve the accuracy and lower the computational burden. Truly adaptive algorithms must profit from these unknown time dependencies, without trying to uncover them. The adaptive approach assumes that the cluster or color representatives found at time $t-1$ can be used as good initial conditions for the optimization process at time t. That implies the assumption of smooth variation of the optimal cluster representatives. Summarizing, the application of Competitive Neural Networks as Adaptive Vector Quantization algorithms is done as follows: At time t the initial cluster (color) representatives are the ones computed from the sample of the process at time $t-1$. The sample vectors at the present time are presented sequentially as inputs to compute the adaptation equations and to obtain a new set of cluster representatives. Obviously, for the Non-Stationary Color Quantization case the time axis is the image number in the sequence, and the sample data are the image pixel colors. To approach real time performance we impose a one-pass adaptation at each time step, and small samples. That means that the sample vectors will be presented only once and that the scheduling of the learning rate and other learning control parameters are adjusted to that time constraint.

3.4.2 Benchmarks

The sequence of images used for the computational experiments is presented in Appendix . As a benchmark non adaptive algorithm we have used a variation of the algorithm proposed by Heckbert [141] as implemented in MATLAB. As a benchmark non adaptive algorithm we have used a variation of the algorithm

proposed by [141] as implemented in MATLAB, we call this algorithm Minimum Variance Heckbert algorithm. This algorithm recursively partitions the RGB unit cube along the axis of maximum variance. The partition is performed by a plane orthogonal to the color axis chosen so as to minimize the sum of the residual variances. A time efficient method to compute the residual variances was presented in [283]. This method is implemented in many standard libraries. However, the algorithm involves the pre-computation of all the potential variances. Its time and space complexity grows exponentially with the dimension of the space, and the number of values of the discretization of each space axis. Color representatives are computed as the center of mass of the resulting partition cubes. This algorithm has been applied to the entire images in the sequence in two ways.

This algorithm has been applied to the entire images in the sequence under stationary and non-stationary assumptions. Figure 3.1 shows the distortion results of the Color Quantization of the experimental sequence to 16 and 256 colors based on both applications of the Heckbert algorithm. The curve $\left\{C^{TV}(t)\,; t = 1, \ldots, 24\right\}$, named *Time Varying Min Var* in the figure is produced assuming the non-stationary nature of the data and applying the Heckbert algorithm to each image independently. The curve $\left\{C^{TI}(t)\,; t = 1, \ldots, 24\right\}$, named *Time Invariant Min Var* in the figure, is computed under the assumption of stationarity of the data: the color representatives obtained for the first image are used for the Color Quantization of the remaining images in the sequence. The gap between those curves gives another indication of the non stationarity of the data. Also this gap defines the response space left for truly adaptive algorithms. All the figures giving distortion results for the experimental sequence will include these two curves as a reference frame.

3.5 Image filtering using Vector Quantization

VQ has been proposed for digital image processing [65]. It has been suggested that the encoding/decoding process introduce some non-linear smoothing of the image that removes some kinds of noise, especially speckle noise. We combine the Bayesian approach to image processing [94, 149] with idea of VQ-based filtering. In Bayesian image processing, given an observed image G we try to find the resotored/segmented image F. The *a posteriori* probability density is computed from the a priori probability model and the *likelihood* model applying Bayes' rule. The desired result F is usually computed as a either the *maximum a posteriori* (MAP) or *maximum likelihood* (ML), which are the modes of $p(F = f\,|G = g)$ and $p(G = g\,|F = f)$, respectively. In general, it is difficult to obtain the marginal density $p(G = g)$, however, the MAP and ML estimates do not require it. Both estimation methods need to postulate models for the prior $p(F = f)$ and likelihood $p(G = g\,|F = f)$ probability distributions. The prior model specifies a broad class of images through the specification of probabilistic relationships. The conditional probabilities can be postulated as a model of the image degradations or of the transformation between the observed and desired

image.

In the VQ Bayesian filter (VQBF) the codevectors are considered as convolution masks, with the coordinate system origin centered in their middle pixels. For VQBF processing, the image is not decomposed into blocks, rather we consider at each pixel a neighborhood window of the same size as the codevectors. The filtering process at each pixel is computed in two steps

- find the best matching codevector for the pixel's neighboring window,
- the processed pixel takes the value of the codevector's central pixel.

Thus, codevectors are a kind of probabilistic models of the pixel neighborhood. To understand VQBF in the framework of Bayesian image processing, the filtering application of the codebook must be interpreted as realizing some kind of approximation of the posterior probability distribution of the desired image result, so that VQBF corresponds to some kind of MAP estimation.

In our works we do consider that the codebook design intends to minimize the Euclidean distortion. The VQ design method employed is not relevant class now as long as it minimizes it. A well-known interpretation [23], in terms of statistical decision theory, of the minimization of the Euclidean distortion beside is as follows: assume a number of classes and feature vectors whose probability density follows a mixture of class conditional densities. Further, assume that the class conditional densities are Gaussian with identical unit covariance matrices , and that the classes are equiprobable, then the minimization of the Euclidean distortion is equivalent to maximum log-likelihood estimation of the parameters of the model, the class means. Based on these parameters the MAP decision is the Bayesian minimum risk decision. Thus, the filtering realized by VQBF corresponds to a MAP image process, in which the classes are the gray levels of the central pixel in the representative neighborhoods extracted from the image. The detailed Bayesian analysis of VQBF properties deduces from the statement of the posterior probabilities the prior and conditional models. This analysis is presented in Appendix

Part II

Convergence and properties for unsupervised learning and clustering

Chapter 4

Generalized Learning Vector Quantization

Generalized Learning Vector Quantization (GLVQ) has been proposed in [209] as a generalization of the Simple Competitive Learning (SCL) algorithm. The main argument of GLVQ proposal is its superior insensitivity to the initial values of the weights (codevectors). We show that the distinctive characteristics of the definition of GLVQ disappear outside a small domain of applications. GLVQ becomes identical to SCL when either the number of codevectors grows or the size of the input space is large. Besides that, the behaviour of GLVQ is inconsistent for problems defined on very small scale input spaces. The adaptation rules fluctuate between performing descent and ascent searches on the gradient of the distortion function.

4.1 Introduction

The GLVQ algorithm has been proposed in [209] as a generalisation of the LVQ algorithm. However, the definition of LVQ given in [209] really corresponds to what is usually known as SCL in the neural network literature. GLVQ proposition is related to the Self-Organizing Map (SOM), (due to Kohonen [168]) in the sense that the updating of a neighbourhood of the winning codevector is the mechanism used to escape from the local minima of the distortion. The authors claim that all the codevectors will be updated using the GLVQ rules, unless there is a perfect matching of the input and winning codevector ($\|\mathbf{x}(t) - \mathbf{y}_i(t)\|^2 = 0$). In which case, the authors state that GLVQ reduces to SCL. We show in section 4.3.1, that GLVQ is identical to SCL, even in the case of very imperfect matching, for clustering problems defined in non-small spaces or that imply the searching for a non-small number of clusters. On the other hand, in section 4.3.2 it is shown empirically that perfect matching is not a sufficient condition for GLVQ to become identical to SCL. Even worse situations may happen. For some very small input spaces, GLVQ may become

inconsistent. Section 4.4 shows how inside very small input spaces GLVQ rules may fluctuate between going down and up the distortion function.

The derivation of GLVQ follows from the proposition of a loss function, closely related to the distortion. The codevector updating rules are derived as a local (stochastic) gradient descent along this loss function. Some parallelism can be found in this method of definition, and learning rules, with Soft Competition [288, 267]. In the case of Soft Competition, the purpose is to minimise the cross entropy (Kullback distance) of the input distribution and the distribution modelled by the codevectors, taken as a mixture of Gaussian distributions whose means are the codevectors. As with GLVQ, the Soft Competition rules update all the codevectors after the presentation of each input vector. Soft-Competition uses as neighbourhood coefficients the estimations of the conditional probabilities of the input for classes represented by the codevectors. Soft Competition has the additional problem of the estimation of the variances of the Gaussian distributions, but it does not share the limitations of GLVQ that will be discussed in this letter. It will become clear that much of the trouble is due to the definition of the loss function, as we refer to it looking for explanations of undesired limit forms of the GLVQ rules.

Section 4.2 gives the definitions of SCL, LVQ and GLVQ. Section 4.3 gives the conditions for GLVQ to behave identically to SCL. Section 4.4 discusses the conditions for inconsistent behaviour of GLVQ. Section 4.5 summarises our conclusions.

4.2 Definition of SCL, LVQ and GLVQ

We denote $\mathbf{x}(t)$ the input vector at time t, $\mathbf{y}_i(t)$ are the codevectors (weights) at time t, $\mathbf{Y}$ is the whole codebook, $\alpha(t)$ is the learning rate (gain), and $\triangle\mathbf{y}_i(t)$ is the adaptation of $\mathbf{y}_i(t)$ after presentation of input $\mathbf{x}(t)$. The definition of LVQ given in [209] coincides with what is known in the neural networks' literature as the Simple Competitive Learning (SCL) rule [168, 142, 173, 169]:

$$\triangle\mathbf{y}_i(t) = \begin{cases} \alpha(t)\left(\mathbf{x}(t) - \mathbf{y}_i(t)\right) & i = \arg\min_k \left\{\|\mathbf{x}(t) - \mathbf{y}_k(t)\|^2\right\} = w \\ 0 & otherwise \end{cases} \tag{4.1}$$

SCL is the basic form of unsupervised competitive learning, and can be derived as a stochastic gradient descent of the distortion:

$$J_e = \sum_{j=1}^{N}\sum_{i=1}^{M} \|\mathbf{x}_j - \mathbf{y}_i\|^2 \delta_{ij}$$

$$\delta_{ij} = \delta(\mathbf{x}_j, \mathbf{y}_i) = \begin{cases} 1 & i = \arg\min_k \left\{\|\mathbf{x}_j - \mathbf{y}_k\|^2\right\} \\ 0 & \text{otherwise} \end{cases}$$

where N is the sample size and M is the number of codevectors. The LVQ algorithm was introduced by Kohonen [168, 169] as a supervised learning algorithm, which assumes *a priori* knowledge about the classes to which the input

vector and the winning codevector belong. Allowing $c(\mathbf{x})$ to denote the class to which $\mathbf{x}$ belongs, then the formal definition of LVQ reads as follows:

$$\triangle \mathbf{y}_i(t) = \begin{cases} \alpha(t)(\mathbf{x}(t) - \mathbf{y}_i(t)) & \left(i = \arg\min_k \left\{\|\mathbf{x}(t) - \mathbf{y}_k(t)\|^2\right\}\right) \wedge c(\mathbf{x}(t)) = c(\mathbf{y}_i(t)) \\ -\alpha(t)(\mathbf{x}(t) - \mathbf{y}_i(t)) & \left(i = \arg\min_k \left\{\|\mathbf{x}(t) - \mathbf{y}_k(t)\|^2\right\}\right) \wedge c(\mathbf{x}(t)) \neq c(\mathbf{y}_i(t)) \\ 0 & \text{otherwise} \end{cases}$$

In the following, we will refer to SCL instead of LVQ. We will assume that all the references to LVQ made in [209] were intended to SCL. This means that the main purpose of the definition of GLVQ was to improve Simple Competitive Learning trying to make it insensitive to initial conditions.

The GLVQ codevector adaptation rules are formally written as follows (with a slight change in notation from the formulas in [209]):

$$\triangle \mathbf{y}_i(t) = \begin{cases} \alpha(t)(\mathbf{x}(t) - \mathbf{y}_i(t)) C_1 & i = \arg\min_k \left\{\|\mathbf{x}(t) - \mathbf{y}_k(t)\|^2\right\} = w \\ \alpha(t)(\mathbf{x}(t) - \mathbf{y}_k(t)) C_2 & \text{otherwise} \end{cases}$$

$$C_1 = 1 - \frac{1}{D(t)} + C_2$$

$$C_2 = \frac{\|\mathbf{x}(t) - \mathbf{y}_w(t)\|^2}{D^2(t)}$$

$$D(t) = \sum_{k=1}^{M} \|\mathbf{x}(t) - \mathbf{y}_k(t)\|^2$$

The values of the coefficients C_1 and C_2 perform the switch between the GLVQ and SCL: $C_1 = 1$ and $C_2 = 0$ imply that GLVQ is identical to SCL. According to [209], the formal derivation of GLVQ follows from the stochastic gradient descent of a loss function defined as follows.

$$L = \sum_{j=1}^{N} \sum_{i=1}^{M} \|\mathbf{x}_j - \mathbf{y}_i\|^2 g_i(\mathbf{x}_j, \mathbf{Y})$$

$$g_i(\mathbf{x}_j, Y) = \begin{cases} 1 & \delta_{ij} = 1 \\ \left[\sum_{k=1}^{M} \|\mathbf{x}_j - \mathbf{y}_k\|^2\right]^{-1} & \text{otherwise} \end{cases}$$

4.3 Convergence of GLVQ to SCL

We use the term *convergence* meaning the identity of behaviour of both algorithms under certain conditions. The main statement of [209] is that GLVQ is insensitive to initial conditions (initial codevectors) and that, so, it behaves better than Simple Competitive learning for clustering applications. The authors did not realise that GLVQ becomes identical to SCL algorithm when some

parameters of the clustering problem grow, and that this happens rather unexpectedly. On top of that, the conditions assumed in [209] for GLVQ convergence to SCL appear to be insufficient. Two extreme situations condition the convergence landscape: imperfect and perfect matching. In the first case, the desired behaviour of GLVQ is the significant updating of all the codevectors, whereas SCL only updates the winner. In the second case, the desired behaviour is the same as SCL. In both cases, GLVQ can behave against the desires of their designers. In this section we discuss when and why this happens.

4.3.1 Imperfect matching: undesired convergence

Let us assume the occurrence of imperfect matching in the sense that all the codevectors are almost at the same distance of the input. This situation roughly corresponds to the initialisation of the codebook with values right outside of the convex hull of the sample, or when the input is a wildshot (far away from any other input). In this case, the distance of the input to the winner codevector ($\mathbf{y}_w(t)$) is close to the distance to any other codevector:

$$\|\mathbf{x}(t) - \mathbf{y}_w(t)\|^2 \cong \|\mathbf{x}(t) - \mathbf{y}_k(t)\|^2$$

Then, the normalisation factor $D(t)$ can be assumed to be approximated as follows:

$$D(t) \cong M \|\mathbf{x}(t) - \mathbf{y}_w(t)\|^2 \tag{4.2}$$

Under these approximations, the expression of the C_1 and C_2 coefficients can be rewritten as follows:

$$C_1 \cong 1 - \frac{1}{M \|\mathbf{x}(t) - \mathbf{y}_w(t)\|^2} + C_2$$

$$C_2 \cong \frac{1}{M^2 \|\mathbf{x}(t) - \mathbf{y}_w(t)\|^2} \tag{4.3}$$

In the case of imperfect matching, GLVQ is meant to behave quite differently from SCL, that is: C_1 must be significantly smaller than 1 and C_2 must be significantly larger than 0. However, as the number of clusters or the expected magnitude of the distance grows, it is obvious from the above expressions that C_1 converges to 1. Simultaneously, C_2 converges even faster to 0.

$$C_1 \underset{\substack{M \to \infty \\ \|.\|^2 \to \infty}}{\longrightarrow} 1$$

$$C_2 \underset{\substack{M^2 \to \infty \\ \|.\|^2 \to \infty}}{\longrightarrow} 0$$

These expressions summarise the sensitivity of GLVQ to the number of classes (codevectors) and the average magnitude of the distance on the input

space. Increasing the number of classes forces the undesired convergence GLVQ to SCL. Large input spaces imply large values of $\|\mathbf{x}(t) - \mathbf{y}_w(t)\|^2$ which also force this convergence. The effect of large input spaces can be also deduced from a close look at the loss function that GLVQ tries to minimise. It can be seen that $L_{\mathbf{x}}$ is bounded as follows:

$$J_e(\mathbf{x}) < L_{\mathbf{x}} \leq J_e(\mathbf{x}) + 1$$

Where

$$J_e(\mathbf{x}) = \sum_{i=1}^{M} \|\mathbf{x} - \mathbf{y}_i\|^2 \delta(\mathbf{x}, \mathbf{y}_i)$$

is the distortion introduced by quantizing input $\mathbf{x}$ with the codevector $\mathbf{w}_i$. When $J_e(\mathbf{x}) \gg 1$ the loss function becomes the distortion, and then, the stochastic gradient descent on $L_{\mathbf{x}}$ becomes the SCL. In some clustering applications such as Vector Quantization of grey level images, $[0, 255]^{4\times4}$ or $[0, 255]^{8\times8}$ are standard input spaces. Obviously, even for extraordinarily good codebooks, in those applications $J_e(\mathbf{x}) \gg 1$ holds. The above discussion implies that GLVQ can become identical to SCL in the case of imperfect matching. Outside some restricting conditions (small number of classes and small scale input spaces) GLVQ loses its more appealing property (the insensitivity to initial conditions) because it unexpectedly becomes identical to SCL.

4.3.2 An illustrative experiment

A simple experiment was carried out to illustrate the foregoing analysis, and to explore the sensitivity of the GLVQ coefficients to the number of codevectors and the scale of the input space. Simulated samples from a mixture of four Gaussian distributions were generated as the sample input. Although the authors in [209] give some discussions about learning rate scheduling, we have found that this is a matter of secondary importance, that does not influence the inherent qualitative features of the algorithms.

The first experiment was addressed to assess the convergence of GLVQ to SCL as the number of codevectors grows. The sample points generated fall inside $[-0.6, 1.2]^2$ and the initial codebooks (+ in the figures) were generated in a corner of this region. The expected behaviour of GLVQ was to update all the codevectors driving them inside the sample. Figure 4.1(a) shows the trajectories of the codevectors as updated by GLVQ in the case of $M = 2$. Figure 4.1(b) shows the trajectories under SCL. The difference in behaviour of both algorithms, putting aside the quality of the final solution, can be easily verified. This difference disappears as the number of codevectors increases. In the case of $M = 10$, the behaviour of GLVQ (shown in Figure 4.1(c)), and that of SCL (shown in Figure 4.1(d)) are almost identical. The stuck codevectors are minimally displaced by GLVQ. When the sample is scaled by a factor of 5, the identity of behaviour of GLVQ and SCL can be appreciated even with only two codevectors $M = 2$. Figure 4.2(a) and 4.2(c) show the behaviour of GLVQ

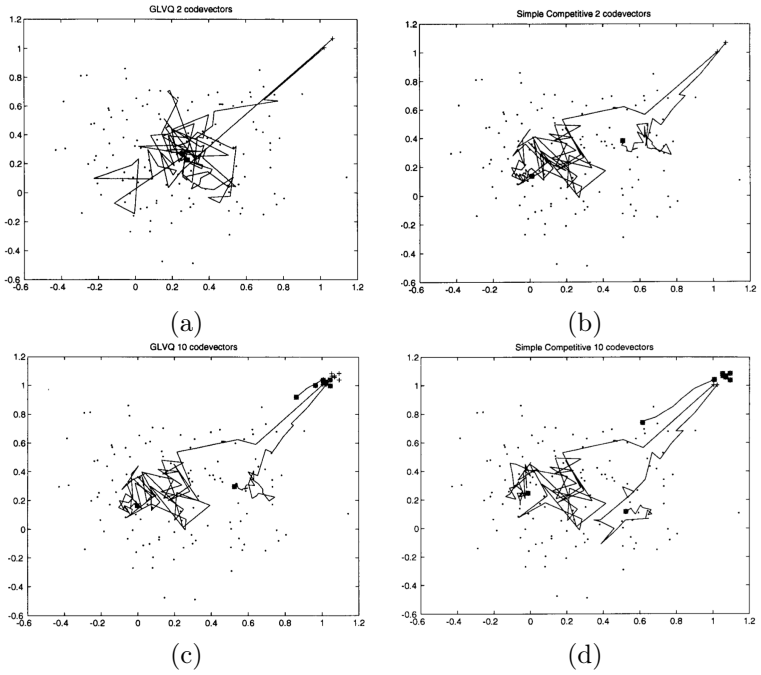


Figure 4.1: Convergence in a toy example

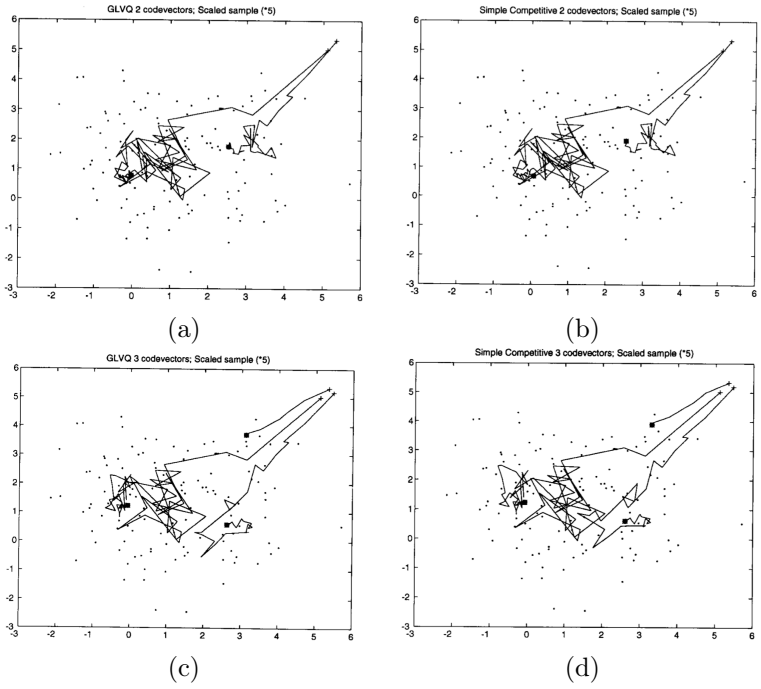


Figure 4.2: Undesired convergence in the toy example

with $M = 2$ and $M = 3$, resp. Figure 4.2(b) and 4.2(d) show the behaviour of SCL.

4.3.3 (Almost) Perfect matching: failed desired convergence

Let us assume the occurrence of (almost) perfect matching of input and winning codevector. Perfect matching can be naively characterised by the following expression ($\mathbf{y}_w(t)$ is the winner codevector):

$$\|\mathbf{x}(t) - \mathbf{y}_w(t)\|^2 \cong 0$$

The authors in [209] state that this is the only condition that can lead to $C_1 = 1$ and $C_2 = 0$ corresponding to desired convergence of GLVQ to SCL. Let us examine the approximate expressions taken by the coefficients under this assumption:

$$C_1 \cong 1 - \frac{1}{D(t)}$$

$$C_2 \cong 0$$

Another way to characterise the almost perfect matching comes from the consideration of the relative magnitudes of the distances of the codevectors to the input ($\mathbf{y}_w(t)$ is the winning codevector):

$$\forall k \neq w; \|\mathbf{x}(t) - \mathbf{y}_k(t)\|^2 \gg \|\mathbf{x}(t) - \mathbf{y}_w(t)\|^2$$

It can be easily verified that this more general characterisation of almost perfect matching leads to the same expressions for C_1 and C_2. It becomes obvious, that the magnitude of $D(t)$ determines if $C_1 \cong 1$, so, (almost) perfect matching is not a sufficient condition for GLVQ to become SCL. Other problem dependent conditions, such as the number of codevectors and the scale of the input space, and the instantaneous positions of the whole codebook determine the value of $D(t)$ and, therefore, the desired convergence of GLVQ to SCL, in the case of almost perfect matching.

4.4 GLVQ inconsistent behaviour

Studying the desired convergence of GLVQ to SCL, we have found conditions for inconsistent behaviour of GLVQ. It is expected that any codebook design algorithm will perform a gradient descent on the distortion function. Inside very small input spaces GLVQ becomes (temporarily) a gradient ascent search on the distortion function, depending on the instantaneous value of $D(t)$. In general, $D(t)$ must be greater than 1 for C_1 to be positive. Positive values of C_1 make the GLVQ learning rule a gradient descent of the distortion, whereas negative values make it a gradient ascent of the distortion. Let us examine the two extreme situations of perfect and imperfect matching.

In the approximate expressions obtained in the case of perfect matching, $D(t) < 1$ clearly imply negative values of C_1. The expression of the GLVQ rules under these circumstances (compare with (4.1) above) becomes:

$$\triangle \mathbf{y}_i(t) = \begin{cases} -\alpha(t)(\mathbf{x}(t) - \mathbf{y}_i(t)) \left|1 - \frac{1}{D(t)}\right| & i = \arg\min_k \left\{\|\mathbf{x}(t) - \mathbf{y}_k(t)\|^2\right\} = w \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

In the case of imperfect matching, the approximate expressions of C_1 and C_2 in terms of $D(t)$, deduced from (4.2) and (4.3), are:

$$C_1 \cong 1 - \frac{M-1}{M \cdot D(t)}$$

$$C_2 \cong \frac{1}{M \cdot D(t)}$$

The expression of the GLVQ rules when become

$$\triangle \mathbf{y}_i(t) = \begin{cases} -\alpha(t)(\mathbf{x}(t) - \mathbf{y}_i(t)) \left|1 - \frac{M-1}{M \cdot D(t)}\right| & i = \arg\min_k \left\{\|\mathbf{x}(t) - \mathbf{y}_k(t)\|^2\right\} = w \\ \alpha(t)(\mathbf{x}(t) - \mathbf{y}_k(t)) \frac{1}{M \cdot D(t)} & \text{otherwise} \end{cases}$$

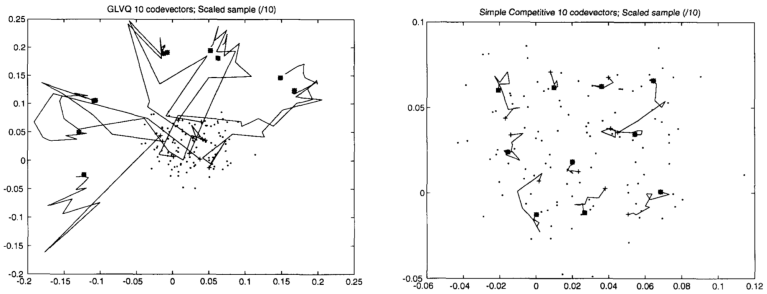
Both expressions of the GLVQ learning rule maximise the distortion instead of minimising it. The behaviour of GLVQ becomes inconsistent for problems defined in very small hypercubes, where the probability of finding $D(t) < 1$ is significant (e.g. $\mathbf{x} \in [0, 1/M]^N$).

Let us return to the loss function that originates GLVQ in search for some explanation of this inconsistent behaviour. Note that it can be rewritten as follows:

$$\begin{aligned} L_{\mathbf{x}(t)} &= \|\mathbf{x}(t) - \mathbf{y}_w(t)\|^2 + \frac{1}{D(t)} \sum_{\substack{k=1 \\ k \neq w}}^{M} \|\mathbf{x}(t) - \mathbf{y}_k(t)\|^2 \\ &= J_e(\mathbf{x}(t)) + \frac{1}{D(t)} \bar{J}_e(\mathbf{x}(t)) \end{aligned}$$

The GLVQ loss function is composed of the conventional distortion $J_e(\mathbf{x}(t))$, plus some kind of global distortion $\bar{J}_e(\mathbf{x}(t))$ defined over the non-winning codevectors, weighted by $D^{-1}(t)$. As long as $D(t) > 1$, the main term in $L_{\mathbf{x}(t)}$ is the distortion $J_e(\mathbf{x}(t))$. However, when $D(t) < 1$, the main term in $L_{\mathbf{x}(t)}$ is $\bar{J}_e(\mathbf{x}(t))$. Then the minimisation of the loss function goes after the minimisation of the "global codebook distortion" instead of (and even against to) the minimisation of the true distortion. In fact, the expression in equation 4.4 tries to decrease $L_{\mathbf{x}(t)}$ through the increase of $D(t)$, something unexpected for an algorithm which is meant to search optimal codebooks.

Sometimes the algorithm will enter an oscillatory behaviour alternating gradient descent steps, which the distortion (and $D(t)$) decrease, with gradient



(a) (b)

Figure 4.3: Inconsistent behavior of GLVQ in the toy example obtained by scaling the data

ascent steps, which increase the distortion and $D(t)$. To illustrate the inconsistent behaviour of GLVQ we scaled by a 0.1 factor the sample used in section 4.3.2. We have applied both GLVQ and SCL to this shrunk sample. This time the initial codebook was a set of sample vectors. This choice of initial codebook highlights the unfitness of GLVQ. Figure 4.3(a) shows the trajectories of the codevectors under GLVQ. It can be appreciated the repulsive and attractive phases of GLVQ rules. At first, the codevectors are expelled from the sample. The expulsion is followed by a slow return, until they take positions on the surface of a kind of "repulsion ball" that the GLVQ dynamics seems to generate around the sample. Figure 4.3(b) shows the trajectories of the codevectors under SCL following natural (distortion gradient descent) patterns. Finally, going back to Figure 4.1(a), we have verified that its strange effects is due to the fluctuation of $D(t)$ around one.

4.5 Conclusions

In spite of the elegance of its definition and its ease of computation, GLVQ must be taken with caution from the practical point of view. A previous analysis of the problem that is intended to solve can show if GLVQ preserves in fact its good qualities (insensitivity to initial codebooks), or if it will behave identically to Simple Competitive Learning, or even if the application is impossible.

We have discussed the GLVQ sensitivity to the number of clusters and the input space size. For a small space, we have shown empirically that a not very large number of clusters is enough to lose the GLVQ characteristics. Also, when the input space is enlarged, the number of clusters that make GLVQ identical to SCL decreases drastically. We have shown how very small input spaces can force the inconsistent behaviour (maximise the distortion) of the GLVQ rules. Empirically, the alternance of behaviours of GLVQ appears under mild conditions. Also, we have seen estrange phenomena such the existence of "repulsion balls" at whose core is the sample. All these undesired results are

related to the definition of the loss function, from which the GLVQ rules are derived as a stochastic gradient descent.

Chapter 5

Basic Competitive Neural Networks as Adaptive Mechanisms for Non-Stationary Color Quantization

In this chapter consider the application of two basic Competitive Neural Networks (CNN) to the adaptive computation of color representatives on image sequences that show non-stationary distributions of pixel colors. The tested algorithms are the Simple Competitive Learning (SCL) and the Frequency-Sensitive Competitive Learning (FSCL). Both, SCL and FCSL are the simplest adaptive method based, respectively, on minimizing the distortion and on the search for a uniform quantization. The aim of this paper is to study several computational properties of these methods when applied to Non-Stationary Clustering (NSC) as Adaptive Vector Quantization (AVQ) algorithms. We study experimentally the effect of the size of the image sample employed in the one-pass adaptation, their robustness to initial conditions and the effect of local versus global scheduling of the learning rate. Results were published in [108].

Section 5.1 is an introduction to the chapter. Section 5.2 reviews FSCL training algorithm. Section 5.3 presents their specific numerical settings for this application. Section 5.4 presents experimental results. Finally, in Section 5.5 we present our conclusions.

5.1 Introduction

When an image sequence is considered, the distribution of the pixel colors in the color space will be time variant in the general case. The underlying stochastic process is, therefore, non stationary and we can't assume any model of the time dependencies. The problem of Color Quantization on image sequences becomes a Non-Stationary Clustering problem. In most of the formulations of the Clustering or Vector Quantization problems, found in the literature, the assumption is that the underlying stochastic process is stationary and that a given set of sample vectors properly characterizes this process. The main line of research that takes into account non stationary processes is that of Adaptive Vector Quantization (AVQ) [97], where the dominant approach is that of codebook replenishment [58, 86]. We will introduce in section 4 a general formulation of the Non-Stationary Clustering problem, and the application of Competitive Neural Networks as Adaptive Vector Quantization methods to solve it.

Competitive Neural Networks [192, 3, 142, 172, 272] are mathematically derived as stochastic gradient minimization procedures. Sometimes the objective function is known, such as it is the case of the Simple Competitive Learning that minimizes the Euclidean distortion. Sometimes it is difficult to specify the objective function and to provide a formal derivation of the learning rule. This is the case of the Frequency-Sensitive Competitive Learning, which combines the distortion minimization of the SCL with the search for a uniform quantization. Uniform quantization implies that the probability distribution of the codevectors is uniform (not to be confused with a uniform decomposition of the color space). To obtain an uniform quantization, FSCL tries to ensure that the sizes of the clusters found in the sample are equal. However it does that penalizing the Euclidean distance while computing the nearest codevector to a sample vector. This can't be easily put into the formal framework of stochastic gradient algorithms, but it constitutes a minimal variation of SCL intended to improve its robustness (to avoid empty clusters associated with stuck codevectors). In this paper we consider both SCL and FSCL as minimal adaptive algorithms whose results can be extrapolated to more sophisticated strategies.

Despite their original definition as stochastic gradient minimization methods, CNN have been little applied to Adaptive Vector Quantization, because of their lengthy convergence times, and their numerical sensitivities [97]. Most works in the literature report their application to stationary VQ problems, such as the codification of still images. As we have said before, the most successful Adaptive Vector Quantization strategy is that of codebook replenishment [97, 101, 58, 90, 86] whose optimality, however, has only been proved for stationary sources [295]. Codebook replenishment algorithms also pose serious parameter tuning problems that have not been properly addressed in the literature. In this paper we try to show the usefulness of the CNN as Adaptive Vector Quantization algorithms for general non-stationary sources. The case of Non-Stationary Color Quantization is, thus, a representative of the general AVQ problem. Our approach is to propose a fast adaptation schedule, based on a one-pass adaptation over a small sample of each image in the sequence. Under

this scheme, the computational cost of the adaptation is proportional to the size of the sample, the size of the codebook and the dimension of the search space, and can be calibrated for real time processing. In section 5, we explore the sensitivity of the SCL and FSCL in this setting to the local/global scheduling of the learning rate, the sample size, the codebook size and the global initial conditions (the starting codebook for the whole sequence). The results reported here will be also of interest when trying to asses the applicability of other CNN architectures to the Non-Stationary Clustering problem. We have found that the sensitivity to the codebook size is shared by the Self Organizing Map of Kohonen, Neural Gas, Soft Competition, and other CNN [112, 110, 102].

5.2 FSCL algorithm

The SCL is, therefore, a local minimization algorithm, whose results will be highly dependent of the initial conditions $\mathbf{Y}(0) = \{\mathbf{y}_1(0), \ldots, \mathbf{y}_c(0)\}$. One of the more salient features of suboptimal solutions is the occurrence of stuck codevectors, that is, codevectors whose corresponding cluster is empty. These codevectors never win the competition and SCL adaptive rule never applies to them. This situation must not be confused with the search for the natural number of clusters, suboptimal solutions found by SCL must not be taken as indicative of the "true" number of clusters. The reasoning behind the proposition of FSCL is that one sure strategy to avoid bad local suboptimal solutions is to ensure that no representative has an associated empty cluster. This is done in an inverse way, by penalizing the bigger clusters. To achieve that, the FSCL [3] keeps a count of how frequently each codevector is the winner and use this information to adjust distances from an input to all codevectors. The distance used to determine the codevector to be updated is

$$d\left(\|\mathbf{x}_j - \mathbf{y}_i\|^2\right) = \tau_i \|\mathbf{x} - \mathbf{y}_i\|^2 \tag{5.1}$$

where

$$\tau_i = \sum_{k=1}^{\tau} \delta_i^{FSCL}(\mathbf{x}(k))$$

is the number of times that a codevector has been the winner and

$$\delta_i^{FSCL}(\mathbf{x}(k)) = \begin{cases} 1 & i = \arg\min\limits_{j=1,\ldots,c} \left\{\tau_i \|\mathbf{x}(k) - \mathbf{y}_j\|^2\right\} \\ 0 & \text{otherwise} \end{cases} . \tag{5.2}$$

This new distance of eq. 5.1 penalizes the codevector that repeatedly wins increasing his distance value and giving other codevectors a chance to win the competition. FSCL employs the learning rule of SCL, but applying eq. 5.2 instead of the conventional nearest neighbor rult to determine the neuron (codevector) to be updated.

The preceding discussion implies that FSCL can not be derived as a stochastic gradient minimizer of the quantization distorsion. In fact, we haven't found

a formal definition of the true function minimized by FSCL. This function may be intuitively seen as a minimization of the Euclidean distortion conditioned to the uniform quantization of the space. That is, FSCL tries to ensure after training that

$$P\left(\delta_i\left(\mathbf{x}\right)=1\right)\cong\frac{1}{c}$$

. Once near uniform quantization is achieved, FSCL will try to minimize the Euclidean distortion. The expected side effect is that the minimization of the Euclidean distortion will become globally optimal. We consider FSCL as a basic CNN because it is the minimal variation of SCL proposed up to date.

5.3 On the application of SCL and FSCL

In the experiments we have applied two kind of scheduling of the learning rate: local and global scheduling. The local scheduling of the learning rate follows the expression:

$$\alpha_i\left(\tau\right)=\alpha_0\left(1-\frac{\tau_i}{n}\right)$$

where

$$\tau_i=\sum_{k=1}^{\tau}\delta_i\left(\mathbf{x}\left(k\right)\right)$$

and n is the subsample size. This expression implies that the learning rate decreases linearly in the number of times that a codevector "wins" the competition. It also implies that a local learning rate only reaches the zero value if the codevector "wins" for all the sample vectors. This local counter is identical to the one maintained by the FSCL. Note however, that here the counter controls the adaptation gain, whereas in FSCL it determines the winning neuron. There is no interference between the two applications of the local counter.

The global scheduling of the learning rate, meaning that $\forall i;\alpha_i\left(\tau\right)=\alpha\left(\tau\right)$, follows the expression:

$$\alpha\left(\tau\right)=\alpha_0\left(1-\frac{\tau}{n}\right).$$

This expression implies that the learning rate decreases with every sample data which is presented to the Neural Network. Zero value is reached when the last sample is presented. While global scheduling conforms to the theoretical formulation, local scheduling of the learning rate is similar to perform as many independent and simultaneous local adaptation processes as units. Part of our works try to determine the benefits of local scheduling experimentally. Obviously the sequences of the learning rate parameters given by both local and global strategies do not comply with the conditions imposed by the convergence of the stochastic gradient approach. But it is the best approximation that works under a "one-pass" adaptation constraint.

Up to this point, the CNN are applied to stationary data, represented by a sample $\mathbf{X}$. In the case of non stationary data, we have a sequence of data samples $\left\{\mathbf{X}\left(t\right)=\left\{\mathbf{x}_1\left(t\right),\ldots,\mathbf{x}_N\left(t\right)\right\};t=0,1,\ldots\right\}$ over which the adaptation

Algorithm 5.1 Application of SCL and FSCL to the color image sequence

1. Assume an initial codebook $\mathbf{Y}(0)$, $t = 0$.
2. Update the clock $t = t + 1$ and take the next sample $\mathbf{X}(t)$ of size n.
3. Assume as the initial codebook the result of the adaptation at the previous time instant $\mathbf{Y}(t, 0) = \mathbf{Y}(t - 1, n)$
4. Compute the sequence of adaptations of the codebook $\{\mathbf{Y}(t, \tau); \tau = 1, \ldots, n\}$ applying either SCL or FSCL, to $\mathbf{x}(\tau)$ extracted from $\mathbf{X}(t)$.
5. Resume the process from step 2.

rules of SCL or FSCL will be applied. In the non stationary case we have two time parameters: that of the reality (t) and that of the internal adaptive computations (τ). At each real time instant a complete adaptation process will take place, the whole process is as Algorithm 5.1.

5.4 Experimental results

In this section we report some experiments performed on image sequences extracted from video sequences, aimed to evaluate the robustness of SCL and FSCL as adaptive VQ algorithms for Non-Stationary Color Quantization, and their sensitivity to diverse numerical parameters. The sequence of images used for the experiment is a panning of the laboratory taken with an Apple CCD color video-camera designed for video-conference. Original images have an spatial resolution of 240x320 pixels. We have created two sequences. In one of them each of two consecutive images overlap roughly 50% of the scene (*sequence1*), in the other the overlap is of 33% (*sequence2*). The experiments refer to the computation of sets of color representatives (color palettes) of size $c = 16$ and $c = 256$. These sizes of color palettes are representative of the ones that can arise in segmentation and compression tasks, respectively.

The benchmark algorithm used is the minimum variance Heckbert [141] algorithm, as implemented in MATLAB [283]. This algorithm has been applied to the entire images in the sequences in two ways. Figure 5.1 shows the distortion results of the Color Quantization of the experimental sequences to 16 and 256 colors based on both applications of the Heckbert algorithm. The results consist of the distortion per image curves, and the total distortion over the entire sequence shown in the figure legend. The curves named *Time Varying* are produced applying the algorithm to each image independently. This corresponds to the optimal strategy for Non-Stationary Clustering, assuming the optimality of the Heckbert algorithm. The curves called *Time Invariant* come from the assumption of stationarity of the data: the color representatives obtained for the first images are used for the Color Quantization of the remaining images

in the sequences. Obviously, the distortion is greater in the *Time Invariant* application. The difference between both curves increases as the time evolution of the color distribution departs from the initial one found in image #1 of the sequence. The gap between those curves gives an indication of the non-stationarity of the data. From our point of view, this gap defines the response space left for truly adaptive algorithms. To accept an algorithm as an adaptive solution its response could not be worse than the *Time Invariant* curve. The *Time Varying* curve defines the best response that we expect, although it is not the sequence of global optima. In Figure 5.1, also it can be appreciated that *sequence2* curves change more smoothly than *sequence1* curves. The changes in color distribution are smoother in *sequence2*, therefore, it can be expected that the results of adaptive algorithms will be better for *sequence2* than for *sequence1*. Also it can be appreciated that the *Time Invariant* curve seems to approach the *Time Varying* curve at the end of the sequence in all cases. This behavior is due to the nature of the image sequences, they are extracted from a closed panning of the scene, so that the final images almost coincide with the initial ones, and their color distributions come close. This feature is by no means general, it is obviously an artifact of our experimental data. From a qualitative point of view the closeness serves also to test the ability of the adaptive algorithms to come back to the original optimum.

In the following, the results of the application of the CNN will be given in the relative framework of the *Time Invariant* and *Time Varying* Heckbert results. The relative distortion ($E_R(t)$) shown in the figures is computed as $E_R(t) = \frac{E_{CNN}(t) - E_{TV}(t)}{E_{TI}(t) - E_{TV}(t)}$, where is the per image distortion of the color quantization with the color palette computed by the CNN (either SCL or FSCL). The relative distortion is negative when the CNN improves over the optimal Heckbert *Time Varying*, and it is greater than 1 when the CNN does not behave adaptively (gives results worse than the *Time Invariant*). We have appended to the legend of the curves the accumulated value of the relative distortion along the sequence.

From the description in it is evident that the computational cost of the CNN applied to find the optimal Color Quantization of each image is of the order $O(d.n.c)$, with n the size of the sample, $d = 3$ the dimension of the space, and c the number of colors. To evaluate the performance of the algorithm as real time restrictions are imposed, we have employed image samples of size $n = 1,200$ (*sample1* $< 2\%$ of image size) and $n = 19,200$ (*sample2* $= 25\%$ of image size), respectively. Whereas the distortion results in Figure 5.1 are produced by the Color Quantization to a set of color representatives computed using all the image pixels, both SCL and FSCL will be applied to a subset of the image pixels, extracted randomly. The sequence of image samples *sample1* represents a stronger real time constraint than *sample2*, and the application of both SCL and FSCL will be one order of magnitude faster for *sample1*. The distortion results shown in the figures and tables are the distortion of the entire images when color quantized with the color representatives computed by the CNN upon the specified image samples. Besides the real time considerations, the use of image samples gives also some hints about the robustness and extrapolation

abilities of both SCL and FSCL.

5.4.1 Sensitivity to codebook and sample size

The first set of experiments are performed assuming as the initial codebook the color palette obtained by the Heckbert algorithm for the first image. This is the best initial condition that we can think of to start the adaptation of the remaining sequence, (note that all the curves start at zero). The results of this set of experiments are shown in Figures 5.2 and 5.3. The experiment includes the computation by SCL and FSCL of the color representatives under all combinations of image sequence, sample size, color palette size and local versus global scheduling of the learning rate. In all the cases, the codification into 256 colors gives worse relative results. This result is quite important because it indicates the sensitivity of the CNN performance to the size of the color palette. This result, that we have found very general, implies that the proposition of adaptive algorithms, CNN-like or other, that would search for the "natural" number of clusters must be taken with great care, moreover if the distortion is a salient component of the clustering criterion function. The inspection of the figures shows that the algorithms perform adaptively in almost all the cases: the relative distortion is less than 1 most of the times. The exception occurs usually at image #2 of the sequence. This can be explained by the narrow gap between the reference curves at this point of the image sequence, however the algorithms quickly recover.

Each of the plots show the result using both sample sizes. In general it can be appreciated that the use of the bigger sample improves the results, although the magnitude of this improvement is related to the codebook size. The above mentioned sensitivity to the number of clusters searched can be appreciated if we compare one-to-one the set of Figures 5.2(a), 5.2(c), 5.2(e), 5.2(g), 5.3(a), 5.3(c,) 5.3(e), 5.3(g) with the set 5.2(b), 5.2(d), 5.2(f), 5.2(h), 5.3(b), 5.3(d), 5.3(f), 5.3(h). This sensitivity is attributable to the ratio between the size of the sample and the number of representatives searched: the sample-to-codebook ratio. We have found that is a good ratio in many cases, *sample1* fits this ratio for 16 colors and *sample2* for 256 colors. The significance of this ratio is confirmed by the following observations over the Figures 5.2 and 5.3:

- When searching for 16 colors the use of *sample2* does not give an improvement over the results obtained from *sample1*, according to the increase in computational complexity.
- When searching for 256 colors, the use of *sample2* improves significatively the results over those given obtained from *sample1*.

This ratio is of interest to bound the real time applicability of our algorithms, or the suboptimal results that can be expected from the use of small samples imposed by real time constraints. In Tables 5.1 and 5.2 we have gathered the global distortion results that summarize all the experiments, including the sensitivity to initial conditions which will be discussed later. From those tables it can

be seen that the impact of the sample-to-codebook ratio depends also of other elements of the algorithms, such as the initial conditions and the scheduling of the learning rate.

5.4.2 The effect of learning rate scheduling

The effect of the scheduling of the learning rate can be appreciated by the comparison of the plots in Figure 5.2 with those in Figure 5.3. Also, in each Table 5.1 and 5.2, we have a separate subtable for each scheduling strategy. Our conclusion is that the global scheduling of the learning rate gives better results than the local scheduling when the amount of available information increases. However as the information becomes scarce, the local scheduling is more robust and gives better results. To support this conclusion observe in Tables 5.1 and 5.2 that the results of the global scheduling subtable improve over the corresponding ones of the local scheduling subtable when *sample2* is used to search for 16 colors, and when applying the algorithm to *sequence2* (which is smoother than *sequence1*) starting from relatively good conditions.

5.4.3 The effect of time subsampling

As we have said, *sequence1* was a more coarse time sampling of the original video sequence. This coarseness would lead to more abrupt changes in distribution, that would make *sequence1* less apt for adaptive computation. In Table 5.3, the mean relative distortion per image has been computed, by averaging the entries in Tables 5.1 and 5.2, and taking into account the number of images in each sequence. This table allows to evaluate the impact of the supposed augmented smoothness of *sequence2*. It can be seen from it that other factors have more impact than the increased smoothness produced by a fine time sampling.

5.4.4 Robustness to initial conditions

If we, at this point, try to compare SCL results with FSCL results, the main conclusions are:

- SCL performs better than FSCL starting from good initial conditions, and high sample-to-codebook ratio is available.
- FSCL is more robust than SCL, as intended, in the sense that it improves SCL when the algorithm starts from bad initial conditions and the sample data is scarce.

These conclusions are consistent with the proposition of SCL as a local optimization procedures, and of FSCL as a global optimization procedure. The global properties of FSCL ensure a good average result, but starting from good initial conditions the pure local algorithm performs better. These conclusions are made stronger when considering the robustness to initial conditions, evaluated in the second experiment. In Tables 5.1 and 5.2, we present the sum

of relative distortion results of the Color Quantization of entire sequences obtained applying the SCL and FSCL (with both global and local learning rates) starting from various initial color representatives: the Heckbert color representatives for image #1 (*Heckbert*), a threshold based selection of the sample of image #1 (*Threshold*), random points in the RGB cube (*RGBbox*) and a random selection of samples of image #1 (*Sample*). The results given are the sum of relative distortion through the sequence excluding image #1. The results are not averaged of normalized anyhow regarding the sequence duration, therefore entries for *sequence2* are bigger than those for *sequence1*. Tables 5.1 and 5.2 show that there is a remarkable increment of distortion results due to initial conditions. The worst case is the application of the SCL algorithms to search 256 representatives starting from the RGB box initialization. Also the tables confirm that FSCL is more robust than SCL to very bad initial conditions but their performance is comparable for good an reasonably good initial conditions. SCL may even improve FSCL in very good initial conditions and numerical circumstances: big sample and global scheduling of the learning rate.

In Figure 5.4 we show some of the responses to diverse initial conditions in detail. The sequence considered is *sequence1*, the number of color representatives is 16. The worst response is in Figure 5.4(a) for SCL using *sample1* and a local scheduling of the learning rate. The algorithm tries to approach the response obtained from the *Heckbert* initial condition, starting from the other initial conditions. Starting from a good initial condition (*Sample*) the SCL gives the same response after 5 images. Starting from a medium quality initial condition (*Threshold*) the SCL recovers from the bad initialization after some 14 images. Finally the worst initial condition (*RGBbox*) can not be recovered in the duration of the sequence. On the other hand, the best case is that of Figure 5.4(d), with *sample2* and the global learning rate that recovers from the bad initial conditions very fast, its response collapses after 2 or 3 images in the one corresponding to the best initial condition. FSCL will be more robust than SCL in the sense that the effect of bad initial codebooks is recovered faster by FSCL than by SCL, under the same numerical settings. This can be appreciated comparing Figures 5.4(a) and 5.4(b) with Figures 5.4(c) and 5.4(d).

5.4.5 Visual results

To give a visual qualitative appreciation of the Color Quantization results, Figures 5.5 to 5.10 show the results of color quantization to 16 color representatives on images #2, #8, #12, and #20 of *sequence2*, under the application of some instances of the algorithms discussed in the paper. These images were selected because they show the sharper transition of color distribution. For each image, we show the color quantized image (left), the color representatives found by the algorithm (color bars in the middle), and color quantization error (right) as a color image obtained from the error in each color axis. Figure 5.5 and 5.6 give the visual results of the *Time Varying* and the *Time Invariant* application, respectively, of the Heckbert algorithm over the entire image. The suboptimality of the latter is appreciable from the inspection of the quantized images (left).

The inspection of the error images (right) shows the increase in magnitude of the error of the *Time Invariant* relative to the *Time Varying* strategy, as time goes on.

Figures 5.7 and 5.8 show, respectively, the results of the application of SCL and FSCL using the optimal initial conditions (*Heckbert*) and the small sample of each image. It can be appreciated that the visual differences between the quantized images (left) obtained from SCL and FSCL are almost negligible. However, looking at the color representatives shown in the middle color bars, it can be appreciated that the yellow color representative is not changed by SCL (Figure 5.7, middle) although it is not used in the codification. This yellow color does correspond to a stuck codevector, that it is not changed by SCL along the whole sequence. The color representatives of FSCL (Figure 5.8, middle) follow more closely the ones found by the Heckbert *Time Varying* application.

To illustrate the relative increase of robustness to initial conditions of FSCL over SCL, the Figures 5.9 and 5.10 show, respectively, the results of their application using the worst initial condition (*RGBbox*) and the small sample of each image. The existence of stuck codevectors is conspicuous in Figure 5.9, while the color representatives computed by FSCL in Figure 5.10 behave very similarly to the ones shown in Figure 5.8, and again follow better the optimal color representatives computed by the *Time Varying* Heckbert shown in Figure 5.5. The error images in Figure 5.10 produced by FSCL are more smooth and of lesser magnitude than those produced by SCL in Figure 5.9. The visual difference of the quantized images obtained from SCL and FSCL is not so noticeable.

Regarding the visual evaluation of the results, we can conclude that the adaptive neural algorithms improve significatively over the stationary solution given by the *Time Invariant* Heckbert algorithm. Despite the existence of stuck codevectors, SCL give visual results with a quality similar to the visual results of FSCL.

5.5 Conclusions

Non-Stationary Color Quantization is the problem of finding optimal color representatives in image sequences. It has been put into the framework of Non-Stationary Clustering problems, and two basic Competitive Neural Network architectures have been proposed as one-pass Adaptive Vector Quantization methods to solve the problem. The solution provided by the CNN is able for real time implementation, given its computational cost, versus the cost of the Heckbert algorithm. Besides real time considerations, these cost figures imply the CNN methods can be applied to higher dimensional instances of Non-Stationary Clustering problems (i.e. hyperspectral images) . In this paper we have concentrate our attention on two basic CNN architectures: SCL and FSCL, trying to asses their performance and numerical sensitivities. This work will be of interest in understanding and anticipating the response of other CNN architectures on this problem. We have found that both SCL and FSCL show some sensitivity to the number of color representatives searched (the size of the codebook). This

		sequence1				*sequence2*			
		$c = 16$		$c = 256$		$c = 16$		$c = 256$	
		S1	S2	S1	S2	S1	S2	S1	S2
SCL	*Heckbert*	2.5	2.9	6.6	2.9	4.7	2.8	9.5	4.7
	Threshold	9.1	3.4	15.0	8.1	11.0	3.2	23.0	13.0
	Sample	3.6	3.2	15.0	7.2	3.6	3.2	23.0	10.0
	RGBbox	12.0	6.4	32.0	16.0	15.0	8.5	52.0	25.0
FSCL	*Heckbert*	3.4	4.7	6.2	2.5	3.7	5.1	9.0	4.8
	Threshold	6.7	4.6	15.0	7.2	8.0	5.7	23.0	12.0
	Sample	3.3	4.6	15.0	5.7	4.7	5.1	23.0	9.6
	RGBbox	3.2	4.8	21.0	5.4	3.1	5.2	33.0	8.6

Table 5.1: Accumulated relative distortion results of the Color Quantization of experimental sequences with the color representatives computed adaptively by the SCL and FSCL with local scheduling of the learning rates, for various initial conditions, sample sizes (S1: *sample1*, S2: *sample2*) and number of color representatives.

is expected to be a general sensitivity, and has been also found in other CNN architectures. We have studied the response to changes in sample size, and the implicit generalization ability of SCL and FSCL. A convenient ratio between sample size and codebook size is $100 : 1$. We have studied the effect of local versus global scheduling of the learning rate, finding that the later is more efficient when there is a surplus of training data, whereas the former provides an additional robustness when the data is scarce. Finally, we have tested the response of both SCL and FSCL to suboptimal initial conditions, finding that the FSCL strategy of looking for uniform quantizers provides greater robustness to bad initial conditions. Our conclusion is that one-pass application of CNN can be a successful family of Adaptive Vector Quantization, qualified for real-time and high dimensional applications.

Figuras y tablas

		sequence1				*sequence2*			
		$c = 16$		$c = 256$		$c = 16$		$c = 256$	
		S1	S2	S1	S2	S1	S2	S1	S2
SCL	*Heckbert*	2.6	-0.43	7.5	3.1	3.8	-1	11.0	4.6
	Threshold	9.8	0.44	18.0	8.5	12.0	0.25	28.0	13.0
	Sample	4.1	0.05	18.0	7.4	4.7	-0.39	28.0	11.0
	RGBbox	13.0	3.7	36.0	17.0	16.0	7.0	57.0	29.0
FSCL	*Heckbert*	3.4	1.1	7.3	2.6	3.0	2.1	10.0	4.0
	Threshold	9.0	1.1	19.0	7.3	9.0	2.6	29.0	13.0
	Sample	3.5	0.95	19.0	6.0	4.4	2.4	29.0	9.6
	RGBbox	4.7	1.4	28.0	6.8	4.5	2.3	45.0	11.0

Table 5.2: Accumulated relative distortion results of the Color Quantization of experimental sequences with the color representatives computed adaptively by the SCL and FSCL with global scheduling of the learning rates, for various initial conditions, sample sizes (S1: *sample1*, S2: *sample2*) and number of color representatives.

	sequence1 (18 images)				*sequence2* (27 images)			
	$c = 16$		$c = 256$		$c = 16$		$c = 256$	
	S1	S2	S1	S2	S1	S2	S1	S2
SCL	0.14	0.07	0.39	0.17	0.16	0.03	0.37	0.17
FSCL	0.19	0.16	0.37	0.14	0.12	0.13	0.35	0.16

Table 5.3: Mean per image relative distortion, computed averaging the entries in Tables 5.1 and 5.2 and taking into account the different number of images in each sequence.

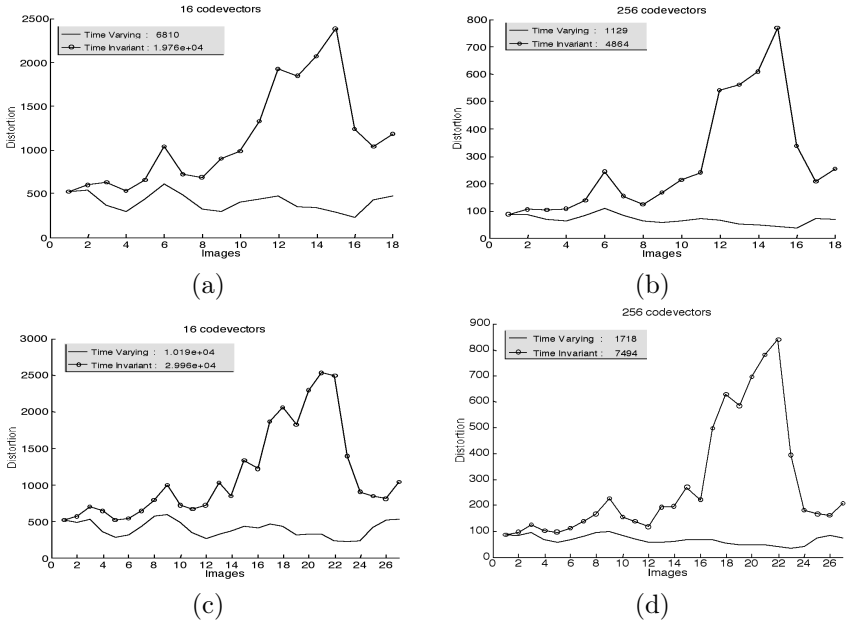


Figure 5.1: Reference distortion values obtained with the application of the *Time Varying* and *Time Invariant* Heckbert algorithm for (a) 16 colors, (b) 256 colors for *sequence1*. And for *sequence2* (c) 16 colors, (d) 256 colors. The amounts in the legend display the total distortion along the entire sequences.

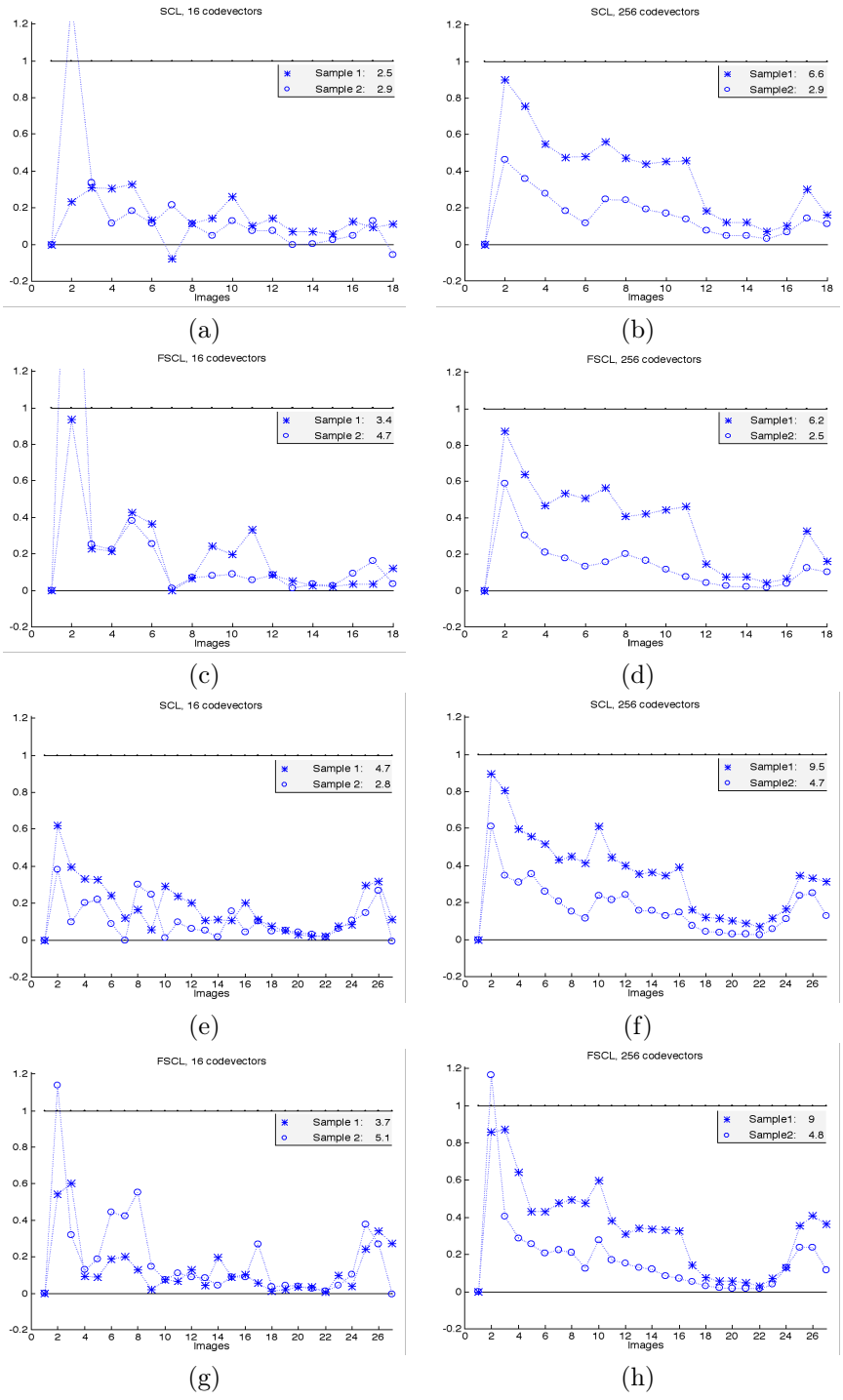


Figure 5.2: Relative distortion results for *sequence1* with local learning rates applying SCL to (a) $c = 16$ and (b) $c = 256$, and applying FSCL to (c) $c = 16$ and (d) $c = 256$, (e) $c = 16$ and (f) $c = 256$, and applying FSCL to (g) $c = 16$ and (h) $c = 256$.

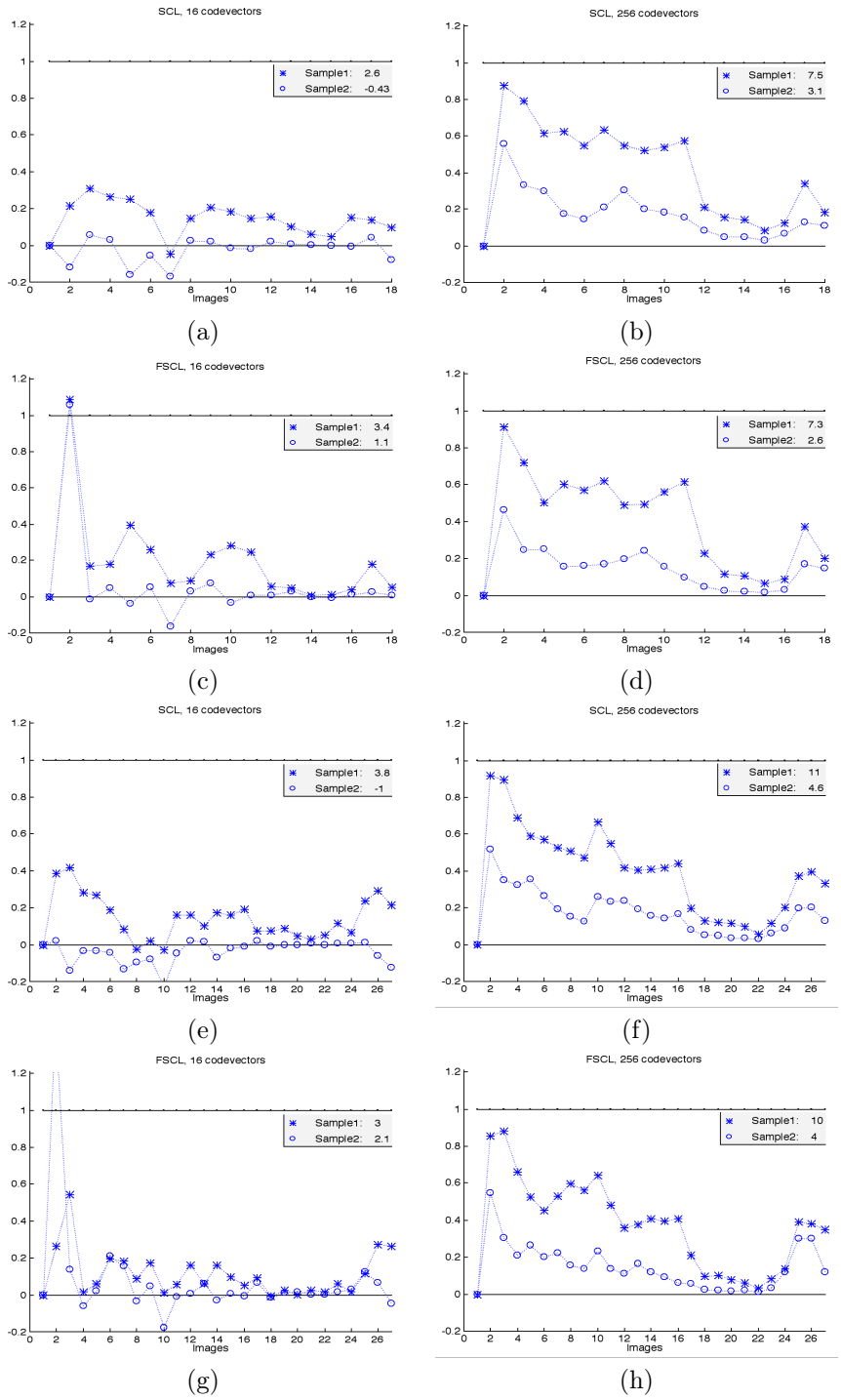


Figure 5.3: Relative distortion results for *sequence1* with global learning rate applying SCL to (a) $c = 16$ and (b) $c = 256$, and applying FSCL to (c) $c = 16$ and (d) $c = 256$. Relative distortion results for *sequence2* with global learning rate applying SCL to (e) $c = 16$ and (f) $c = 256$, and applying FSCL to (g) $c = 16$ and (h) $c = 256$.

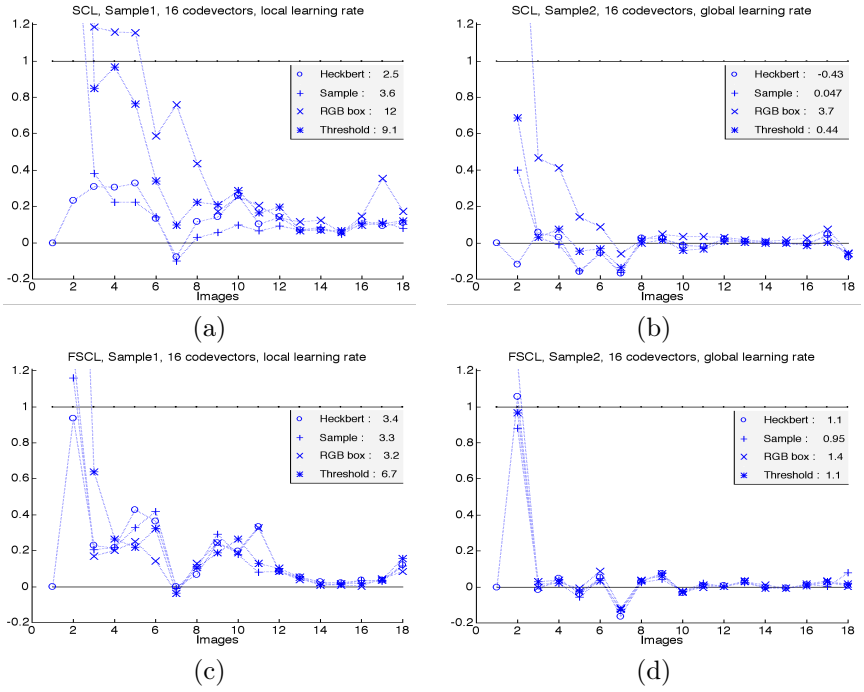


Figure 5.4: Robustness: relative distortion results of the Color Quantization of experimental *sequence1* with the 16 color representatives computed adaptively by the SCL and FSCL with optimal learning rates for both sample sizes, and various initial conditions.

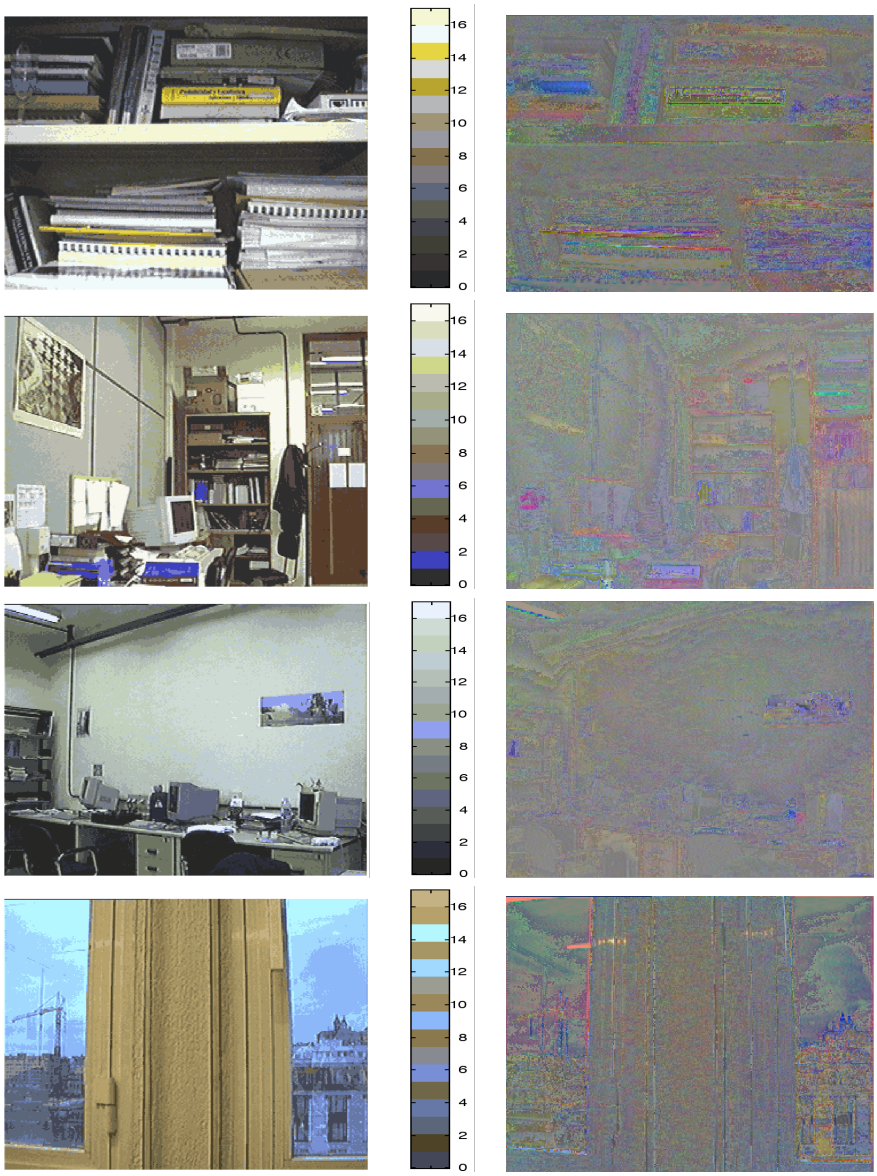


Figure 5.5: Results of the Color Quantization of image #2, #8, #12, and #20 of the experimental *sequence2* with the 16 color representatives (middle images) computed by *Heckbert* using full size images (*Time Varying*). On the left the quantized images, and on the right the error images.

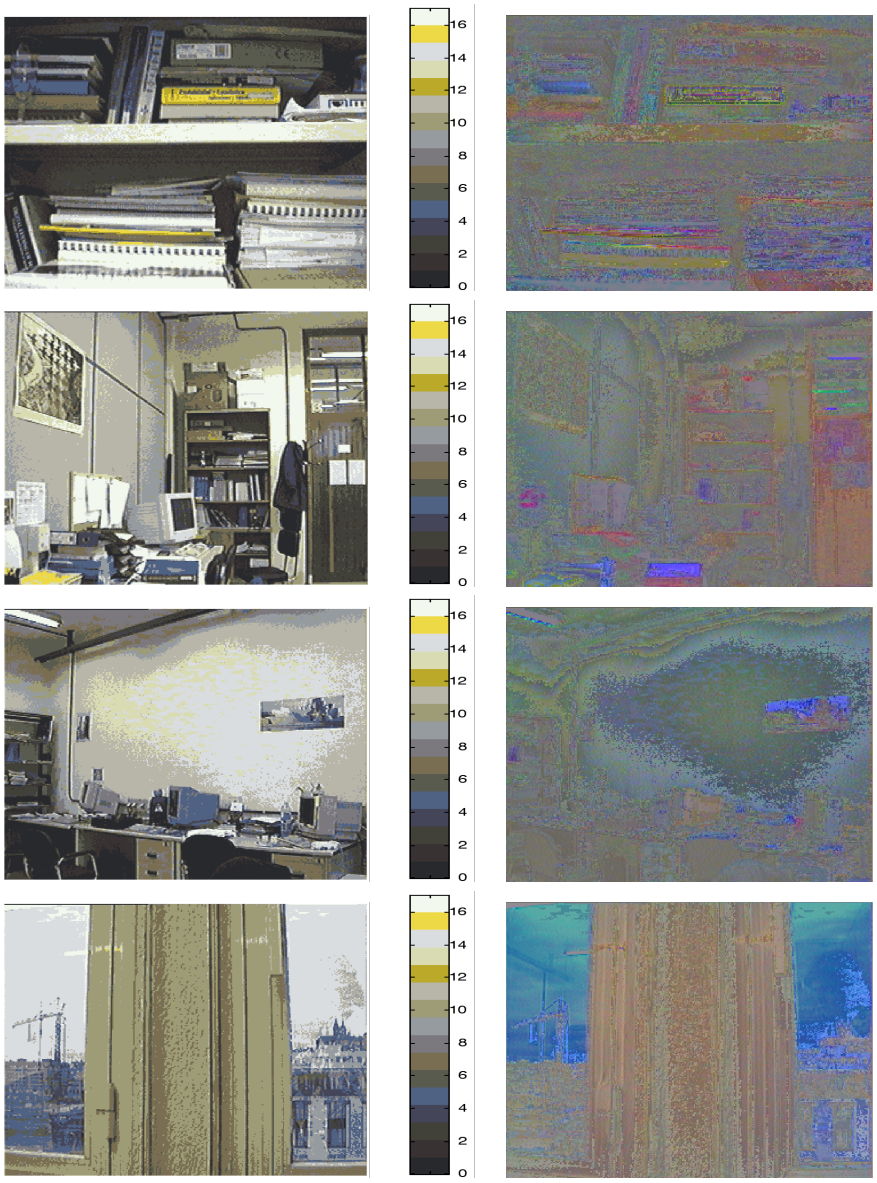


Figure 5.6: Results of the Color Quantization of image #2, #8, #12, and #20 of the experimental *sequence2* with the 16 color representatives (middle images) computed by *Heckbert* using #1 image (*Time Invariant*). On the left the quantized images, and on the right the error images.

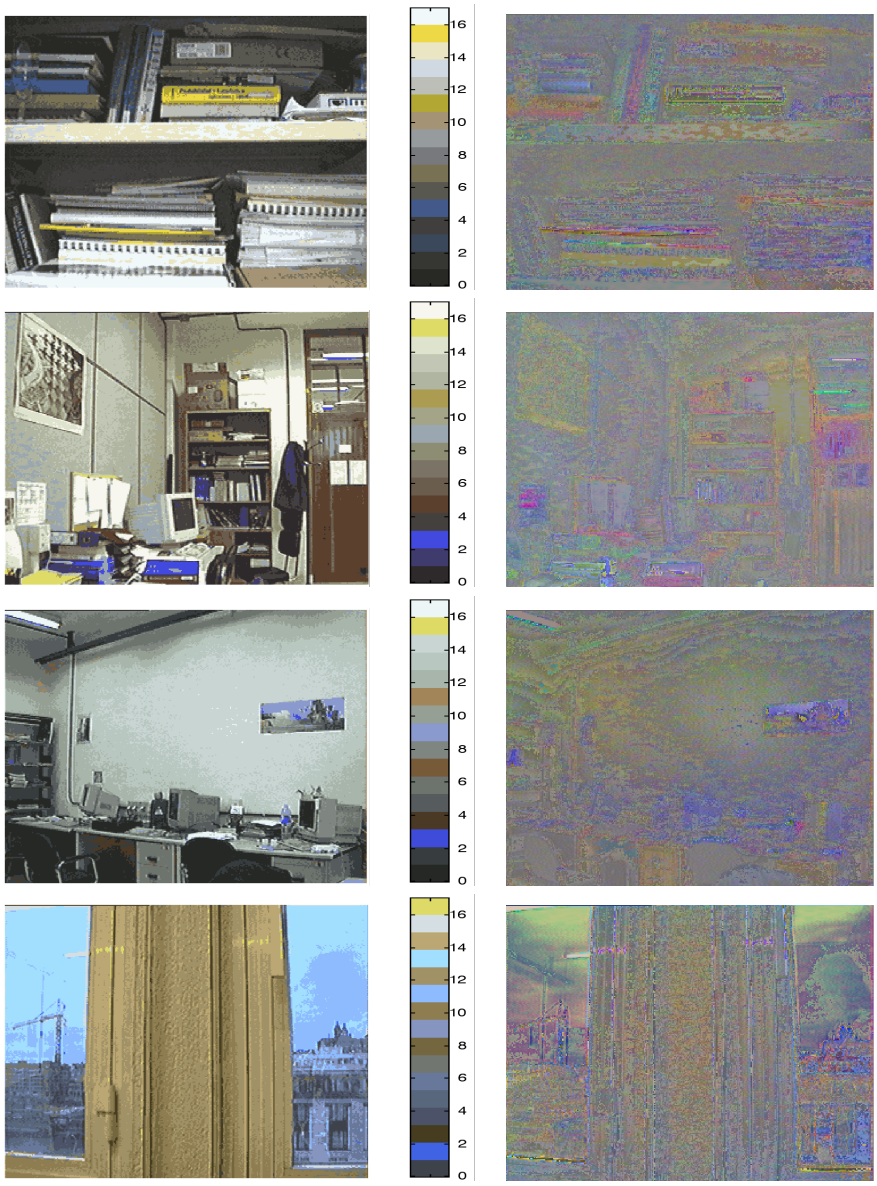


Figure 5.7: Results of the Color Quantization of image #2, #8, #12, and #20 of the experimental *sequence2* with the 16 color representatives (middle images) computed adaptively by the SCL with local learning rates, using *sample1*, and *Heckbert* initial condition. On the left the quantized images, and on the right the error images.

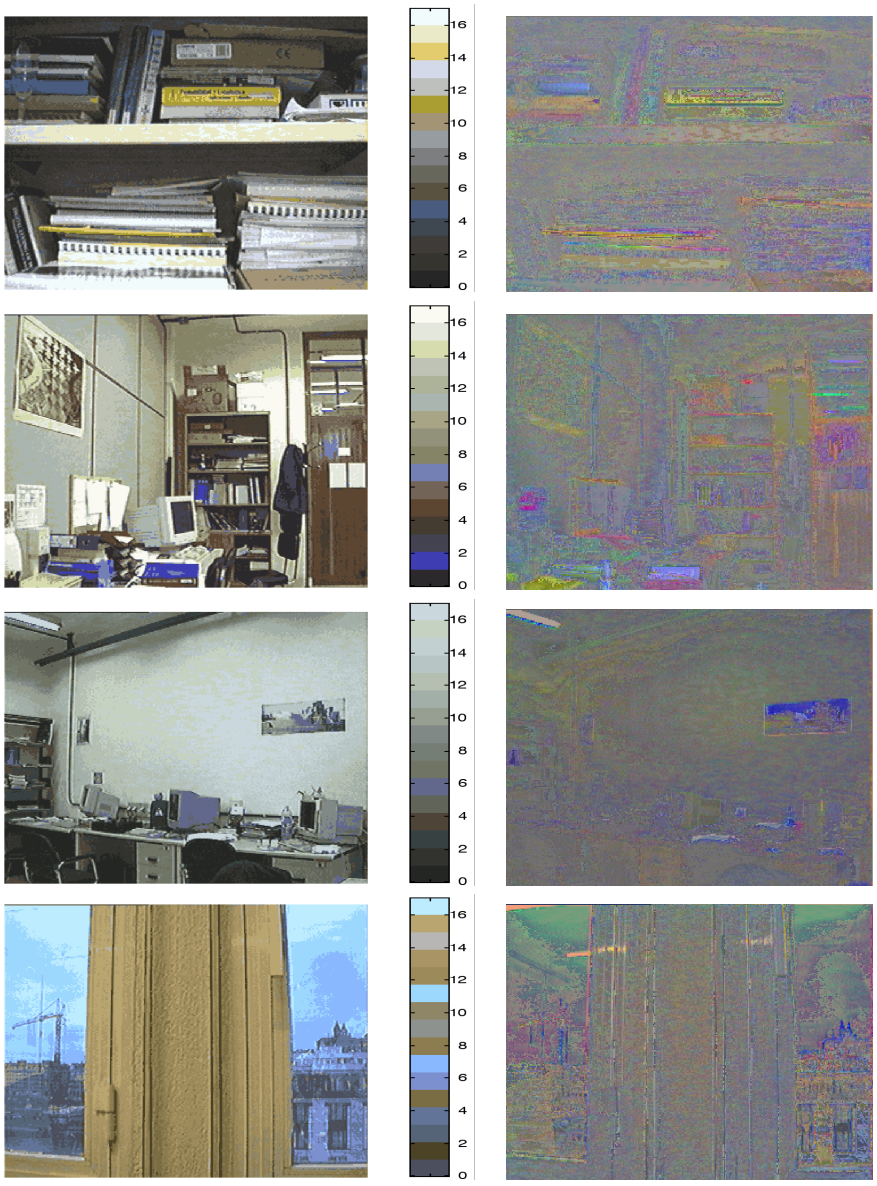


Figure 5.8: Results of the Color Quantization of image #2, #8, #12, and #20 of the experimental *sequence2* with the 16 color representatives (middle images) computed adaptively by the FSCL with local learning rates, using *sample1*, and *Heckbert* initial condition. On the left the quantized images, and on the right the error images.

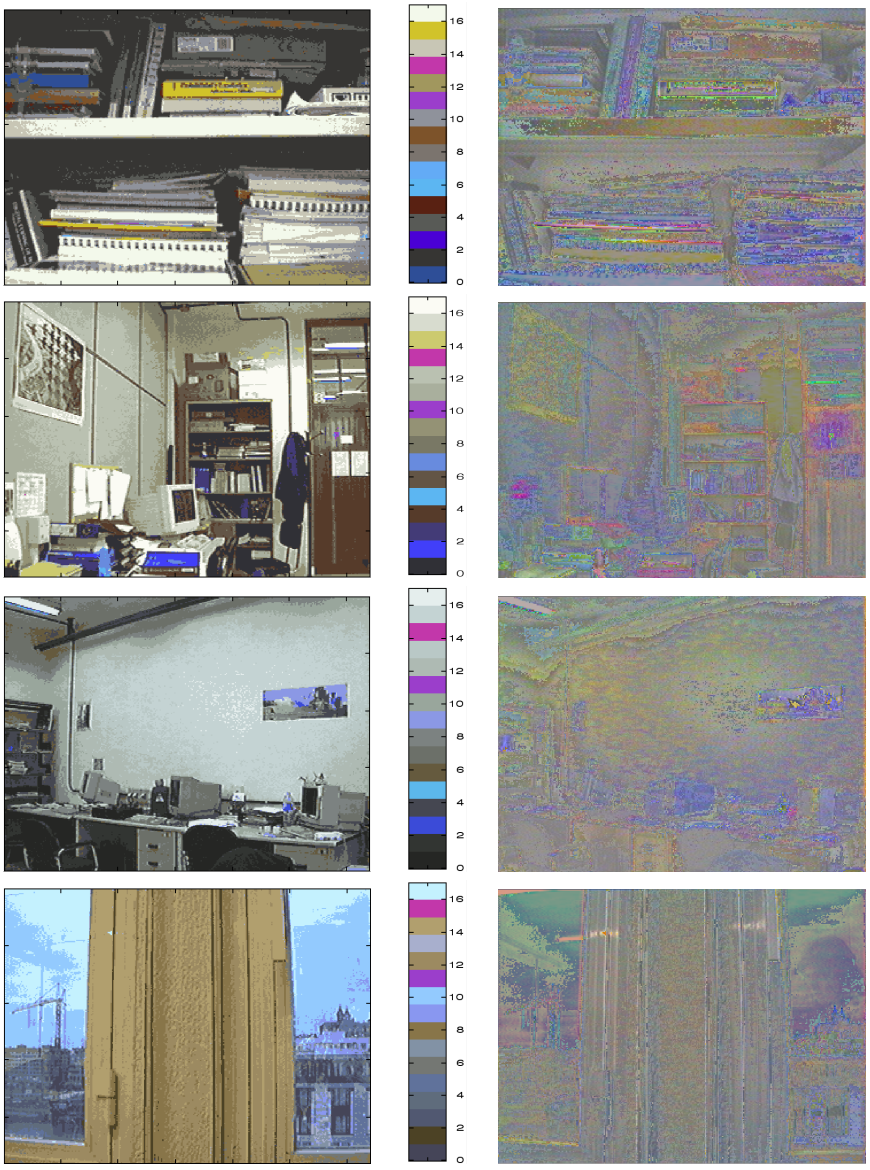


Figure 5.9: Results of the Color Quantization of image #2, #8, #12, and #20 of the experimental *sequence2* with the 16 color representatives (middle images) computed adaptively by the SCL with local learning rates, using *sample1*, and *RGBbox* initial condition. On the left the quantized images, and on the right the error images.

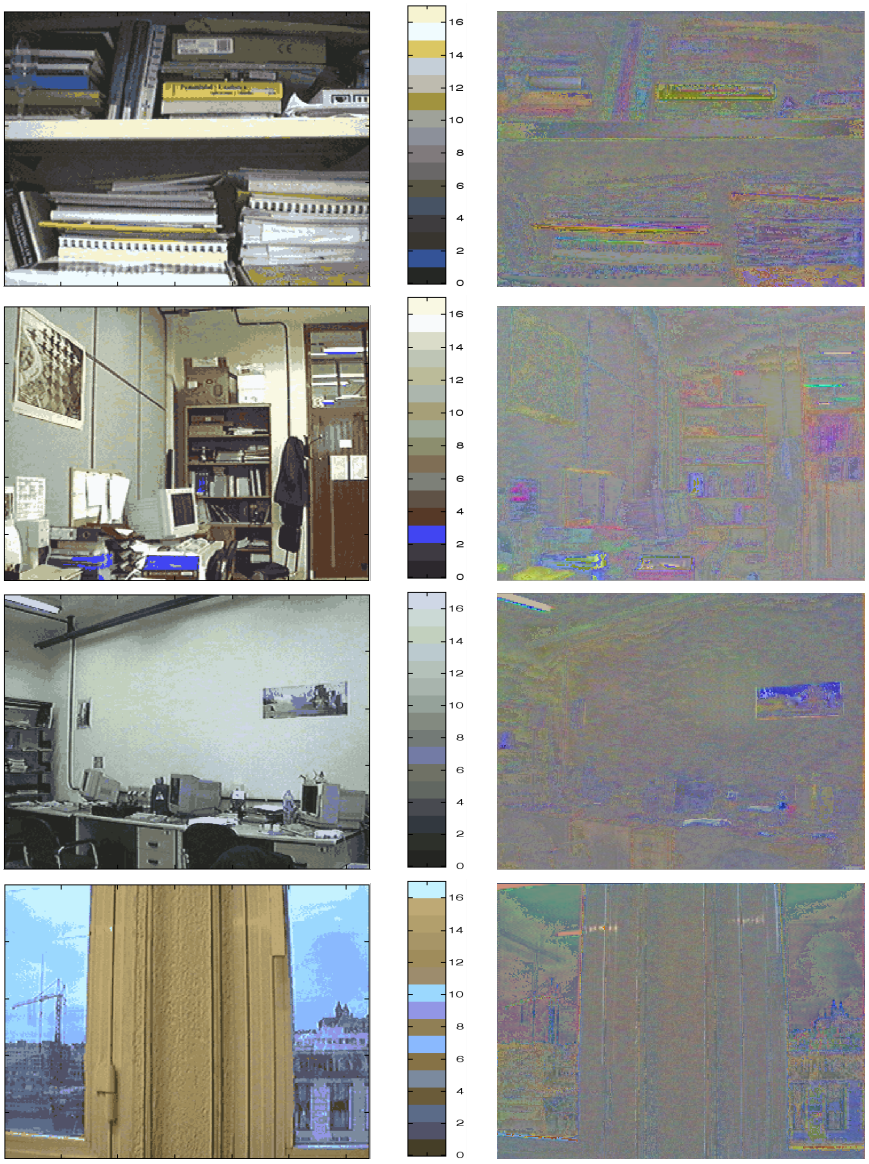


Figure 5.10: Results of the Color Quantization of image #2, #8, #12, and #20 of the experimental *sequence2* with the 16 color representatives (middle images) computed adaptively by the FSCL with local learning rates, using *sample1*, and *RGBbox* initial condition. On the left the quantized images, and on the right the error images.

Chapter 6

Experiments Color Quantization of Image Sequences using Neighborhood based Competitive Neural Networks

In this chapter we discuss the application of several competitive Neural Networks to Frame-Based Adaptive Vector Quantization (FBAVQ) methods. The task of Non-Stationary Color Quantization is a manageable instance of FBAVQ, for which there is an almost optimal benchmark algorithm that serves to validate the neural network results. The Simple Competitive Learning (SCL) is the basic adaptive algorithm for Vector Quantization (VQ). The Self Organizing Map (SOM), Fuzzy Learning Vector Quantization (FLVQ) and Soft Competition Scheme (SCS) are presented as adaptation rules that converge to the SCL.

6.1 Introduction

The approach taken in this work is to assume that the end application is the design of a Vector Quantizer that minimizes the Euclidean distortion. Therefore, the straightforward neural approach is the application of the SCL. We view other competitive neural networks as modifications of the SCL aimed to improve its robustness against bad initial conditions [43] and its overall performance. Here we will consider the Self Organizing Map (SOM) [168], an on-line version of the Fuzzy Learning Vector Quantization (FLVQ) [30] and the Soft-Competition Scheme (SCS)[30, 288]. We have focused in the SOM, FLVQ and SCS because they can be seen to become the SCL rule for limit values of the

neighboring function control parameters. Therefore, their objective functions can be assumed to converge to the Euclidean distortion. An efficient scheduling of these control parameters can lead to a fast, robust and efficient minimization of the Euclidean distortion. The main computational constraint imposed in the experiments described here is the one-pass adaptation. We have imposed the same scheduling of the learning rate parameter in all the algorithms, trying to isolate the influence of the neighboring functions. The neighborhood reduction was the same exponentially decreasing function for the SOM, FLVQ and SCS. We try to isolate the effect of the shape of the neighboring function from the scheduling of its decrease. Best results in all cases were found when the convergence to the SCL algorithm was fast.

The consideration of time varying data in its more general case leads to Adaptive Vector Quantization (AVQ). Which becomes Frame Based AVQ when dealing with image sequences. We formulate FBAVQ problems as the dynamic search of the optimal representatives based on the sample data at each time instant. Neural Networks have been of little practical application in AVQ, because of their slow convergence [97]. The most common AVQ approaches are based on codebook replenishment algorithms [86], which pose severe tuning problems. One-pass neural networks described here can be easily tuned, posses strong robustness and are relatively fast. Their computational complexity grows linearly with the dimension of the space and the sizes of the codebook and the sample. Adaptive Color Quantization (ACQ) of image sequences is an instance of FBAVQ. The task is to find the optimal set of color representatives in the RGB color space for each image in the sequence. The non-stationary character of the data comes from the unpredictable color distribution of the pixels in the images. We report the results of a series of computational experiments that show the performance of the SOM, SCS and FLVQ on this task. These experiments continue those reported in [109, 110].

6.2 Experimental results on Adaptive Color Quantization

Color Quantization can be stated as a Clustering problem in a 3D space: the unit cube of RGB color representations. Color Quantization has applications in visualization and compression of color images [71, 141, 152, 277, 285], image segmentation based in color features [187, 207, 272] and image retrieval from image databases [156]. When image sequences are considered [90], the Adaptive Color Quantization is an instance of Frame-Based AVQ in the RGB color space. Over the experimental non-stationary data we have applied the minimum variance Heckbert algorithm that gives the optimal benchmark to validate the results of our competitive neural network algorithms. After that, we apply the SCL to show the basic neural adaptation. Then the SOM, FLVQ and SCS are applied as robust modifications of SCL. We explore their robustness against the degradation of the global initial conditions.

6.2.1 The Non-stationary experimental data

The sequence of images used for the experiment is described in Appendix The reference Heckbert algorihtm and the results that set the benchmark for the experiments in this chapter are described in Chapter

The experiments perform the quantization to 16 and 256 colors, we assume that 16 colors is representative of segmentation tasks, and 256 of compression/visualization tasks. For the Heckbert algorithm the color quantizers were computed using the entire images. The remaining algorithms (SCL, SOM, FLVQ, SCS) were applied to pixel samples of the images to compute de color representatives. These color representatives are then used to color quantize the entire images, and their distortion results are compared with the results of the Heckbert algorithm. From previous chapter results , we have detected a certain sensitivity of the algorithms to the sample size, so we have selected a priori adequate sample sizes for the tasks intended: 1600 pixels for $c = 16$ and 25600 for $c = 256$.

As results we give the distortion results along the image sequence shown in the figures. All these distortions refer to the quantization of the full size images. In the headings of the graphs the global distortion (the sum of the individual image distortions) is given for a more global comparison. We have gathered the most significant global distortion results for the full size images in Table 3. The magnitudes of the distortions are, obviously, greater for the quantization to 16 colors than for the quantization to 256 colors.

6.2.2 The reference algorithm: Minimum Variance Heckbert

As reference non adaptive algorithm we have used a minimum variance version of the algorithm proposed by Heckbert [141] with the enhancements proposed in [283] to compute the local variances for each partition (we have used the MATLAB implementation). This algorithm performs the successive partition of the unit cube based on the minimization of the inner variance of the resulting partitions. It implies the computation of the residual variances produced by each cutting plane, and its complexity is therefore of the order of the RGB cube discretization. It is almost optimal, but its computational cost is very high and can not be applied to higher dimensional problems, because its complexity grows exponentially with the dimension of the data space. On the other hand, our neural network algorithms have a complexity that grows linearly with the number of color representatives, the size of the sample and the space dimension. That means that our approach can be extended to higher dimensional problems, while the Heckbert algorithm could not. Figure 2 shows the results of the application of the Heckbert algorithm to the non-stationary experimental data searching for the optimal quantizer to 16 and 256 colors. This algorithm has been applied to the entire images in the sequence in two ways. Assuming the non-stationary nature of the data it has been applied to each image independently, this gives the optimal results shown in Figure 2a,b. The curve is

denoted *Time Varying* in the figures. The assumption of the stationarity of the data supposes that all the sequence is quantized with the color representatives obtained for the first image, which gives the worst results in Figures 2a,b. The curve is denoted *Time Invariant* in the figures. The Time Varying and Time Invariant applications of the Heckbert algorithm are useful to define the adaptive behavior in our experiments. They give upper and lower bounds for the remaining algorithms. An algorithm will not be considered to perform adaptively if its distortion is greater in any point than the Time Invariant Heckbert. On the other side, the Time Varying Heckbert is the best response that we expect from any adaptive algorithm.

6.2.3 The Simple Competitive Learning

In Figure 6.1 we show the results of the application of the one-pass adaptation of SCL, with the scheduling of the learning rate discussed above, embedded in the results of the Heckbert algorithm. In Figures 6.1(a) and 6.1(b), the initial codebook for the sequence was the Heckbert codebook of the first image, and the adaptation starts in the second image. It can be appreciated that the SCL performs adaptively in the sense discussed in previous section: it improves upon the Time Invariant application of the Heckbert algorithm, but is less optimal than the Time Varying. To stand out the relative distance to the optimal we present in Figures 6.1(c) and 6.1(d) the distortion of the SCL application relative to the Time Varying and Time Invariant Heckbert results. These curves are computed as:

$$SCL_{relative}\,(\#i) = \frac{SCL\,(\#i) - \ \mathsf{Time\ Varying}\,(\#i)}{\mathsf{Time\ Invariant}\,(\#i) - \mathsf{Time\ Varying}\,(\#i)};\ i = 2, 3, ... \quad (6.1)$$

Therefore $SCL_{relative}\,(\#i)$ is negative when the SCL response improves the optimal Time Varying. It is greater than 1 when it gives a non adaptive response, worse than Time Invariant. It is clear from the comparison of 6.1(c) and 6.1(d) the degradation induced by the increase in the codebook size.

The initial condition for the sequence, the Heckbert codebook for the first image, is rather optimal. In Figures 6.1(e) and 6.1(f) we test the response to other initial conditions. These are assumed to be the codebooks for the first image in the sequence, and are used to start the adaptation process. These initial codebooks are:

- in Sample: is a good initial codebook extracted from the sample of the first image. They do not coincide in the 16 and 256 color case.

- in RGBbox: is an arbitrary codebook randomly generated in the RGB cube, it is the worst initial-codebook case.

- Threshold (umbral): corresponds to a threshold guided selection of elements in the sample of the first image.

	SOM		FLVQ				SCS		
	$v_0 =$		$m_0 =$				$\sigma_0 =$		
	1	8	10	7	4	2	0.1	2	$\widehat{\sigma_{i,0}}$
$r = 1$	102.20	106.20	99.07	96.87	93.00	84.04	95.31	442.2	236.20
$r = 2$	72.08	71.49	91.47	90.94	86.96	84.74	85.85	149.2	85.24
$r = 4$	70.80	70.15	85.34	85.91	84.87	84.56	81.77	125.8	85.62
$r = 6$	72.70	69.37	85.39	85.31	85.48	84.66	81.67	113.7	84.43
$r = 8$	74.11	70.48	87.18	86.15	87.22	86.01	82.43	108.8	82.57

Table 6.1: Sensitivity exploration in the case of 16 colors.

It can be appreciated that for relatively good initial conditions, SCL partially recovers after the second image. However, it remains performing worse than when starting from the optimal initial codebooks of Figures 6.1(a) and 6.1(b). The worst response corresponds to the worst initial condition, the effect of the bad initialization is propagated through all the image sequence. We will show, reproducing this experiment with SOM, FLVQ and SCS, that they effectively improve the robustness of SCL.

6.2.4 Sensitivity to control parameters of SOM, FLVQ and SCS

We have started the experimental study of SOM, FLVQ and SCS performing a sensitivity experiment. The initial condition is the Heckbert codebook of the first image, and the adaptation is performed as proposed in The sensitivity experiment is restricted to the distortion results over the sample data. From these results we decide the optimal setting of the neighboring function control parameters, and we use the corresponding codebook to perform the quantization of the full size images in the sequence, assuming that the optimality will extrapolate from the sample to the full size image.

Table 6.1 shows the global distortion results on the sequence of samples of 1600 pixels when quantized to 16 colors with SOM, FLVQ and SCS under different combinations of the setting of the initial neighboring parameters and convergence rate to SCL. The inspection of the table reveals that the worst results are obtained when there is no proper SCL phase ($r = 1$). In general the fast convergence to SCL improves the results. In the SOM the initial radius is a secondary factor of performance. In the FLVQ, the initial exponent is most significative when $m_0 = 2$, for $m_0 > 2$ the FLVQ is rather insensitive to this parameter. In the SCS the effect of the initial standard deviation is strong. The local estimations $\widehat{\sigma_{i,0}}$ (computed to give the maximum non overlapping

	SOM		FLVQ		SCS	
	$v_0 =$		$m_0 =$		$\sigma_0 =$	
	1	128	10	2	0.1	$\widehat{\sigma_{i,0}}$
$r = 1$	385.5	394.8	387.5	350.5	439.5	527.5
$r = 2$	302.4	304.5	351.8	341.2	361.7	359.6
$r = 4$	294.2	294.6	352.2	346.3	342.9	352.0
$r = 6$	295.5	291.2	343.4	349.0	329.4	349.8
$r = 8$	299.2	288.7	357.0	349.9	323.1	355.6

Table 6.2: Sensitivity exploration in the case of 16 colors.

confidence balls of 95% confidence) give good performances, although not the optimal. The optimal rates of convergence to SCL is $r = 6$ in all cases The optimal initial neighboring parameters are $v_0 = 8$, $m_0 = 2$ and $\sigma_0 = 0.1$.

In Table 6.2 shows the global distortion results on the sequence of samples of 25600 pixels when quantized to 256 colors. The performance in the table, improves with the rate of convergence to SCL. The initial values of the neighboring function parameters are secondary factors. The optimal convergence rates to SCL are $r = 8, 2, 8$ for SOM, FLVQ and SCS, respectively. The optimal initial values are $v_0 = 128$, $m_0 = 2$ and $\sigma_0 = 0.1$. Figure 6.2 shows the results of the application of the SOM, FLVQ and SCS under the optimal settings of the control parameters for each algorithm deduced from the results in Tables 6.1 and 6.2.

Figures 6.2(a) and 6.2(b) show the distortion of the quantization to 16 and 256 colors of each full size image in the image sequence. The SOM gives the best results. In general, the three algorithms improve over the SCL. Figures 6.2(c) and 6.2(d) show the relative distortion (computed similarly to (6.1)). It can be appreciated that the one-pass SOM sometimes finds better color representatives than the optimal application of Heckbert. Finally, to stand out the improvement over SCL, 6.2(e) and 6.2(f) show the per image substraction of the distortion results of the algorithms from the ones of SCL. The three algorithms show significative improvements.

6.2.5 Sensitivity to initial codebooks

The experiments discussed in this subsection extend the works reported in [109] and [110] were we have already explored the sensitivity of the SOM to problem and control parameters. In the experiments of previous subsections, the adaptive process starts in the second image, assuming as initial codebook the Heckbert codebook of the first image, which is a rather good initial condition. Now

codevectors number	Initialization	SCL	SOM	FLVQ	SCS
16	*Heckbert*	5686	4733	5689	5441
	in Sample	6512	4733	6398	5970
	in RGBbox	10320	4740	10470	10080
	Threshold	6044	4740	5884	5560
256	*Heckbert*	1412	1209	1414	1338
	in Sample	1653	1227	1665	1607
	in RGBbox	2191	1223	2227	1833
	Threshold	1653	1204	1693	1642

Table 6.3: Sensitivity to initial conditions.

we consider the response of the FBAVQ with SOM, FLVQ and SCS to worse initial conditions and compare them to the results obtained with SCL. The initial codebooks are the same used in 6.1(e) and 6.1(f) of . The setting of the neighboring function parameters is the same applied to obtain Figure 6.2. As in these figures the results given are the per image distortion of the quantization of the full images with the codebooks computed from the samples. Global distortion results are reproduced in Table 6.3.

Figures 6.3(a) and 6.3(b) give the results of the FLVQ with 16 and 256 color representatives, respectively. It can be appreciated that the improvements respect to SCL are minor. Figures 6.3(c) and 6.3(d) give the results of the SCS with 16 and 256 color representatives, respectively. They show a systematic improvement over SCL, although not a very big one. Finally, the best results are obtained with the SOM, which shows an astonishing robustness. Whereas the effect of the bad initial conditions is propagated by the FLVQ and SCS through the whole sequence, the SOM collapses almost completely to the optimal behavior from the second image. This robustness has great practical implications for real time video processing. It allows almost arbitrary initializations.

6.3 Conclusions and further work

We have started recalling the definitions of Clustering and Vector Quantization in the stationary case, to arrive to the definition of Frame-Based Adaptive Vector Quantization (FBAVQ). The problem of Adaptive Color Quantization of image sequences fits in the framework of FBAVQ. Competitive Neural Network algorithms are stochastic gradient descent minimizations of given objective functions. These objective functions can be seen to converge to the Euclidean

distortion in some cases. These functional convergence depends on some neighboring function parameters. We consider the SOM, FLVQ and SCS algorithms as minimization processes whose final goal is the minimization of the Euclidean distortion. They perform that by minimizing a sequence of objective functions that converge to the Euclidean distortion. This process is aimed to obtain greater robustness to initial conditions and enhanced performance compared with the bare SCL. For each algorithm we identify the parameters of the neighboring functions that control the functional convergence. We apply an exponential schedule of these parameters that results in fast adaptation, enhanced performance and robustness. Fast and robust adaptation allow the application of SOM, FLVQ and SCS to approach real time FBAVQ. We tested them on the Adaptive Color Quantization of image sequences. From the computational experiments, the SOM appears as the more robust and efficient algorithm.

We plan to work on the extension of the approach presented here to other neural and fuzzy algorithms for clustering. Other line of work is the application of this fast scheduling to higher dimensional problems, and to embed Adaptive Color Quantization into real time video processing systems. Our approach is computationally competitive: its complexity grows linearly with the dimension of the space and the codebook and sample sizes.

6.4 Sensitivity to sample size¿?

A random sample of 1600 image pixels is presented once to the Neural Network for both size of codevectors $c = 16$ and $c = 256$. The comparison of Figures 2a,3a,4a,5a,6a with Figures 2b,3b,4b,5b,6b shows that the algorithms behave much better in the case of 16 representatives. This difference is attributable to the ratio between the size of the sample and the number of representatives searched. For 256 representatives a sample of 1600 pixels constitutes a very small sample. Some sensitivity experiments that we are performing support this analysis. However, note that the algorithms still perform adaptively most of the time. The SOM and Neural Gas algorithms do imply entropic criteria that try to maximize the distribution of the cluster representatives. To introduce this kind of criteria in the application of the SCL and Soft Competition, we have considered the heuristic [153] of penalizing the most numerous clusters by weighting the distance to the cluster representative: $d(\mathbf{x}, \mathbf{y}_i) = \tau_i \|\mathbf{x} - \mathbf{y}_i\|^2$, where $\tau_i = \sum_{k=1}^{\tau} \delta_i(\mathbf{x}(k))$ for SCL (also called FSCL in [3]), and $\tau_i = \sum_{k=1}^{\tau} H_i(\mathbf{x}(k), \mathbf{Y}(k))$ for the SCS. The results of this approach are show in Figures 6.4(c, d) and Figures 6.5(c, d). It can be appreciated that the use of this weighted distance has some effect, greater in the case of SCS.

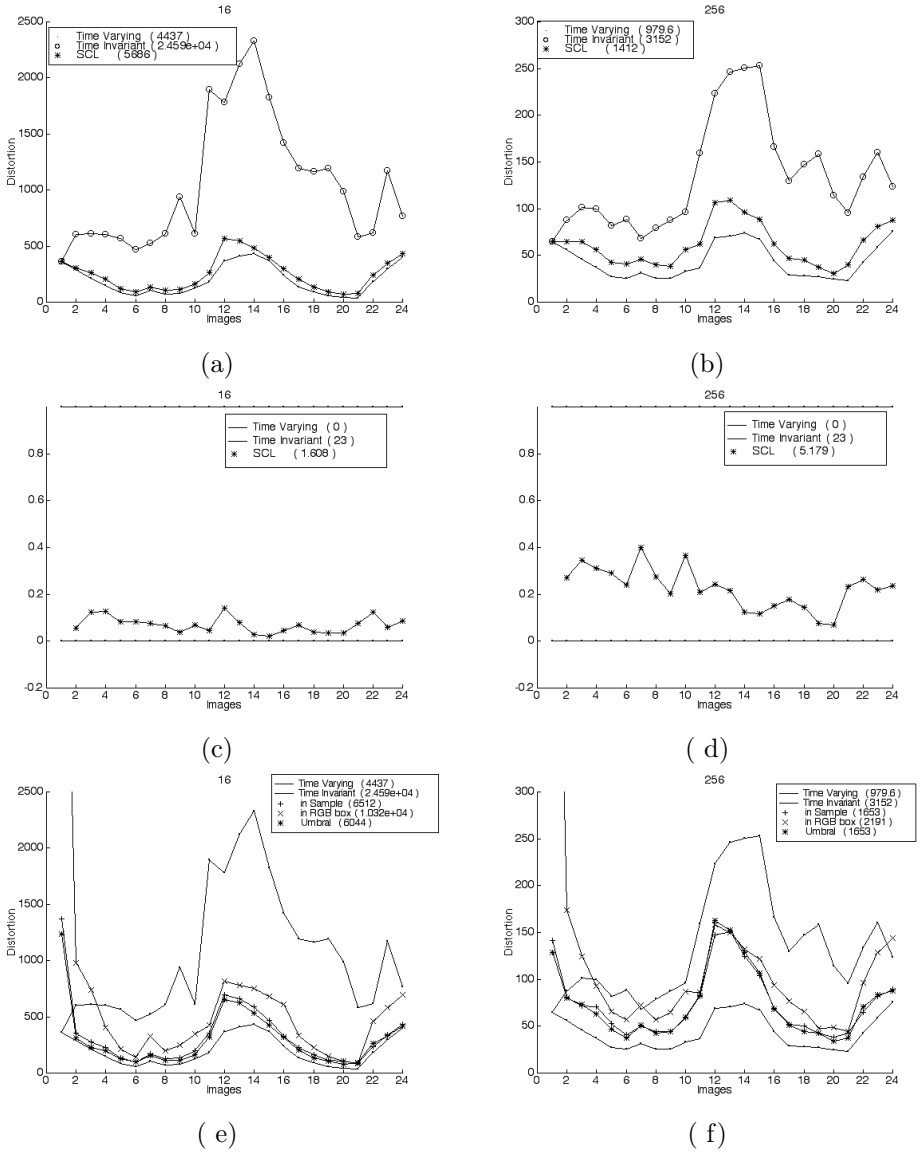


Figure 6.1: Per image distortion results of the quantization of the full size images with the codebooks computed by the SCL on the image samples. (a, c, e) 16 color representatives and samples of 1600 pixels. (b, d, f) 256 color representatives and samples of 25600 pixels. (a, b) distortion results. (c, d) relative distortion results. (e, f) sensitivity results starting from initial codebooks different from the Heckbert codebook of the first image.

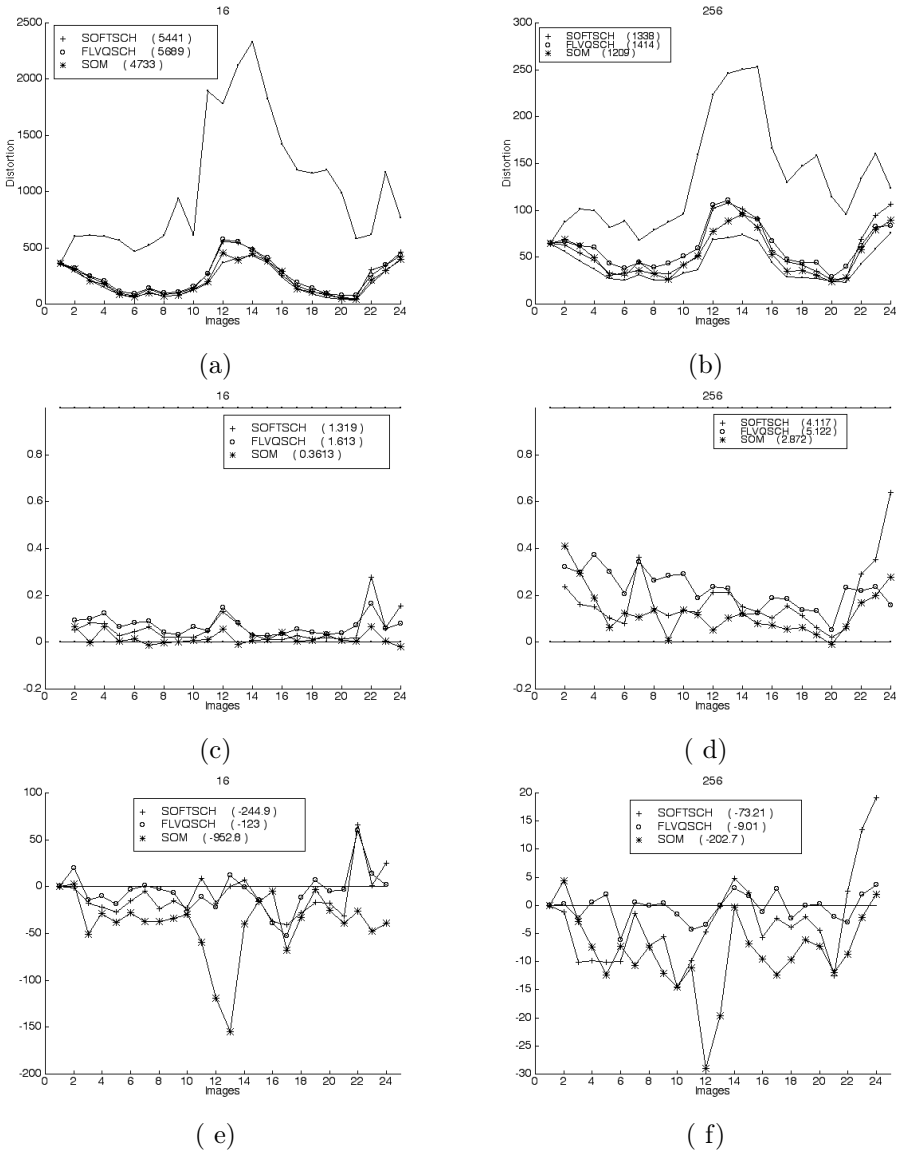


Figure 6.2: Distortion results of the quantization of the full size images in the sequence with the codebooks obtained by the SOM, FLVQ and SCS with optimal settings of neighborhood parameters deduced from Tables 6.1 and 6.2. (a, c, e) 16 color representatives computed from the samples of 1600 pixels. (b, d, f) 256 color representatives computed from the samples of 256 pixels. (a, b) absolute distortion results per image. (c, d) relative distortion results per image. (e, f) per image substraction from the SCL distortion results.

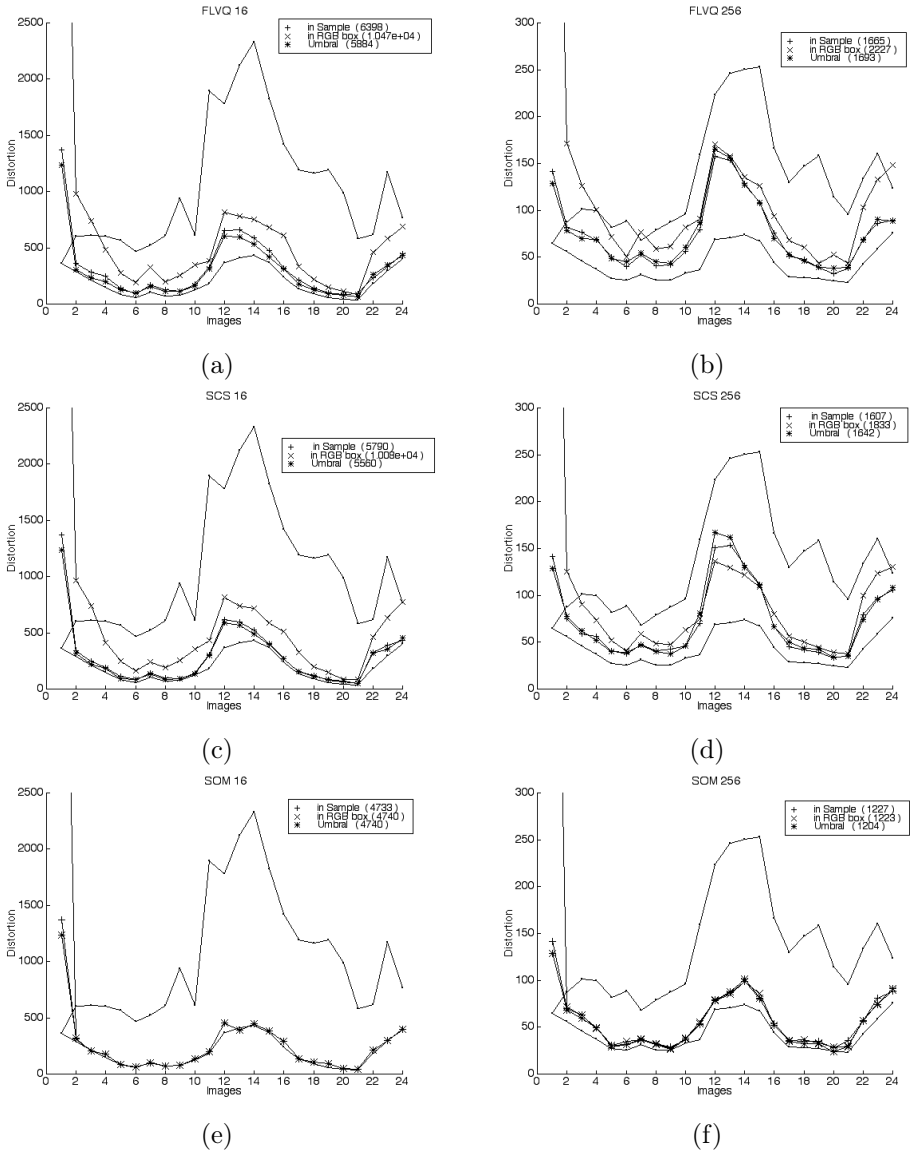


Figure 6.3: Per image distortion results that show the sensitivity to initial conditions of the SOM, FLVQ and SCS. The codebooks for the first image selected as discussed in the text. The neighboring function parameters set as in Figure 6.2. (a, c, d) 16 color representatives computed from the samples of 1600 pixels. (b, d, f) 256 color representatives computed from the samples of 256 pixels. (a, b) distortion results of the FLVQ. (c,d) distortion results of the SCS. (e, f) distortion results of the SOM.

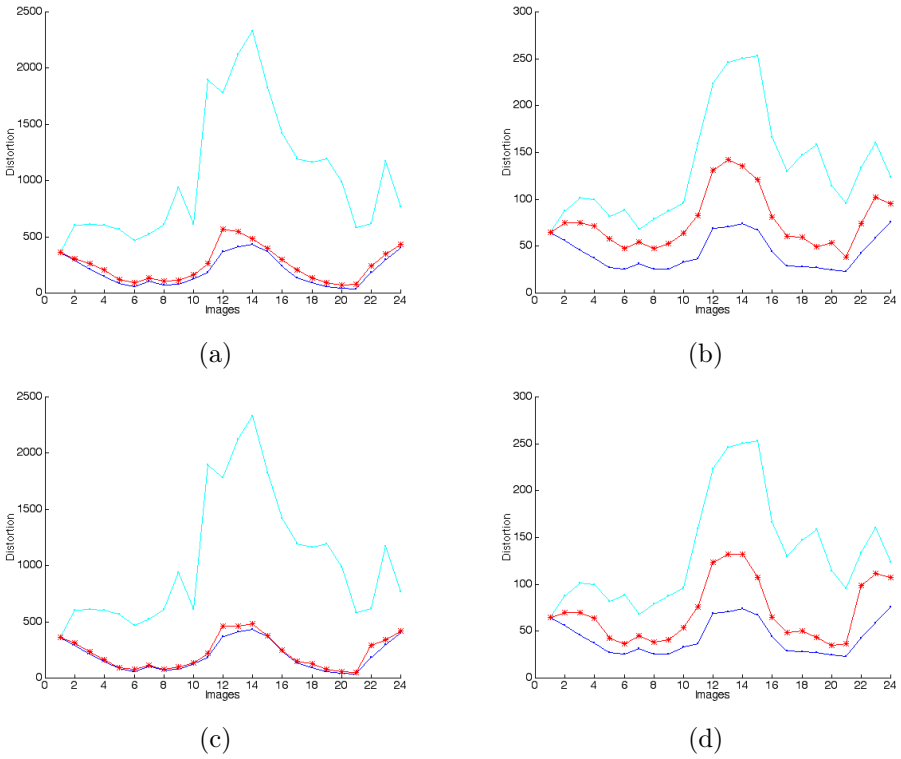


Figure 6.4: Distortion results of the quantization of the full size images in the sequence with the codebooks obtained by the SCL for (a) 16 and (b) 256 color representatives. SCL with penalized distance or FSCL for (c) and (d) 256 color representatives.

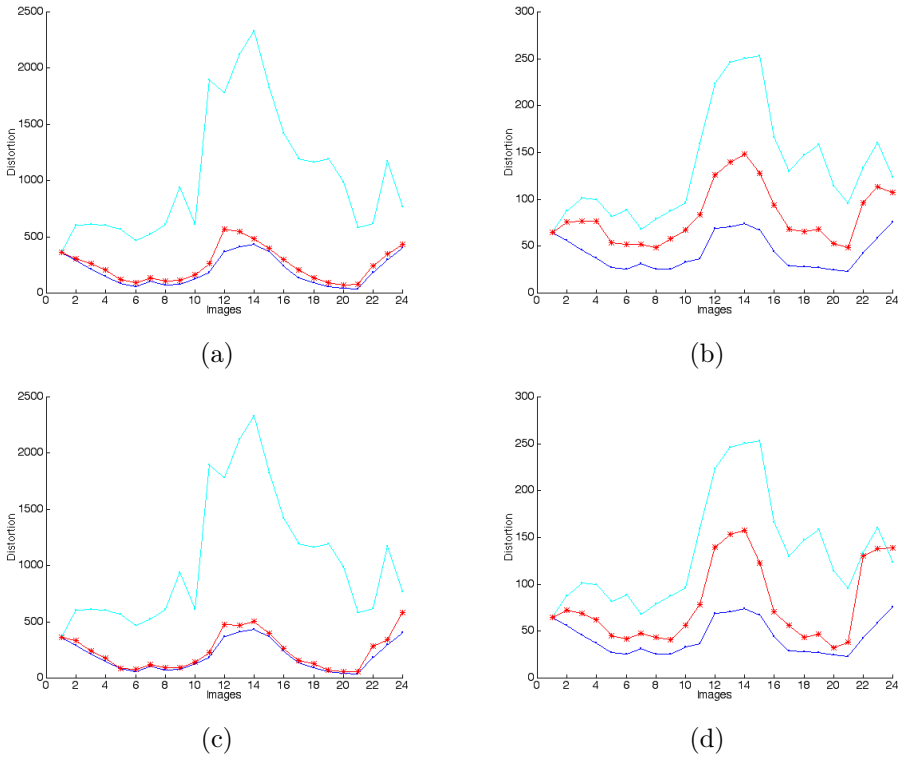


Figure 6.5: Distortion results of the quantization of the full size images in the sequence with the codebooks obtained by the SCS for (a) 16 and (b) 256 color representatives. SCS with penalized distance for (c) and (d) 256 color representatives

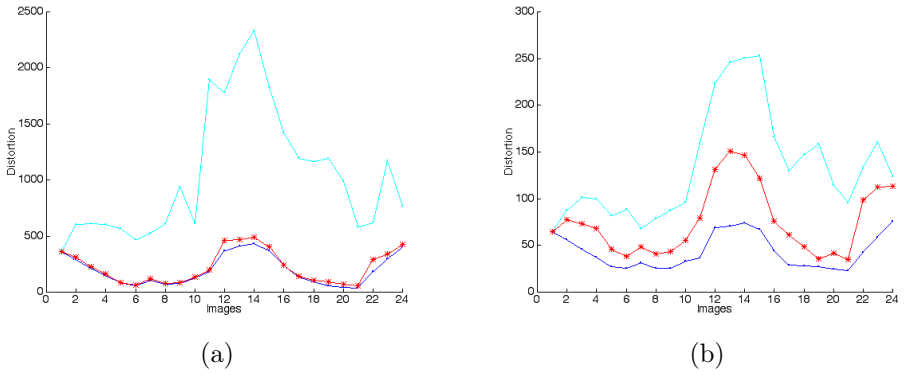
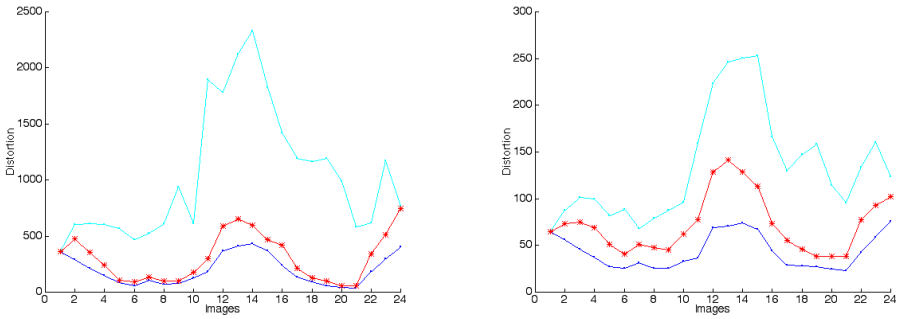


Figure 6.6: Distortion results of the quantization of the full size images in the sequence with the codebooks obtained by the SOM (a) 16 color representatives and (b) 256 color representatives.



(a) (b)

Figure 6.7: Distortion results of the quantization of the full size images in the sequence with the codebooks obtained by the NG (a) 16 color representatives and (b) 256 color representatives.

Chapter 7

A sensitivity analysis of the SOM for NSC

In this chapter we recall [110] the study on the sensitivity of the Self Organizing Map (SOM) parameters in the context of the one-pass adaptive computation of cluster representatives over non-stationary data. The paradigm of Non-stationary Clustering (NSC) is represented by the problem of Color Quantization of image sequences has been introduced in Chapter .

Section 7.1 gives some introductory remarks. Section 7.2 specifies the application of the SOM to the one-pass adaptive computation of cluster representatives in the general NSC problem. Section 7.3 discusses the experimental results obtained. Finally, section 7.4 gives some conclusions.

7.1 Introduction

Cluster analysis and Vector Quantization have applications in signal processing, pattern recognition, machine learning and data analysis [97, 138, 73, 76, 148, 89]. A vast number of approaches have been proposed to solve these problems, among them Competitive Neural Networks [192, 3, 168]. Conventional formulations assume that the underlying stochastic process is stationary and that a given set of sample vectors properly characterizes this process. Non-stationary processes are dealt with applying a predictive approach to reduce the non-stationary problems to the stationary framework [97]. This chapter continues on the exploration of the efficiency of competitive neural networks as one-pass adaptive algorithms for the computation of clustering representatives in the non-stationary case whithout knowledge of a time dependence model [112, 109]. [110] and [109] focus on the Self Organizing Map (SOM) [168]. The one-pass adaptation framework is not very common in the neural networks literature, in fact the only related references that we have found are [53, 180]. This restriction imposes very strong computational limitations. The effective scheduled sequences of the learning parameters applied to meet the fast adaptation requirement fall far

from the theoretical conditions for convergence. A sensitivity analysis is justified in order to asses the behaviour of the SOM under a wide range of conditions and parameter values.

Color Quantization of image sequences is an instance of the Non-stationary clustering problem. Color Quantization has practical applications in visualisation [141, 207, 185], color image segmentation [272], data compression [101] and image retrieval [156]. One-pass adaptation is enforced by the real time constraints of the processing of each image whithin the sequence.

7.2 Adaptive application of SOM to Non-stationary Clustering

Stationary cluster analysis assume that the data is a sample $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ of an stationary stochastic process, whose statistical characteristics will not change in time. Non-stationary Clustering assume that the data come from a non-stationary stochastic process sampled at diverse time instants. That is, the population can be modelled by a discrete time stochastic process $\{X_t; t = 1, 2, \ldots\}$ of unknown joint probability distribution. We do not assume any knowledge of the time dependencies that could allow a predictive approach [97].

A working definition of the Non-stationary Clustering problem could read as follows: Given a sequence of samples $\mathbf{X}(t) = \{\mathbf{x}_1(t), \ldots, \mathbf{x}_n(t)\}; t = 0, 1, \ldots$ of the population obtain a corresponding sequence of partitions of each sample that consists of a sequence of sets of disjoint clusters $P(\mathbf{X}(t)) = \{\mathcal{X}_1(t), \ldots, \mathcal{X}_1(t)\}$. This sequence of partitions minimizes a criterium function along time $C = \sum_{t \geq 0} C(t)$. In the general statement of the problem the difinition of the criterium function is based on the definition of an appropriate distance. We follow the conventional approach of using the Euclidean distance. We consider a sequence of codevectors or cluster representatives $\mathbf{Y}(t) = \{\mathbf{y}_1(t), \ldots, \mathbf{y}_n(t)\}$ such that the desired partitions are defined by the nearest (Euclidean) representative.

$$\mathbf{x}_j(t) \in \mathcal{X}_i(t) \Leftrightarrow i = \arg \min_{k=1,\ldots,c} \left\{ \|\mathbf{x}_j(t) - \mathbf{y}_k(t)\|^2 \right\}$$

The criterium function that we will consider at each time step is, therefore, the distortion (or whithin cluster variance)

$$C(t) = \sum_{j=1}^{n} \sum_{i=1}^{c} \|\mathbf{x}_j(t) - \mathbf{y}_i(t)\|^2 \delta_i(\mathbf{x}_j(t), \mathbf{Y}(t)),$$

$$\delta_i(\mathbf{x}_j(t), \mathbf{Y}(t)) = \begin{cases} 1 & i = \arg \min_{k=1,\ldots,c} \left\{ \|\mathbf{x}_j(t) - \mathbf{y}_k(t)\|^2 \right\} \\ 0 & \text{otherwise} \end{cases}.$$

The adaptive computation of the cluster representatives along time can be stated as follows:

- At time t take as initial cluster representatives the ones already computed from the sample of the process at time $t-1$.
- Use the sample vectors $\mathbf{X}(t) = \{\mathbf{x}_1(t), \ldots, \mathbf{x}_n(t)\}$ to perform an adaptive computation leading to the new estimates of the cluster representatives. Specifically, we use the Self Organizing Map (SOM) in this chapter.

For notational simplicity, let us denote $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ the sample of the process at a given time instant. We recall that the SOM is a particular case of the general Competitive Neural Network algorithm:

$$\mathbf{y}_i(\tau+1) = \mathbf{y}_i(\tau) + \alpha_i(\tau) V_i(\mathbf{x}(\tau), \mathbf{Y}(\tau)) (\mathbf{x}(\tau) - \mathbf{y}_i(\tau)); \mathbf{x}(\tau) \in \mathbf{X}, 1 \leq i \leq c$$

where τ is the order of presentation of the sample vectors. One-pass adaptation means that each sample vector will only be presented once for the learning of representatives. In their general statement, Competitive Neural Networks are designed to perform stochastic gradient minimisation of some distortion-like function. In order to guarantee theoretical convergence, the (local) learning rate must comply with the conditions:

$$\lim_{\tau \to \infty} \alpha_i(\tau) = 0,$$

$$\sum_{\tau=0}^{\infty} \alpha_i(\tau) = \infty,$$

$$\sum_{\tau=0}^{\infty} \alpha_i^2(\tau) < \infty.$$

However, the "one pass" adaptation schedule does not comply with these conditions implying very lengthy adaptation processes. In the experiments, the learning rate follows the expression :

$$\alpha_i(\tau) = 0.1 \left(1 - \frac{\sum_{k=1}^{\tau} \delta_i(\mathbf{x}(k), \mathbf{Y}(k))}{n}\right).$$

This expression implies that the learning rate decreases proportionally to the number of times that a codevector "wins" the competition. The adaptation induced by the neighbouring function does not alter the local learning rate. It also implies that a local learning rate only reaches the zero value if the codevector "wins" for all the sample vectors. This expression of the learning rate is the best we have found that fits in the one-pass adaptation framework. Obviously the sequences of the learning rate parameters given by it do not comply with the conditions that ensure the theoretical convergence of the stochastic gradient minimization algorithm.

We recall that the function $V_i(\mathbf{x}, \mathbf{Y})$ is the so-called *neighbouring functio*n. According to its definition, the shape of the distortion function minimized and, therefore, the qualitative properties of the learning rule equilibria can be very different. In the case of the SOM the neighbouring function is defined over the

space of the neuron (cluster) indices. In our works we have assumed a 1D topology of the cluster indices. The neighbourhoods considered decay exponentially following the expression:

$$V_i\left(\mathbf{x}\left(\tau\right),\mathbf{Y}\left(\tau\right)\right)=\begin{cases}1 & |i^*-i|\leq\left\lceil\left(v_0+1\right)e^{\frac{1}{n}v^*\tau\log\left(\frac{1}{v_0+1}\right)}\right\rceil\\ 0 & \text{otherwise}\end{cases};1\leq i\leq c\,,$$

where $i^*=\arg\min_{k=1,\ldots,c}\left\{\left\|\mathbf{x}\left(\tau\right)-\mathbf{y}_k\left(\tau\right)\right\|^2\right\}$.

The size of the sample considered at each time instant is n. The initial neighbourhood radius is v_0. The expression ensures that the neighbouring function reduces to the simple competitive case (null neighbourhood) after the presentation of the first $1/v^*$ vectors of the sample. Along the experiments v^* is the neighbourhood reduction rate.

7.3 Experimental sensitivity results on the Color Quantization of an image sequence

The sequence of images used for the computational experiments is presented in Appendix . As a benchmark non adaptive algorithm we have used a variation of the algorithm proposed by Heckbert [141] as implemented in MATLAB. This algorithm has been applied to the entire images in the sequence under stationary and non-stationary assumptions. Figure 7.1 shows the distortion results of the Color Quantization of the experimental sequence to 16 and 256 colors based on both applications of the Heckbert algorithm. The curve $\left\{C^{TV}\left(t\right);t=1,\ldots,24\right\}$, named *Time Varying Min Var* in the figure is produced assuming the non-stationary nature of the data and applying the Heckbert algorithm to each image independently. The curve $\left\{C^{TI}\left(t\right);t=1,\ldots,24\right\}$, named *Time Invariant Min Var* in the figure, is computed under the assumption of stationarity of the data: the color representatives obtained for the first image are used for the Color Quantization of the remaining images in the sequence. The gap between those curves gives another indication of the non stationarity of the data. Also this gap defines the response space left for truly adaptive algorithms. All the figures giving distortion results for the experimental sequence will include these two curves as a reference frame.

The adaptive application of the SOM assumes that the adaptation process starts with the second image, taking as initial cluster representatives the assumed color representatives for the first image. In the two first experiments the initial codebook was the Heckbert palette for the first image. The adaptation is performed over a random sample of the pixels of each image. In the experiments that foloow, we have tried to explore the sensitivity of the SOM to the following parameters: number of clusters (codebook size), size of the sample taken from each image, neighbouring function parameters: neighbourhood initial size and reduction rate, and, finally, the initial color representatives of the whole process

(the assumed color representatives of the first image). The scheduling of the learning rate remains the same through all the experiments.

The first experiment tries to evaluate the sensitivity of the SOM to the sample size and the number of cluster representatives (codebook size) searched. Two codebook sizes have been considered 16 and 256 colors. The neighbouring function parameters were set to $v_0 = 1$ and $v^* = 4$ for 16 colors, and $v_0 = 8$ and $v^* = 4$ for 256 colors. Figure 7.2 shows the results of the SOM for several sample sizes. These results consist of the sequence of distortions over the image sequence of the Color Quantization using the color representatives computed adaptively by the SOM over the image samples. The first general conclussion that can be drawn from this figure is that the SOM performs adaptively under a wide variety of conditions, but that it is clearly sensitive to the sample size. A closer inspection of the figure leads to the conclussion that the SOM is highly sensitive to the number of color representatives (clusters) searched. The sample sizes 100 for 16 colors and 1600 for 256 have the same ratio of sample size to codebook size (roughly 6:1). However, the response of the SOM in either case is qualitatively very different, it is clearly worse in the 256 colors codebook case. In the case of the the 16 color codebook, as the sample size grows, the distortion curves overlap very fast in near optimal results. In the case of 256 colors this convergence to near optimal results (as the sample size grows) is very smooth. The influence of the sample size seems to be stronger in the 256 colors codebook case. Finally, if we consider the highest sample:codebook ratio that appears in both figures (100:1), we note that the response in the 16 colors codebook case is qualitatively better than in the 256 colors codebook case. Our main conclussion from this first experiment is that the codebook size is the prime factor in the performance of the SOM. Once the codebook size is fixed, the size of the sample used for the one-pass adaptation can be a very sensitive performance factor.

The second experiment was intended to explore the sensitivity of the SOM to the neighbouring function parameters: the initial neighbourhood v_0 and the neighbourhood reduction rate v^*. Not all the combinations of codebook and sample size tested in Figure 7.2 are retried in this experiment. The measure of the behaviour of the color quantizers computed by the SOM is the accumulated distortion along the entire image sequence. This measure was computed from the samples instead of the entire images (the magnitudes of the errors can not be compared between surfaces). This simplification is justified because we are interested in the qualitative shape of the response surface, and because we have observed that the distortion of the color quantization of the entire image is proportional to that of the of the sample. The values of the neighbouring reduction factor tested were $\{1, 2, 3, 4, 5, 8\}$ and $\{1.25, 1.3, 1.5, 2, 3, 4\}$ in the case of 16 and 256 colors, respectively. The initial neighbourhoods considered were $\{1, 2, 3, 4, 5, 8\}$ and $\{2, 8, 16, 32, 64, 128\}$ in the case of 16 and 256 colors, respectively. Figures 7.3 and 7.4 show the results, and Table 7.1 summarizes the experimental design. Shown in the figures are both the response surfaces (Figures 7.3(a), 7.3(c), 7.3(e), 7.4(a), 7.4(c)) and the projections on the experiment axes (Figures 7.3(b), 7.3(d), 7.3(f), 7.4(b), 7.4(d)).

The study of Figures 7.3 and 7.4 confirm the previous assertion of the impor-

		Sample Size						
codebook		100	400	1600	4096	6400	12800	25600
16	surface	fig. 7.3(a)	fig 7.3(c)	fig. 7.3(e)				
	project.	fig. 7.3(b)	fig 7.3(d)	fig. 7.3(f)	–	–	–	–
256	surface			fig. 7.4(a)				fig. 7.4(c)
	project.	–	–	fig. 7.4(b)	–	–	–	fig. 7.4(d)

Table 7.1: Summary of the neighbouring function sensitivity experiment results

tance of codebook and sample size. The sensitivity of the SOM to the setting of the neighbouring function parameters varies strongly with them. In the case of the smaller sample:codebook ratio (6:1) (Figures 7.3(a), 7.4(a)) the response surface has a counter intuitive shape. It appears that for this ratio the best results are obtained with the smaller initial neighbourhoods. This result may be due to fluctuations produced during the reordering phase of the SOM by the combined effect of the sparse distribution of the small sample and the relatively big initial neighbourhood. For a more sensible ratio (100:1), whose results are shown in Figures 7.3(e) and 7.4(c), the response surface has a more natural shape giving the best results for the largest initial neighbourhood. The comparison of Figures 7.3(c) and 7.3(e) confirms the quick convergence of the SOM to the optimal behaviour as the sample:codebook ratio grows, in the case of 16 colors. The examination in both Figures 7.3 and 7.4 of the projections of the surfaces reveals a very clear trend for the neighbourhood reduction rate. In general, a reduction factor such that the neighbourhood disappears after presentation of one quarter of the sample gives the best results in all the cases. After codebook and sample size, the neighbouring reduction rate seems to be the next significant performance factor. With all the other performance factors set to appropriate values, the optimal values of the initial neighbourhood are the largest ones.

The last experiment conducted was the exploration of the sensitivity to the initial codebooks. As said before, the previous experiments were conducted starting the adaptive process in the second image of the sequence, assuming the initial codebook to be the Heckbert palette (Matlab) for the first image. In Figures 7.5 and 7.6 it is shown the response of the SOM to other settings of the initial codebook: a threshold based selection of the sample of image #1 (Thresh), random points in the RGB cube (RGBbox) and a random selection of the sample of image #1 (Sample). For 16 colors the SOM parameters were: sample size 1600, $v_0 = 1$, and $v^* = 4$. For 256 colors sample size was 25600, $v_0 = 128$, and $v^* = 4$. Figures 7.5(a), 7.6(a), 7.6(c) show the distortion along the image sequence of experimental images together with the benchmark results. Let us denote $\left\{C^{SOM}(t)\,; t = 1, \ldots, 24\right\}$ the sequence of distortion values obtained from the color quantizers computed by the SOM starting from a given initial codebook. Figures 7.5(b), 7.6(b), 7.6(d) show these sequences relative to

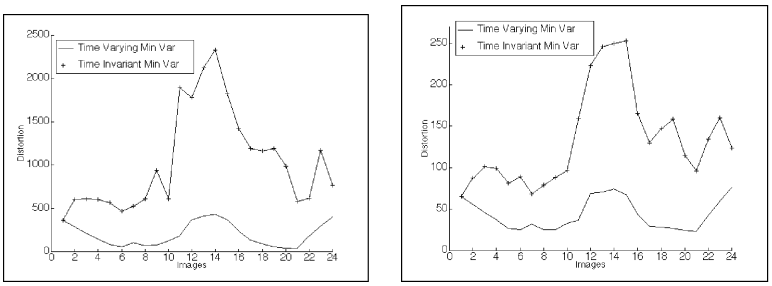
the error committed when assuming stationarity, that is for each initial condition we plot:

$$\left\{ \frac{C^{SOM}(t) - C^{TV}(t)}{C^{TI}(t) - C^{TV}(t)}; t = 1, \ldots, 24 \right\}$$

Figure 7.5 shows that the SOM is quite insensitive to initial conditions for small codebooks. However Figures 7.6(a) and 7.6(b) show a rather high sensitivity to the initial codebook. The obvious hypothesis for this degradation is that our one-pass implementation of the SOM can not perform properly the self-organization phase when the codebook size is relatively large. To test this idea, we have applied a simple ordering by components to the codebooks before starting the adaptation with the SOM. The results are shown in Figures 7.6(c) and 7.6(d). Given a good ordering of the initial cluster representatives, the SOM becomes insensitive to initial conditions regardless of codebook size. We can conclude that the our one-pass SOM is capable of performing fast self-organization in the case of small codebooks, but as the size of the codebook grows it becomes very sensitive to the bad ordering of the initial cluster representatives. The strong influence of the network size (the number of clusters) extends to the ability of our one-pass SOM to recover from bad initial topological orderings of the neurons that incorporate the cluster representatives.

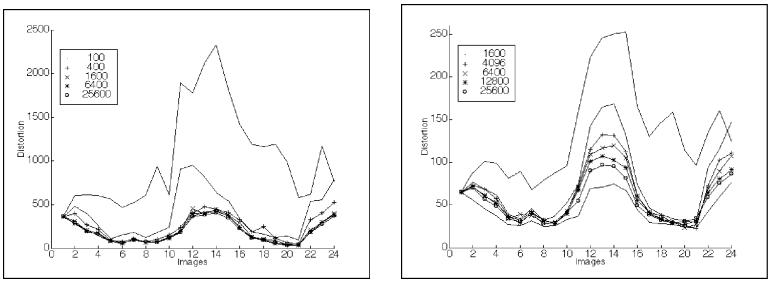
7.4 Conclusions

This chapter has explored the sensitivity of the SOM as a one-pass adaptive algorithm for the computation of cluster representatives in the framework of NSC problems from an experimental point of view. The NSC paradigm is exemplified in the problem of Color Quantization of image sequences. The experiment results reported in this chapter show that the SOM is a very robust algorithm when we try to perform one-pass adaptive computation of cluster representatives in the non-stationary case. In the sensitivity experiments, we have found that the SOM is highly sensitive to the number of clusters searched, that is, to the size of the network to be adapted. The number of clusters searched impose restrictions on the size of the sample used. These two problem parameters condition the response of the SOM to changes in the neighbouring function parameters. Finally, when the SOM is quite insensitive to initial conditions, for small codebook sizes. For larger codebooks, the one-pass SOM is sensitive to the topological ordering of the initial cluster representatives.



(a) (b)

Figure 7.1: Benchmark distortion values obtained with the application of the Matlab implementation of the Heckbert algorithm to compute the color quantizers of 16 (a) and 256 (b) colors of the images in the experimental sequence.



(a) (b)

Figure 7.2: Distortion results obtained with the adaptive application of SOM over samples of diverse sizes to compute the color quantizers of (a) 16 (with $v_0 = 1$ and $v^* = 4$) and (b) 256 colors (with $v_0 = 8$ and $v^* = 4$).

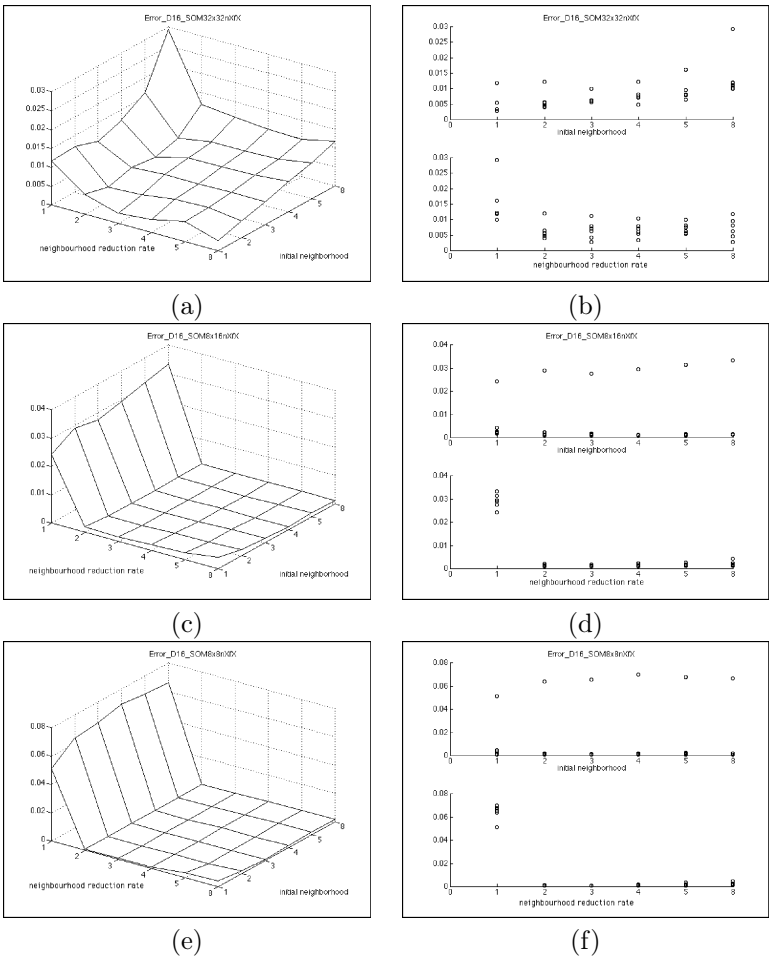
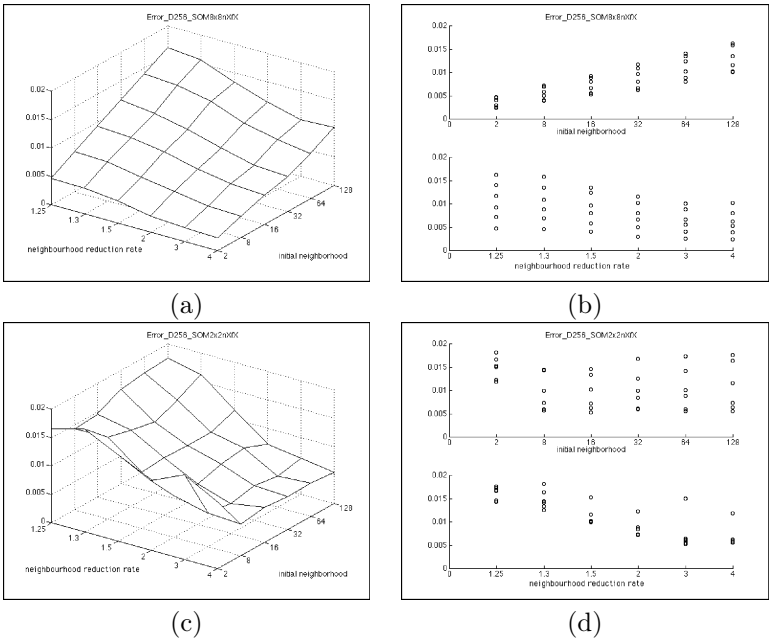


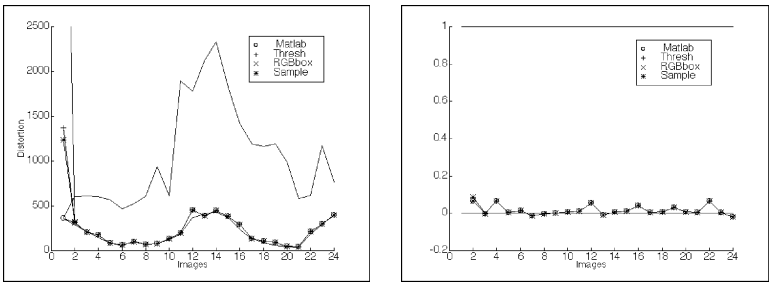
Figure 7.3: Sensitivity to the neighbouring function parameters v_0 and $v^{(0)}$ of the SOM applied to compute the color quantizers of 16 colors (see Table 7.1), measured by the accumulated distortion along the experimental sequence.



(a) (b)

(c) (d)

Figure 7.4: Sensitivity to the neighbouring function parameters v_0 and v^* of the SOM applied to compute the color quantizers of 256 colors (see Table 7.1), measured by the accumulated distortion along the experimental sequence.



(a) (b)

Figure 7.5: Distortion of the color quantizers of 16 colors computed by the adaptive application of the SOM starting from several intial cluster representatives (sensitivity to initial conditions) (a) absolute values, (b) normalized relative to the stationarity assumption.

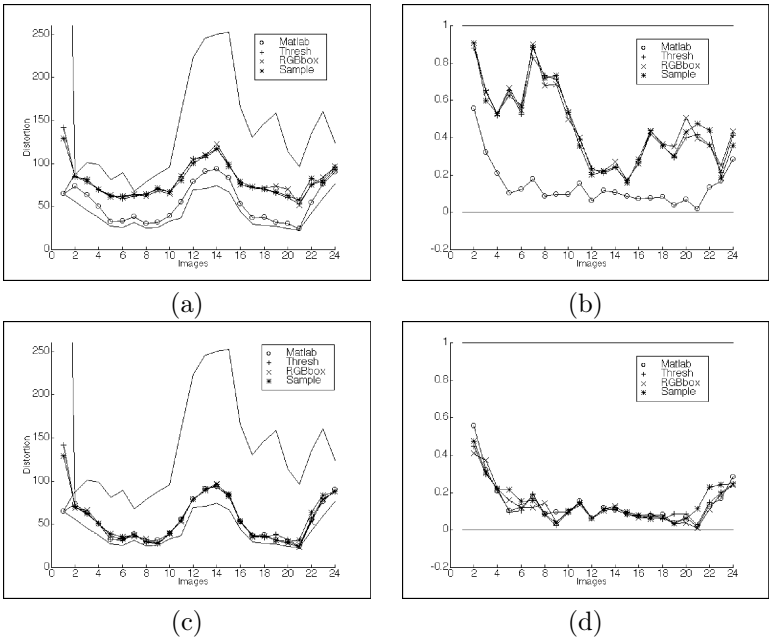


Figure 7.6: Distortion of the color quantizers of 256 colors computed by the adaptive application of the SOM starting from several intial cluster representatives (sensitivity to initial conditions) (a), (b) unprocessed initial cluster representatives, (c), (d) the same initial cluster representatives ordered before starting the adaptation process.

Chapter 8

Convergence of the SOM from the point of view of GNC

We want to present controversial empirical results about the relative convergence of batch and online neural network vector quantization (VQ) learning algorithms. Convergence of the Self-Organizing Map (SOM) and Neural Gas (NG) is usually contemplated from the point of view of stochastic gradient descent (SGD) algorithms of an energy function. SGD algorithms are characterized by very restrictive conditions that produce a slow convergence rate. Also they are local minimization algorithms, very dependent on the initial conditions. It is the commonplace belief that online algorithms have a very slow convergence, while the batch are faster and give better results. However, some empirical results show that one-pass on-line training realizations of SOM and NG may perform comparable to more careful (slow) realizations. Moreover, other empirical works suggest that SOM is quite robust against initial conditions. In both cases the performance measure is the quantization distortion. That empirical evidence leads us to propose that the appropriate setting for the convergence analysis of SOM, NG and similar competitive artificial neural network clustering algorithms is the theory of Graduated Nonconvexity (GNC) algorithms.

Section 8.1 gives an introduction to the chapter. In Section 8.2, we present the definitions of online and batch versions of SOM and NG. Section 8.3 discusses the formulation of the SOM and NG as GNC algorithms. Section 8.4 gives the results the experimental results over several datasets. Finally, Section 8.5 is devoted to conclusions.

8.1 Introduction

Both SOM and NG algorithms have the appearance of stochastic gradient descent (online) algorithms [89] in their original definitions, that is, whenever an input vector is presented, a learning (adaptation) step occurs. It has been shown

that an online version of the NG algorithm can find better local solutions than the online SOM [193]. Online realizations are very time consuming due to the slow convergence rate of the stochastic gradient descent. To speed up computations, there have been proposed batch versions of both algorithms. Batch realizations correspond to deterministic gradient descent algorithms. The parameter estimation is performed using statistics computed over the whole data sample. The batch version of SOM was already proposed in [170] as a reasonable speed-up of the online SOM, with minor solution quality degradation. In the empirical analysis reported in [82], the main batch SOM drawback is its sensitivity to initial conditions, and the bad organization of the final class representatives, that may be due to poor topological preservation. Good execution instances of the batch SOM may improve the solutions given by the online SOM. On the other hand, the online SOM is robust against bad initializations and provides good topological ordering, if the adaptation schedule is smooth enough. The batch version of the NG algorithm has been studied in [298] as a clustering data algorithm. It has been proposed as a convenient speed-up of the online NG too.

Both the online and batch algorithm versions imply the iteration over the whole sample several times. On the contrary, one-pass realizations visit only once the sample data. This adaptation framework is not very common in the neural networks literature; in fact, the few related references that we have found [180, 92, 54] come from the signal processing and speech coding literature. The effective scheduled learning parameter sequences applied to meet the fast adaptation requirements of one-pass training fall far from the theoretical convergence conditions of SGD algorithms. However, in the computational experiments shown in this chapter, the distortion results are competitive with the conventional SOM and NG online and batch versions. If we take into account the computation time, the superior performance of the one-pass realization becomes spectacular.

SGD algorithms are local minimization algorithms, therefore sensitive to the initial conditions. However, the works reported in [42] show that the SOM can be very insensitive to initial condition variability when the goal is VQ design. These results lead us to think that may be other frameworks for the study of SOM and NG convergence better suited than the theory of SGD algorithms. We postulate that both SOM and NG are instances of the Graduated Nonconvexity (GNC) algorithms [40, 154, 202, 203, 204], which are related to the parameter continuation methods [9] (more detailed information in Appendix A). GNC algorithms try to solve the minimization of a non-convex objective function by the sequential search of the minima of a one-parameter family of functions, which are morphed from a convex function up to the nonconvex original function. In the SOM and NG the neighborhood control parameters may be understood as performing the role of graduating the nonconvexity of the energy function minimized by the algorithm. Therefore the training of both the SOM and the NG can be seen as a continuation of the minimum of a sequence of energy functions starting from a convex one and ending with the highly nonconvex distortion function.

8.2 Algorithm definitions

Each algorithm applied below has some control parameters, like the learning ratio, the neighbourhood size and shape, or the temperature. The online realizations usually modify their values following each input data presentation and adaptation of the codebook. The batch realizations modify their values after each presentation of the whole input data sample. Both online and batch realizations imply that the input data set is presented several times. On the contrary, the one-pass realizations imply that each input data is presented at most once for adaptation, and that the control parameters are modified after each presentation. In the experiments, the learning rate follows the expression [60]:

$$\alpha(t) = \alpha_0 \left(\frac{\alpha_N}{\alpha_0} \right)^{\frac{t}{N}}$$

where α_0 and α_n are the initial and final value of the learning rate, respectively. Therefore after N presentations the learning rate reaches its final value.

Both the SOM and NG are particular cases of the general Competitive Neural Network algorithm. The definitions of the online versions of SOM and NG algorithms has been gathered in Chapter 2 in Section 2.3. For one-pass versions go to Section . Next the batch versions.

8.2.1 Batch versions of SOM and NG

So called batch algorithms correspond to deterministic gradient descent algorithms. The adaptation is performed based on the whole data sample. Goal: speed up online versions with minor solution quality degradations.

Kononen's Batch Map [170, 168] defined the batch version of SOM algorithm. Among its advantages, there is no learning rate parameter and the computation is faster than the conventional online realization. This algorithm can be viewed as the LBG algorithm [186] plus a neighbouring function. The input space is partitioned into Voronoi regions associated to unit weights, and the input data sample is partitioned accordingly:

$$V_i(t) = \{ \mathbf{x} \in \mathbf{X} \,|\, w(\mathbf{x}) = i \}$$

where t corresponds to the iteration number over the sample and unit weights are fixed during iteration, so that $w(\mathbf{x})$ is the winning unit index **??** . The estimation of the unit weights is performed computing an arithmetic mean over the unit Voronoi region and those of its neighbor units:

$$\mathbf{y}_i(t+1) = \sum_{\mathbf{x} \in U_i(t)} \frac{\mathbf{x}}{|U_i(t)|}$$

where

$$U_i(t) = \bigcup_{|j-i| \leq h(t)} V_j(t)$$

and $|U_i|$ means the cardinality of U_i. As in the other versions, to determine the radius of the neighborhood we applied the following expression:

$$h(t) = \left\lceil h_0 \left(\frac{h_N}{h_0} \right)^{t/\tau_N} \right\rceil - 1$$

The expression ensures that the neighboring function reduces to h_N after τ iterations. This final neighborhood radius is equivalent to the LBG algorithm: the unit weight is computed as the arithmetic mean of its corresponding Voronoi region. It must be noted that the neighboring function of the Batch SOM is equivalent to the crisp neighborhood for the online training.

A definition of Batch NG [298] arises from the thought of changing the contribution of each input vector to the unit weight estimation as a function of the ordering of the unit relative to the input vector, freezing the online realization of NG. The estimation of the unit weights in the Batch NG reads as follows:

$$\mathbf{y}_i(t+1) = \frac{\sum_{\mathbf{x}} H_i(\mathbf{x}, \mathbf{Y}(t), \lambda(t))\, \mathbf{x}}{\sum_{\mathbf{x}} H_i(\mathbf{x}, \mathbf{Y}(t), \lambda(t))}$$

As with the Batch SOM, the Batch NG converges to the LBG algorithm: only the Voronoi region corresponding to each codevector contributes to its estimation when the temperature reaches its final value. The ranking function and λ parameter are equal than in the one-pass case. The neighbour-region contribution decays exponentially due to the evolution of λ according to the following expression:

$$\lambda(t) = \lambda_0 \left(\frac{\lambda_N}{\lambda_0} \right)^{\frac{t}{N}}$$

Where λ_0 and λ_N its initial and final value. The expression ensures that the neighbouring function reduces to the simple competitive case (null neighbourhood) as it happens with SOM.

8.3 SOM and NG as GNC algorithms

8.3.1 GNC definitions

GNC algorithms appear in the field of image and signal processing for segmentation, restoration and filtering applications. As such, there is some need to adapt the vocabulary and general formulation to the SOM and NG. The basic formulation of the GNC approach [40, 202, 204] is that the function to be minimized is the MAP estimate of a sampled surface corrupted by additive noise $M(x) = D(x) + N(x)$. This MAP estimate $p(R = D|M)$ is obtained minimizing the energy:

$$E[R] = -\log p(M|D = R) - \log p(D = R) = E_d[R] + E_s[R]$$

where $E_d[R]$ is the *data term* and $E_s[R]$ is the *smoothness term*. The data term is quadratic under the usual signal independent Gaussian noise assumption, and the smoothness term express any *a priori* information about the surface. In [202] the smoothness term is formulated over the surface gradient. The GNC function general formulation is:

$$E[R] = \sum_x (M(x) - R(x))^2 + E_s[R] \tag{8.1}$$

where the smoothness term depends on some parameter $E_s[R] = f_\sigma(R)$. The key of GNC methods is that the function to be minimized $E[R]$ is embedded in a one-parameter family of functionals $E_\sigma[R]$ so that the initial functional is convex, and the final functional is equivalent to the original function $E_0[R] = E[R]$. The minimization is performed tracking the local minimum of $E_\sigma[R]$ from the initial to the final functional. One key problem in GNC is to ensure that the initial functional is convex [202]. Other problem is to ensure that there are no bifurcations or other effects that may affect the continuation process. If the initial functional is convex, the algorithm becomes independent of the initial conditions because it will be feasible to obtain the global minimum of the initial functional regardless of the initial condition, thereafter the continuation of the local minima will be strongly determined. If there are no bifurcations or other effects in the continuation process, then the global minimum of the initial functional can be tracked to a global minimum of the target functional. It must be recalled that one of the properties that the SOM and NG show over the bare SCL algorithms is the robustness against bad initial conditions [42], given a powerful clue to interpret them as GNC algorithms. It seems that for NG and SOM it is very easy to ensure the convexity of the initial functional and that the continuation is also an easy process

8.3.2 SOM and NG functionals.

The NG was proposed [193] as the minimization of the following functional,

$$E_{ng}(w, \lambda) = \frac{1}{2C(\lambda)} \sum_{i=1}^{M} \int d^D v P(v) h_\lambda(k_i(v, w)) (v - w_i)^2$$

that we discretize and rewrite according to previous definitions of data sample $\mathbf{X}$ and codebook $\mathbf{Y}$, as here

$$E_{ng}(\mathbf{X}, \mathbf{Y}, \lambda) = \frac{1}{2C(\lambda)} \sum_{i=1}^{M} \sum_{j=1}^{N} H_i(\mathbf{x}, \mathbf{Y}, \lambda) \|\mathbf{x}_j - \mathbf{y}_i\|^2$$

where $H_i(\mathbf{x}, \mathbf{Y}, \lambda)$ is the neighboring function of equation . Note that we can reorganize it as follows:

$$E_{ng}(\mathbf{X},\mathbf{Y},\lambda) = \sum_{j=1}^{N}\left\|\mathbf{x}_j-\mathbf{y}_{w(\mathbf{x}_j)}\right\|^2 + \sum_{j=1}^{N}\sum_{\substack{i=1\\ i\neq w(\mathbf{x}_j)}}^{M} H_i(\mathbf{x}_j,\mathbf{Y},\lambda)\left\|\mathbf{x}_j-\mathbf{y}_i\right\|^2, \tag{8.2}$$

Where we have noted that $H_{w(\mathbf{x}_j)}(\mathbf{x}_j,\mathbf{Y},\lambda)=1$ and therefore the first term in equation (8.2) is equivalent to the first term in equation (8.1). The second term in equation (8.2) corresponds to the smoothing term in equation (8.1). For the SOM, when the neighborhood function is the crips one given in equation it is assumed that the functional minimized by the SOM is the extended distortion (although [241] proves that minima of the extended distortion are not the fixed points of the SOM rule in the general):

$$E_{SOM}(\mathbf{X},\mathbf{Y},h)=\sum_{i=1}^{M}\sum_{j=1}^{N}H_i(\mathbf{x}_j,\mathbf{Y},h)\left\|\mathbf{x}_j-\mathbf{y}_i\right\|^2 \tag{8.3}$$

$$H_i(\mathbf{x},\mathbf{Y},h)=\begin{cases}1 & |w(\mathbf{x})-i|\leq h\\ 0 & \text{otherwise}\end{cases}$$

Again it is easy to decompose the functional in a structure similar to that of equation 8.1.

$$E_{SOM}(\mathbf{X},\mathbf{Y},h) = \sum_{j=1}^{N}\left\|\mathbf{x}_j-\mathbf{y}_{w(\mathbf{x}_j)}\right\|^2 + \sum_{j=1}^{N}\sum_{\substack{i=1\\ i\neq w(\mathbf{x}_j)}}^{M} H_i(\mathbf{x}_j,\mathbf{Y},h)\left\|\mathbf{x}_j-\mathbf{y}_i\right\|^2,$$

Therefore, it seems that the SOM can also be assimilated to a GNC algorithm.

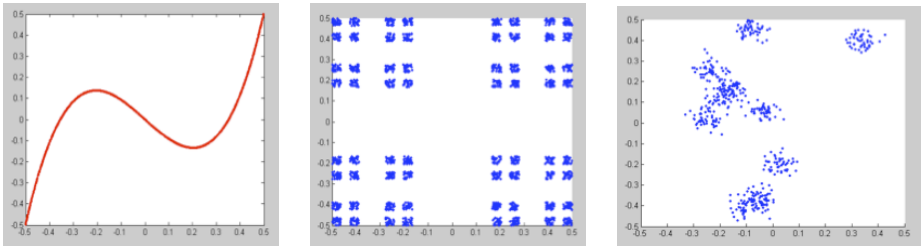
8.3.3 Convexity of SOM and NG initial functionals

The next problem is to find the conditions for convexity regarding the neighborhood parameters, so that they can be set to ensure an initial convex functional. This is a trivial task for both SOM and NG. The second derivative of the SOM functional relative to the unit $\mathbf{y}_i$ reads:

$$\nabla_i^2 E_{SOM}(\mathbf{X},\mathbf{Y},h)=\frac{1}{2}\sum_{j=1}^{N}H_i(\mathbf{x}_j,\mathbf{Y},h) \tag{8.4}$$

The condition for convexity is that all these second derivatives must be greater than zero for any given sample data:

$$\forall\mathbf{X};\forall i;\nabla_i^2 E_{SOM}(\mathbf{x},\mathbf{Y},h)>0$$



(a) (b) (c)

Figure 8.1: The three benchmark data sets (a) S-shaped, (b) Cantor set and (c) mixture of Gaussian distributions.

Setting the neighborhood radius to encompass the whole network ensures that condition. For 1D unit index topologies $h = {}^{M}/_{2}$ is a sufficient condition for convexity of the initial functional minimized by SOM. Note that if the neighborhood function is not always nonnegative (i.e.: the Mexican hat neighborhood function) this condition will work. The second derivative of NG functional relative to a unit is like equation (8.4). As the NG neighborhood function is always positive, although it can take very small values, the condition for convexity is even more general, any non zero temperature will ensure theoretical convexity of the target functional. Moreover, it guarantees that the successive functionals minimized as the temperature decreases are convex up to the limit of zero temperature, which may be a reason for the good performance of NG. This is not true for SOM. A subject for further work is the study of the minimum continuation process performed while decreasing the neighborhood.

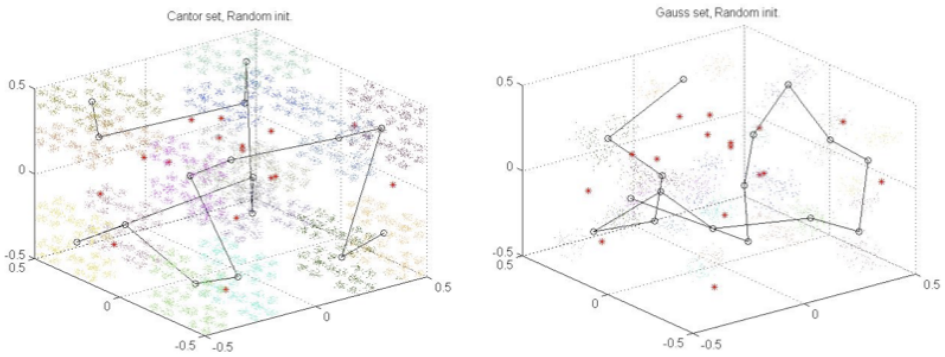
8.4 Experimental results

The experimental results presented is this section are meant to stress the idea that SOM and NG must be considered as a kind of GNC algorithms. . They show that, contrary to what is to be expected from a SGD algorithm, SOM and NG fast training sequences of the one-pass realizations performance is comparable to slow careful training sequences and Batch realizations.

8.4.1 Experimental data sets

We have done the computational experiments on three 2D benchmark data sets which have already been used in [60, 88, 54] for evaluation of clustering and VQ algorithms. They are visualized in Figure 8.1: (a) S-shaped distribution (b) Three-level Cantor distribution, (c) A mixture of Gaussian distributions.

The first distribution is constructed with 2000 data points that fall on an S-shaped curve defined by the equation $y = 8x^3 - x$, where x is uniformly distributed in the interval $[-0.5, 0.5]$. The three-level Cantor set (2048 data points) is uniformly distributed on a fractal; it is constructed by starting with



(a) (b)

Figure 8.2: The 3D benchmark data sets (a) Cantor set and (b) mixture of Gaussian distributions, with sample VQ solutions obtained by the SOM

a unit interval, removing the middle third, and then recursively repeating the procedure on the two portions of the interval that are left. And the third data set is a collection of 500 data points generated by a mixture of ten Gaussian distributions with $\overline{x} \in [-0.5, 0.5]$ and $\sigma^2 = 0.001$.

We have applied the algorithms to higher dimension data, thus the three level Cantor Set (16384 data points) and the mixture of Gaussians (2000 data points) have been extended to 3D. The Figure 8.2 shows two 3D data sets with some SOM VQ solution.

The well known Iris dataset has been also tested. The data contains four measurements for 50 individuals from each of the three northern american species of iris flowers.

We have also worked on well known AVIRIS hyperspectral image: the Indian Pines used and described, for instance in [134]. The image is a 145x145 pixel image, where each pixel has 220 bands. In Figure 8.3 display the 170th band. For the hyperspectral image the task is to find the clustering of the pixel spectra.

8.4.2 Algorithm details

The codebook initialization used in this paper is a random selection of input sample data. The codebook size is set to $c = 16$ codevectors for 2D and 3D datasets, $c = 3$ for the Iris data and $c = 17$ for the Indian Pines image. The number of presentations has been established in a maximum of 50 for conventional online and batch realizations of the algorithms. Nevertheless, we introduce a stopping criterion on the relative decrement of the distortion, the process will stop if it is not greater than $\xi = 0.001$. For SOM algorithms the neighbourhood parameter values have been: $h_0 = {}^{M}/_{2} + 1$; $h_N = 0.1$, and for NG algorithms $\lambda_0 = {}^{M}/_{2}$; $\lambda_N = 0.01$. In both one-pass version algorithms the learning rate values are $\alpha_0 = 0.5$ and $\alpha_N = 0.005$. We have executed 100 times each algorithm for 2D and 3D datasets, and 50 times for others. The algorithms tested are:

Figure 8.3: The 170th band of the hyperspectral image Indian pines.

the online conventional realizations of SOM and Neural Gas (NG), the batch versions (BSOM and BNG) and the online One Pass realizations (SOMOP and NGOP).

8.4.3 Quantitative results of the experiments

Begin by analyzing the results obtained on 2D benchmarks datasets. In Figures 8.4(a), 8.4(b), and 8.4(c) we present the mean and 0.99 confidence interval of the distortion results of the tested algorithms for each data set. We have also taken into account the computation time. In Figures 8.4(d), 8.4(e), and 8.4(f) the y-axis corresponds to the product of the final distortion and the computation time as measured by Matlab. The inspection of the figure reveals that the relative efficiencies of the algorithms measured by the final distortion, depend on the nature of the data. For example, the One Pass SOM improves the online and batch algorithms on the Cantor data set, it is similar on the Gaussian data set and falls behind in the S-shaped data set. Although this is not the main concern of this paper, the distortion results show that the Neural Gas improves the SOM most of the times, confirming the results in the literature [193]. The batch realization sometimes improves the online realization (Cantor and S-shape), sometimes not (Gaussian).

The main motivation of our work was to compare batch and One Pass realizations, from Figures 8.4(a), 8.4(b), and 8.4(c) the One Pass SOM improves the batch SOM in some cases (Gaussian and Cantor) and is almost equal in the S-shape data, when all the algorithms behave almost identically. However, the One Pass Neural Gas improves the batch realization only on the Cantor data

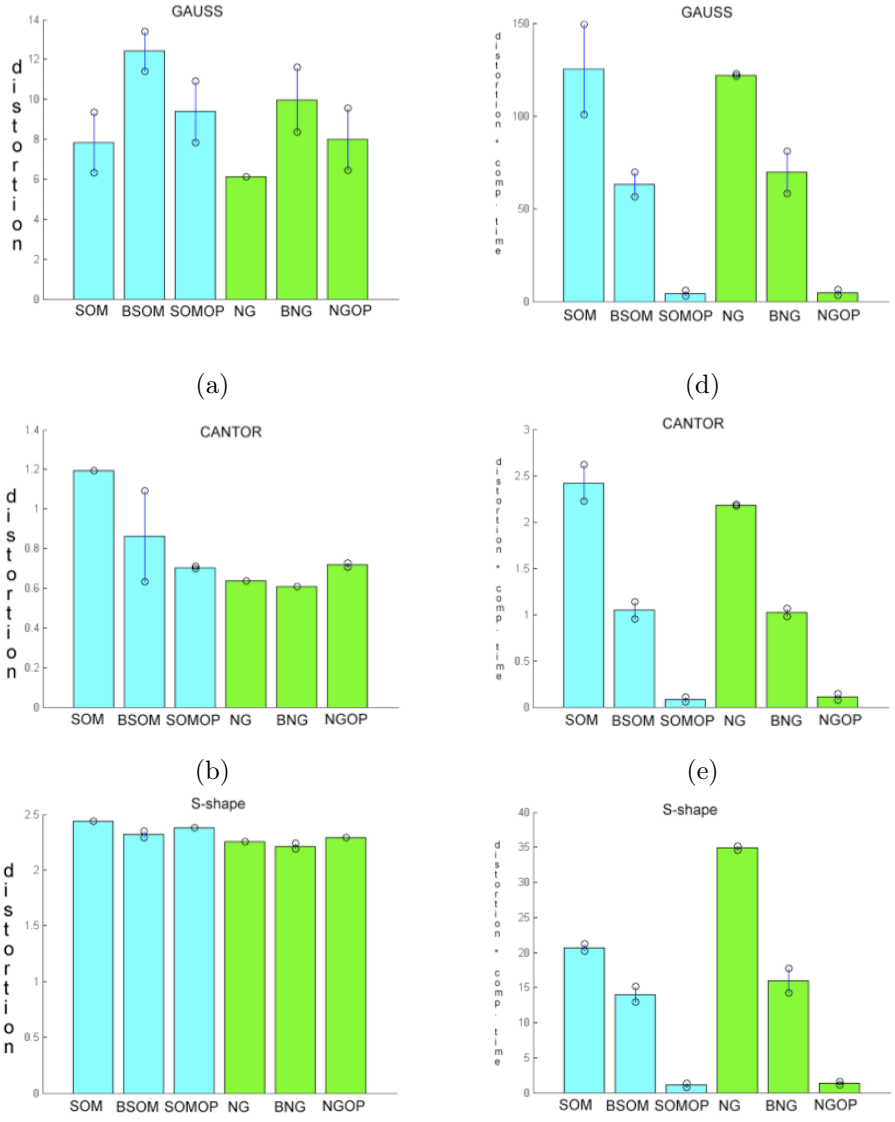
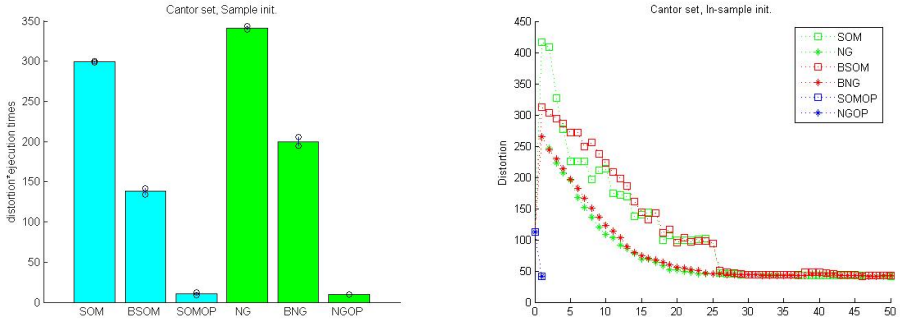


Figure 8.4: Results on the three 2D benchmark algorithms. The distortion on the (a) Gaussian data, (b) Cantor data set and (c) S-shaped data. The distortion times the computational time for (d) Gaussian data, (e) cantor data, and (f) S-shaped data.



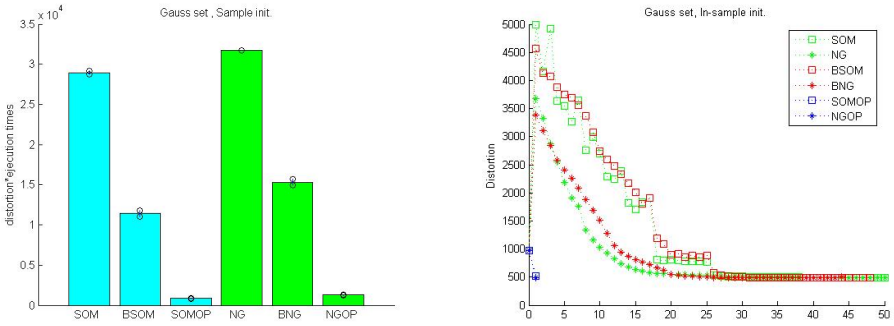
(a) (b)

Figure 8.5: 3D Cantor Set. (a) The distortion times the computational time. (b) Evolution of the quantization distortion as function of the number of input vectors used in the estimation, measured in multiples of sample size.

set. When we take into account the computation time in Figures 8.4(d), 8.4(e), and 8.4(f), the improvement of the One Pass realization over the batch and conventional realizations is spectacular. It can also be appreciated the improvement of the batch realization over the conventional online realization. The reasoning behind Figures 8.4(d), 8.4(e), and 8.4(f) is that, even when the One Pass realization give a much worse solution than the batch and online realizations, the time constraints may be so strong that a suboptimal solution is preferable than the optimal one. In this setting, the One Pass realization could be very useful.

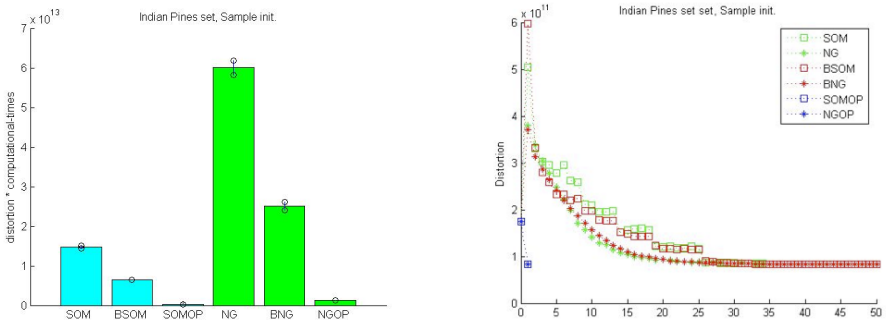
For the 3D Cantor Set, the mixture of 3D Gaussians, the Indian Pines image and the 4D Iris dataset, Figures 8.5(a), 8.6(a), 8.7(a), and 8.8(a), respectively, display the product of the final distortion and the computation time as measured by Matlab. Figures 8.5(b), 8.6(b), 8.7(b), and 8.8(b), a sample trajectory of the error during the training process. Some common pattern of all these plots are: (a) NG and BNG have a much more smooth decrease than SOM and BSOM, however they reach comparable results. (b) The one-pass instances perform a very fast decrease to values comparable to the other realizations. A pure SGD algorithm without any GNC property could not obtain the results in the Figures 8.5, 8.6, 8.7, and 8.8. These results are repeating in all the experiments we are performing up to date.

In Tables 8.1 and 8.2 we present the results for the 50 repetitions of the algorithms on the 4D Iris Dataset using random initializations. We note the shocking result of BSOM that may be due to its inability to produce a proper topological organization of the units or to its high sensitivity to the initial conditions. Table 8.1 presents the average, standard deviation and width of the 99% confidence interval of the quantization distortion at the end of the training. The one-pass realizations give results comparable to the conventional online algorithms, which improve over the batch realizations. The results show very small variance, therefore the realizations are very insensitive to initial conditions



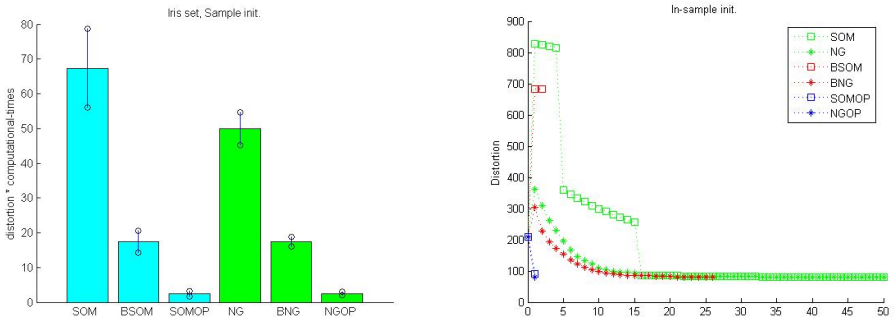
(a) (b)

Figure 8.6: 3D Mixture of Gaussians. (a) The distortion times the computational time. (b) Evolution of the quantization distortion as function of the number of input vectors used in the estimation, measured in multiples of sample size.



(a) (b)

Figure 8.7: Indian Pines hyperspectral image. (a) The distortion times the computational time. (b) Evolution of the quantization distortion as function of the number of input vectors used in the estimation, measured in multiples of sample size.



(a) (b)

Figure 8.8: Iris dataset. (a) The distortion times the computational time. (b) Evolution of the quantization distortion as function of the number of input vectors used in the estimation, measured in multiples of sample size.

	Distortion		
	mean	Std. Dev.	99 CI
SOM	80.10	4.306e-014	1.569e-014
BSOM	681.91	4.594e-013	1.673e-013
SOMOP	79.91	4.002e-005	1.458e-005
NG	80.10	0	0
BNG	82.35	1.005e-013	3.660e-013
NGOP	81.38	4.8667e-010	1.773e-010

Table 8.1: Distortion results obtained for the 4D Iris dataset.

like GNC algorithms.

In Table 8.2 we present the results in terms of distortion multiplied by computational time (measured in minutes). In this measure the computational time acts as an amplifier of the distortion results. We prefer it to others (e.g. the ratio distortion/time) because it preserves the optimality sense of the distortion (the lower the better). In this table we find that the one-pass realizations perform almost an order of magnitude better than conventional online and batch realizations.

In Tables 8.3 and 8.4 we present the results for 20 repetitions of the algorithms applied to the the Indian Pines hyperspectral image. Again, over this data set the one-pass realizations perform equally well than the conventional online realizations and the batch realizations. Batch realizations show an extremely high variance, due to their sensitivity to initial conditions. The online realizations, on the other hand, are almost insensitive to initial conditions. When taking the computational time into account the one-pass realizations per-

	Distortion*Comp.time		
	mean	Std. Dev.	99 CI
SOM	11.39	3.992e-014	1.454e-014
BSOM	19.18	11.592e-013	4.223e-013
SOMOP	1.89	0.882e-005	0.321e-005
NG	10.94	0	0
BNG	9.01	3.695e-013	1.346e-014
NGOP	2.72	1.259e-010	0.459e-010

Table 8.2: Distortion times computational time for the 4D Iris dataset.

	Distortion		
	mean	Std. Dev.	99 CI
SOM	8.256e+10	1.565e-05	9.017e-06
BSOM	8.274e+10	3.346e+08	1.927e+08
SOMOP	8.270e+10	2.475e-05	1.425e-05
NG	8.256e+10	1.750e-05	1.008e-05
BNG	8.273e+10	4.501e+08	2.592e+08
NGOP	8.250e+10	1.819e-05	1.048e-05

Table 8.3: Distortion results obtained for the Indian Pine hyperspectral image.

	Distortion*Comp.time		
	mean	Std. Dev.	99 CI
SOM	1.491e+13	1.333e-05	7.680e-06
BSOM	3.940e+12	3.086e+08	1.777e+08
SOMOP	2.453e+11	1.036e-05	5.966e-05
NG	5.402e+13	9.792e-05	5.640e-05
BNG	1.998e+13	5.921e+08	3.410e+08
NGOP	1.229e+12	5.379e-05	3.098e-05

Table 8.4: Distortion times computational times for the Indian Pine hyperspectral image.

formance is an order of magnitude better than the other realizations.

8.5 Conclusions

The paradoxical empirical results reported here showing that the one-pass realization of the SOM and NG can obtain competitive performance in terms of distortion, and much better than the "conventional" batch and online realizations in terms of computational efficiency (time x distortion) leads us to the idea that these algorithm's performance is more sensitive to the neighborhood parameters than to the learning gain parameter. Neighborhood parameters can be seen as the parameter of a family of functional whose limit is the quantization distortion. Therefore, training of the SOM and the NG can be seen not as a straight minimization of an energy function but as a continuation of the minimization process over a sequence of functionals tuned by the neighborhood control parameter. If the starting functional is convex the algorithms can be considered as Graduated Nonconvextity (GNC) algorithms. Indeed we have shown that the NG and SOM energy functionals can be easily seen as the canonical GNC functionals [202, 204, 203, 39]. Also, it ease to find sufficient conditions on the neighborhood radius for the initial functional to be convex. Moreover, it is ease to verify that in fact the NG functionals are convex up to limit zero temperature, which explains its smooth behavior.

Part III

Proposals for architectures

Chapter 9

Local Stochastic Learning Rule

We have worked [122, 121] on studying the effect of substituting the deterministic encoding process by an stochastic counterpart in the competitive learning. A stochastic approximation to the nearest neighbour (NN) classification rule is proposed for Vector Quantization (VQ). This approximation is called Local Stochastic Competition (LSC). Some convergence properties of LSC are discussed, and experimental results are presented. The approach shows a great potential for speeding up the codification process, with an affordable loss of codification quality. Moreover, when we apply this approach to a competitive learning process, such as SOM, we obtain a Local Stochastic Learning Rule (LSLR) which is applied to compute the codebook for image vector quantization (VQ).

9.1 Local Stochastic Competition

Following work started in [122, 121], we propose here a Local Stochastic Competition (LSC) decision rule for the enconding phase of VQ. The LSC rule is intended as a distributed stochastic approximation to the Nearest Neighbour (NN) rule usually applied in VQ to perform the mapping of the input vector into the codebook. We have found in the literature [273, 249, 64] some attempts to speedup the computation of the NN decision rule employed in the VQ codification phase. These approaches used the fact that quite frequently the decision that a codevector is not the nearest neighbour of an input vector can be taken without fulfilling the computation of the Euclidean distance. They didn't involve any loss of accuracy and didn't propose any kind of computational distributed scheme. The LSC is a significative departure from that, because it is a stochastic approximation that involves loss of classification accuracy, thought it has a potential for greater speedups given fully distributed implementations.

The LSC is related to Radial Basis Function (RBF) neural network archi-

tectures [139, 35, 212, 287, 274] applied to function approximation or interpolation. The function domain is decomposed into a set of overlapping regions, each characterised by a kernel function whose parameters usually are the centroid and width of the region. The most usual kernel functions are Gaussian. From a Bayesian classification point of view [76], the use of Gaussian kernels can be interpreted as the approximation of the input distribution by a mixture of Gaussian distributions, each characterised by its mean and variance parameters. Each Gaussian distribution models the probability that a given input belongs to a class. Our approach assumes this kind of probabilistic framework. We assume that each codevector represents a separate class, being the mean of the Gaussian distribution. We assume that variance parameters can be estimated, either by the codebook design algorithm, or from the codebook itself. In the experiments reported below, the later has been assumed. Under these assumptions, LSC, then, consists in the parallel sampling of the *a posteriori* probabilities of the codevector classes, taken as independent one-class problems. Note that, in the same framework, NN corresponds to the optimal Bayesian decision rule when the class variances are identical.

Given an input vector to be encoded $\mathbf{x}_n$ and a codebook $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_M\}$, the LSC proceeds by sampling independently for each codevector $\mathbf{y}_i$ a Bernouilli distributed random variable of parameters $(p_n(i), 1 - p_n(i))$:

$$p_n(i) = P(\mathbf{x}_n \in R_i) = e^{-\frac{\|\mathbf{x}_n - \mathbf{y}_i\|}{\sigma_i}},$$

where R_i corresponds to the input space region corresponding to $\mathbf{y}_i$, σ_i is the standard deviation associated with codevector $\mathbf{y}_i$. That is, each $p_n(i)$ is interpreted as the probability that $\mathbf{x}_n$ belongs to the class represented by $\mathbf{y}_i$, taken as an independent one-class classification problem for each codevector. If the sampling is positive for more than one codebook, the tie-breaking is a random selection. Sampling is repeated until at least one codevector is selected. An algorithmic description of the sequential execution of the LSC classification of an input vector $\mathbf{x}_n$, as done for our experiments is given in Algorithm 9.1. Function $f(k)$ is and increasing function to ensure that the input vector is classified eventually. We use $f(k) = 2^{k-1}$ for fast convergence. This process could be easily implemented in parallel, running a separate processor for each codevector in the codebook. An algorithm of the process associated with each codevector could read as Algorithm 9.2. The **wait** operation means that the process is idle waiting for either an input vector or the signal to try again the classification. The **signal** operation outputs the positive or negative response. A dedicated process solves ties and decides retrying classification. The expected speedup of the parallel implementation of the codification process comes from the substitution of the sequential search through the codebook by the parallel test of the separate one-class probabilities.

Algorithm 9.1 LSC sequential execution

1. Step_0: $k = 1$
2. Step_1: Built up the probability vector $\mathbf{p} = (p_n(i,k); i = 1, ..., M)$ computed as follows:
$$p_n(i,k) = e^{-\frac{\|\mathbf{x}_n - \mathbf{y}_i\|}{t_i(k)}},$$
with $t_i(k) = f(k)\sigma_i$.
3. Step_2: Sample the probabilities in $\mathbf{p}$: Built up the set
$$S_k = \{\mathbf{y}_i \in \mathbf{Y} \,|p_n(i,k) \geq u_i\},$$
where $(u_1, \ldots, u_M)$ are random numbers uniformly distributed in $[0, 1)$.
4. Step_3: If $|S_k| = 0$ then increase k by 1 and goto step-1.
5. Step_4: If $|S_k| > 1$ perform a random selection with equal probabilities in the set S_k. If codevector $\mathbf{y}_i$ is chosen then the codification is: $C_{LSC}(\mathbf{x}_n, \mathbf{Y}) = i$

Algorithm 9.2 LSC independent process of each codevector

wait for input $\mathbf{x}_n$ or *retry*

case new input $k = 1$

case reintent $k = k + 1$

Compute
$$p_n(i,k) = e^{-\frac{\|\mathbf{x}_n - \mathbf{y}_i\|}{t_i(k)}}.$$

Generate a random number u. If $p_n(i,k) > u$ **signal** 1, if not **signal** 0.

9.1.1 Convergence

To ensure that LSC will converge, eventually giving some response, function $f(k)$ must be monotonically increasing with k. The faster the increase, the shorter the response time. Mathematically, LSC generates, for a given $\mathbf{x}_n$, a random sequence of sets $\{S_1, \ldots, S_K\}$ with $|S_k| = 0$ for $k < K$, and $|S_K| = 0$. The stopping condition for this process is, therefore, to find a non-empty set of positive responses. It is easy to verify that the probability of finding a non-empty set increases as the process goes on. The probability of finding an empty set at trial k is

$$P\left(|S_k| = 0 \,|\mathbf{x}_n, \mathbf{Y}\right) = \prod_{i=1}^{M} \left(1 - p_n\left(i,k\right)\right) = \prod_{i=1}^{M} \left(1 - e^{-\frac{\|\mathbf{x}_n - \mathbf{y}_i\|}{t_i(k)}}\right).$$

Given that $f(k)$ is increasing:

$$\lim_{k\to\infty} e^{-\frac{\|\mathbf{x}_n - \mathbf{y}_i\|}{t_i(k)}} = 1,$$

therefore,

$$P\left(|S_k| = 0 \,|\mathbf{x}_n, \mathbf{Y}\right) = 0,$$

and finally:

$$P\left(|S_k| > 0 \,|\mathbf{x}_n, \mathbf{Y}\right) = 1.$$

The increasing nature of $f(k)$ is of great relevance, both theoretical and practical. In our experiments we have chosen the exponential expression $f\left(k\right) = 2^{k-1}$, because of the emphasis we put in speeding up the classification process. The fast increase of the variance term has the side effect of increasing the probability of bad classifications [122].

The last topic that remains to be discussed is the estimation of the variance parameters. In the image VQ experiments we have estimated these variance parameters from the codebook itself as follows. Let D_i denote the minimum distance from codevector $\mathbf{y}_i$ to any other codevector:

$$D_i = \min\left\{\|\mathbf{y}_i - \mathbf{y}_j\|\,; j = 1, ..., M; j \neq i\right\}.$$

We compute the mean minimum distance between codevectors

$$\overline{D} = \frac{1}{M}\sum_{i=1}^{M} D_i.$$

The estimate of the standard deviation associated with codevector $\mathbf{y}_i$ is, then, computed as follows:

$$\hat{\sigma}_i = \begin{cases} \frac{D_i}{2d} & D_i < \overline{D} \\ \\ \frac{\overline{D}}{2d} & D_i \geq \overline{D} \end{cases}.$$

Figure 9.1: Original grayscale image for the experiments

9.2 Results of LSC on image VQ

We apply LSC to image VQ based on a given codebook. A gray level image is a matrix of pixels, each pixel taking values in the discrete set $\{0, 1, \ldots, 255\}$. A row-wise vector decomposition of dimension d of the image is a succession of vectors $\mathbf{X} = \{\mathbf{x}_1, .., \mathbf{x}_N\}$, each $\mathbf{x}_i$ composed of d row-adjacent pixels. Consecutive vectors correspond to non overlapping sequences of pixel indices. Column-wise and matrix-wise decompositions can be defined as well. A VQ of the image is a map from each $\mathbf{x}_i$ into a natural number, that transforms the vector decomposition into a sequence of codes $\{c_1, .., c_N\}$, each code corresponding to the index of a codevector in the given codebook $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_M\}$. The compression rate obtained by VQ depends on the number of different codes and the dimension d of the image vector decomposition. In this experiment we use 256 codes, so that the compression rate will always be $d : 1$.

We have performed a set of experiments of codification/decodification on the to the vector decomposition of the image shown in Figure 9.1. The codebook was obtained applying the threshold algorithm to obtain the initial codebook, and SCL to improve over it. The experiment parameters were the dimension of the vector decomposition d, and the threshold parameter θ. The quality

		NN		LSC		
θ	d	SNR	δ	SNR	δ	s
8	8	27.0	466	23.4	1012	70
	16	24.3	1328	20.5	3532	69
	32	22.6	1878	19.1	5851	68
	64	22.3	1655	19.3	8063	72
32	8	26.4	373	23.3	1217	79
	16	23.4	822	19.0	4281	74
	32	22.2	1196	18.0	8251	77
	64	24.9	1057	20.6	4530	78

Table 9.1: Enconding results for NN and LSC coding for varying d and θ ($M = 256$).

	NN		LSC		
M	SNR	δ	SNR	δ	s
128	25.4	791	22.0	1547	34
256	27.0	466	23.4	1028	71
512	28.3	208	24.6	602	144
1024	30.8	50	26.6	234	304

Table 9.2: Encoding results for NN and LSC for increasing number of codevectors M ($d = 8$ and $\theta = 8$).

Figure 9.2: NN codification with a $M = 1024$ codebook.

Figure 9.3: LSC codification with a $M = 1024$ codebook.

measures computed are the distortion (δ) and signal-to-noise ratio (SNR). The expected speedup (s) of LSC over NN is computed as the number of codevectors divided by the mean number of trials that LSC performs until giving a classification response. Tables 9.1 and 9.2 show the numerical results. Table 9.1 shows the results of NN and LSC codification with $M = 256$ codevectors of varying dimension $(8, 16, 32, 64)$ obtained by application of SCL to the result of the threshold algorithm with threshold parameters $\theta = 8$ and $\theta = 32$. Increasing d gives greater compression ratio. The variation of θ was intended to give different initial conditions for SCL.

Table 9.2 shows the results of increasing number of codevectors M. The codebooks are also obtained by application of the threshold and SCL algorithms. Figure 9.2 shows the decodification of the test image after NN codification using a codebook with 1024 codevectors of dimension 8. Figure 9.3 shows the decodification of the LSC codification with the same codebook. This precise codebook was chosen because it is the one that has the greater expected speedup.

From the data in both tables, it can be perceived an almost constant degradation of the LSC codification relative to NN. The observation of the images in Figure 9.2 and 9.3 shows that this degradation can be acceptable, for applications without severe quality requirements. On the hand, table 9.2 shows how the expected speedup increases with the number of codevectors, which makes LSC a suitable alternative for applications with large codebooks.

9.3 Local Stochastic Learning Rule

We proposed the Local Stochastic Learning Rule (LSLR) as a variation of the Soft-Competition Scheme (SCS) [96] which is a variation of the SCL and SOM where updating the representatives is weighted by the estimations of the *a posteriori* probabilities $P_k(i)$ of vector $\mathbf{x}_k$ belonging to the class of codevector $\mathbf{y}_i$.

It is defined as follows:

$$\triangle y_i\left(k\right) = \alpha_i\left(k\right) P_k\left(i\right)\left(\mathbf{x}_k - \mathbf{y}_i\left(k\right)\right)$$

$$P_k\left(i\right) = \frac{e^{-\|\mathbf{x}_k - \mathbf{y}_i(k)\|^2 \sigma_i^{-2}(k)}}{\sum_{j=1}^{M} e^{-\|\mathbf{x}_k - \mathbf{y}_j(k)\|^2 \sigma_j^{-2}(k)}}$$

The neighborhoods depend on the input vector and all the codevectors are adapted in each presentation of an input vector. This rule can be derived as a stochastic gradient descent of the cross-entropy or Kullbak distance between the input distribution and the distribution modelled by the codevectors, which is assumed as a mixture of gaussians whose means are the codevectors. Derivation over the variances of these gaussians (assumed $\sigma_i I$ covariance matrices), gives the stochastic gradient descent rule on the variances:

$$\triangle\sigma_i\left(k\right) = \alpha_i\left(k\right) P_k\left(i\right) \frac{\left\|\mathbf{x}_k - \mathbf{y}_i\left(k\right)\right\|^2 - d\sigma_i^2\left(k\right)}{\sigma_i^3\left(k\right)}$$

(remember that d was the dimension of the vector decomposition). Our Local Stochastic Learning Rule (LSLR) is derived from the above rules simply by taking as "a posteriori" probabilities the unnormalized exponential of the distance. The LSLR rules read:

$$\triangle y_i(k) = \alpha_i(k)\, p_k(i)\, (\mathbf{x}_k - \mathbf{y}_i(k))$$

$$\triangle \sigma_i(k) = \alpha_i(k)\, p_k(i)\, \frac{\|\mathbf{x}_k - \mathbf{y}_i(k)\|^2 - d\sigma_i^2(k)}{\sigma_i^3(k)}$$

where

$$i \in C_{LSC}(\mathbf{x}_k, \mathbf{Y}(k))$$

$$p_k(i) = e^{-\|\mathbf{x}_k - \mathbf{y}_i(k)\|^2 \sigma_i^{-2}(k)}$$

The adaptation is applied to the codevectors selected by LSC [2, 11], denoted C_{LSC} in the above formulas. Taking the unnormalized $p_k(i)$ as the estimates of the "a posteriori" probabilities, introduces numerical instabilities, and the loss of the formal justification of the rules. However, it allows fully distributed computation, because the rules can be applied in parallel to all the codevectors. In the experiments reported here, we have performed a scheduling of the gain parameter identical to the one performed for SCL. Figure 4 shows the image after codification /decodification with the codebook obtained by LSLR, starting from the initial codebook that produces the image in Figure 2. From the comparison of Figures 3 and 4, it can be concluded that the loss of quality of LSLR is acceptable, and that it definitely improves over the initial codebook. More extensive numerical results are given in the next section.

9.4 Results of LSLR on VQ of images

We have performed a set of experiments of codebook computation on the image in Figure 9.1, applying the threshold algorithm to obtain the initial codebooks, and SCL and LSLR to improve over them. Both, SCL and LSLR, were applied only once to the vector decomposition of the image. The experiment parameters were the dimension of the vector decomposition d, and the threshold parameter θ. The quality measures computed are the distortion (δ) and signal-to-noise ratio (SNR). The expected speedup (s) of LSLR over SCL is computed as the number of codevectors divided by the mean number of trials that LSC performs to select a non-empty set of codevectors to be updated by LSLR rules [128]. Table 9.3 shows the numerical results. From the data in Table 9.3 we may conclude that LSLR generally improves over the initial codebook, but less than SCL. The SNR seems to be a more faithful measure of the quantification quality. LSLR produces a steady increase of the SNR, without reducing significantly the distortion. The examination of both table 9.3 and Figure 9.4 leads to the conclusion that LSLR is a good stochastic approximation to SCL. From the

		INITIAL		SCL		LSLR		
θ	d	SNR	δ	SNR	δ	SNR	δ	s
8	8	24.2	789	27.0	466	25.5	693	90
	16	21.8	2296	24.3	1328	22.8	2219	80
	32	19.8	3801	22.6	1878	21.0	3796	74
	64	19.1	3712	22.3	1655	21.1	3735	74
32	8	22.1	613	26.4	373	25.0	527	118
	16	20.8	1395	23.4	822	22.7	1295	106
	32	20.4	1965	22.2	1196	22.2	1813	97
	64	23.2	1513	24.9	1057	24.7	1290	95

Table 9.3: Comparative results of SCL and LSLR for varying threshold parameter θ and image vector decomposition dimension d.

point of view of the performance, LSLR shows a great potential for speeding up the codebook computation, provided a fully parallel implementation. It can be expected that this speedup would increase with the number of codebooks, making it useful for practical applications, where large codebooks may happen.

Figure 9.4: Image after codification/decodification with the codebook obtained with LSLR.

Chapter 10

An Evolution Strategy for near real-time Color Quantization of image sequences

Color Quantization of image sequences is a case of Non-stationary Clustering problem. The approach we adopt to deal with this kind of problems is to propose adaptive algorithms to compute the cluster representatives. In previous chapters we have studied the application of Competitive Neural Networks to the one-pass adaptive solution of this problem. One-pass adaptation is imposed by the near real-time constraint that we try to achieve. In this chapter we define an Evolution Strategy for this task . Experimental results show that the proposed Evolution Strategy can produce results comparable to that of Competitive Neural Networks.

The chapter is organized as follows. Section 10.1 gives an introduction. Section 10.2 recalls the adaptive approach to Non-stationary Clustering/VQ. Section 10.3 presents the Evolution Strategy for Non-stationary Vector Quantization. Section 10.4 presents the experimental results. Section 10.5 gives some conclusions on the application of Evolution Strategy to NSC.

10.1 Introduction

Evolution Strategies [19, 20, 197]have been developed since the sixties. They belong to the broad class of algorithms inspired by natural selection. The features most widely accepted as characteristic of Evolution Strategies are: (1) vector real valued individuals, (2) the main genetic operator is mutation, (3) individuals contain local information for mutation so that adaptive strategies can be formulated to self-regulate the mutation operator. However, many hy-

brid algorithms can be defined, [197] so that it is generally difficult to assign a definitive "label" for a particular algorithm. Nevertheless, we classify the algorithm proposed here as an Evolution Strategy because it fits better in the above characterization than in the accepted characterizations of Genetic Algorithm or Genetic Programming.

Non-stationary Clustering problems assume a time varying population sampled at discrete times. Therefore the Clustering of the data must be recomputed at each time instant. A related problem is that of Adaptive Vector Quantization [97]. The works reported here belong to the class of shifting-mean Adaptive Vector Quantization [99]. Both the Evolution Strategy and the Competitive Neural Networks are applied as adaptive algorithms for the computation of the cluster representatives given by the cluster means at each time instant. Our work fits in the original proposition of Holland [146] of evolution as a mechanism for adaptation to uncertain and varying environment conditions

Evolution Strategies fall in the broad family of stochastic algorithms. These algorithms are characterized by slow convergence and large computation times. As we are trying to apply them in a near real-time framework, we impose two computational restrictions in their application:

1. One-pass over the sample adaptation and

2. we use subsamples of the data to estimate the cluster representatives for the whole data set at each time instant.

3. The adaptation must be performed in one generation. Therefore there is no possibility of performing a process that may resemble a global random search at each adaptation time instant.

4. The computations must be based on subsamples of the data available at each time instant. The available data is a frame in the sequence, a sample of size much lesser than that of image will be used to reduce the computation.

For the Evolution Strategy, restriction (1) implies that only one generation is computed at each time instant. For the Competitive Neural Networks, this restriction implies that the sample data is presented only once, therefore the learning parameters must have a very fast decrease.

Color Quantization [141, 207, 185, 272] is an instance of Vector Quantization (VQ) [97] in the space of colors. Color Quantization has application in visualization, color image segmentation, data compression and image retrieval [156]. In this work we do not deal with the problem of finding the natural number of colors. This is a more involved problem than looking for a fixed number of color representatives, and some of the results discussed in Section 10.4 recommend that it must approached cautiously, and after being satisfied with the results of quantization to a fixed number of colors. In summary, Color Quantization to a fixed number of colors of image sequences [101] is a case of Non-stationary Clustering, dealt with performing Adaptive Vector Quantization.

The specific features of the Evolution-based Adaptive Strategy analysed in this chapter are:

1. It is intended to deal with a time varying environment. Therefore the fitness function will be time dependent.

2. Individuals are defined as components of the solution instead of representing complete solutions. This implies the existence of a global fitness function for the entire population, on top of the individual fitness functions. The algorithm is expected to produce as a cooperative solution the best population to solve the problem. In this respect, our approach resembles largely the so-called Michigan approach to the design of Classifier Systems.

3. Due to the problem representation chosen, the selection looks for entire populations, so that there is a strong interaction between parents and children in the evaluation of the selection operator.

4. Mutation is the only operator that introduces evolution-like variability. The parameters guiding the mutation, which can be referred as self-adaptation parameters, are deduced from the interaction with the environment.

5. In our final proposition mutation is performed deterministically to approach real time response. However, even in this extreme setting the Evolution-based Adaptive Strategy that we will propose below remains a stochastic algorithm whose source of stochasticity is the input data

6. The population of the proposed algorithm is the set of cluster representatives, each individual is a color representative in the $[0, 1]^3$ real subspace that defines a cluster in the color space of the image pixels via the Voronoi tesselation induced by the whole population.

7. The main evolutive operator is mutation.

8. The mutation operator is based on the local cluster covariance matrices, which guide and regulate its realization. We have not used recombination operators.

The assumption of a time varying environment must not be confused with the case of a noisy environment [81, 4]. From our point of view the latter is a particular case of the former. The uncertainty associated to the environment is perceived by the algorithm through the variability of the fitness function. In the case of noisy environment as formulated in [81], each evaluation of the fitness function involves a sampling procedure. This sampling procedure assumes a stationary random process as the source of the fitness function values. On the other hand, our approach assumes that the fitness function value will vary due to the inherent Non-stationarity of the environment. The fitness is measured upon a sample of the process. This sample is considered as representative of the environment for a limited amount of time. While the data sample remains the

same, the fitness is a deterministic function. As far as the environment remains stationary, succesive data samples will posses the same statistical characteristics and the fitness function will continue to be (almost) the same. Unpredictable changes in the environment produce significant changes on the statistics of the data sample and, therefore, changes in the landscape of the fitness function. The Evolution-based Adaptive Strategy tries to adapt as fast and smoothly as possible to the environment changes in an unsupervised way.

10.2 Non-Stationary Clustering/VQ

Cluster analysis and Vector Quantization are useful techniques in many engineering and scientific disciplines [97, 138, 73, 76, 148, 89]. In their most usual formulation it is assumed that the data is a sample of an stationary stochastic process, whose statistical characteristics will not change in time. Non-stationary Clustering and Vector Quantization assume a time varying population sampled at diverse time instants that can be modelled by a discrete time non-stationary stochastic process . If a model is known (or assumed), a predictive approach [97] would reduce the problem to a stationary one. The general formulation of the Non-stationary Clustering problem does not assume any model of the process.

A working definition of the Non-stationary Clustering problem could read as follows: Given a sequence of samples $\mathbf{X}(t) = \{\mathbf{x}_1(t), \ldots, \mathbf{x}_n(t)\}; t = 0, 1, \ldots$ obtain a corresponding sequence of partitions of each sample $P(\mathbf{X}(t)) = \{\mathcal{X}_1(t), \ldots, \mathcal{X}_1(t)\}$ given by a sequence of sets of disjoint clusters minimizing a criterium function $C = \sum_{t\geq 0} C(t)$. The Non-stationary Vector Quantization design problem can be stated as the search for a sequence of representatives $\mathbf{Y}(t) = \{\mathbf{y}_1(t), \ldots, \mathbf{y}_c(t)\}$ that minimizes the error function, aka reconstruction distortion, $E = \sum_{t\geq 0} E(t)$. The squared Euclidean distance is the dissimilarity measure most widely used to formulate the criterium/error functions. The Non-stationary Clustering/VQ problem can be stated as an stochastic minimization problem:

$$\min_{\{\mathbf{Y}(t)\}} \sum_{t\geq 0} \sum_{j=1}^{n} \sum_{i=1}^{c} \|\mathbf{x}_j(t) - \mathbf{y}_i(t)\|^2 \delta_{ij}(t),$$

$$\delta_{ij}(t) = \begin{cases} 1 & i = \arg\min_{k=1,\ldots,c} \left\{\|\mathbf{x}_j(t) - \mathbf{y}_k(t)\|^2\right\} \\ 0 & \text{otherwise} \end{cases}.$$

The proposition of adaptive algorithms to solve this stochastic minimization problem is based in two simplifying assumptions:

- The minimization of the sequence of time dependent error function can be done independently at each time step.
- Smooth (bounded) variation of optimal set of representatives at successive time steps. Then the set of representatives obtained after adaptation in a time step can be used as the initial conditions for the next time step.

The adaptive application of Evolution Strategies to Non-stationary Clustering/VQ is done as follows:

1. At time t the initial population is given by the set of representatives/ codevectors computed from the sample of the process at time $t-1$.

2. A series of generations are computed starting from this initial population to compute the representatives for the clusters of the sample at time t.

3. The fitness function is related to the distortion of the representatives, coded somehow in the population, relative to the sample at time t.

4. This process is repeated for the sample at time $t+1$, and thereafter.

A distinctive feature of our proposed Evolution Strategy is that only one generation is computed to perform the adaptive step. In practice, we have extracted a subsample of each image in the sequence as the data samples for both the Evolution Strategy and the Competitive Neural Networks.

10.2.1 Clustering with ES

A representative sample of the works found in the literature dealing with clustering problems via Evolutionary Computation is [8, 15, 18, 29, 27, 32, 41, 47, 151, 163, 188, 189, 198]. The common approach of all these works is the mapping of complete clustering solutions to population individuals. The fitness function is the ad-hoc clustering criterion function. The authors propose a wide variety of representation of clustering solutions as population individuals, ranging from the set of cluster representatives to the membership (hard or fuzzy) matrices of the clusters. Evolution operators, recombination and mutation, are defined suitably to be closed operators on the representation chosen.

Our conclusion from the literature review, is that most of the Evolutionary approaches suggested for clustering could not be applied to the non-stationary case in a stringent time frame. They can not guarantee a reasonable response in a reasonable time. Most of the approaches found in the literature have a big uncertainty about the proper setting of the algorithm parameters (population size, mutation and crossover rate, the appropriate operators,...). Assuming that the previous criticisms could be properly answered, the computational complexity of each generation is usually very big , so that even in the case that the evolutionary approach is used with a computational limit imposed, this limit will be necessarily very high for practical applications of the kind we are interested in. We have honestly tried to address the problem in a way that is both computationally effective and gives good solutions, assuming its suboptimality.

10.3 The Evolution Strategy for NSC/VQ

A pseudocode representation of the general structure of the algorithm of Evolution Strategies is presented in Algorithm 10.1 [20]:

Algorithm 10.1 Evolution Strategy pseudocode

1. t:= 0
2. initialize $P(t)$
3. evaluate $P(t)$
4. while not terminate do
   (a) $P'(t)$:= recombine $P(t)$
   (b) $P''(t)$:= mutate $P'(t)$
   (c) evaluate $P''(t)$
   (d) $P(t+1)$:= select $(P''(t) \cup Q)$
   (e) t:= $t+1$
5. end while

We have defined each individual as a single cluster center, so that the entire population gives a single solution to the VQ of a given sample. In the case of NSC/VQ the sample is time varying, in fact the generation number is the discrete time of sample capture. That is, t is the data sampling time as well as the Evolution Strategy generation counter. The population at generation t is denoted $P(t) = \{\mathbf{y}_i(t); i = 1, \ldots, c\}$.

10.3.1 Fitness function

The local fitness of each individual is, its local distortion relative to the sample considered in this generation:

$$F_i(t) = \sum_{j=1}^{n} \|\mathbf{x}_j(t) - \mathbf{y}_i(t)\|^2 \delta_{ij}(t).$$

The fitness of the population as a whole can be evaluated as

$$F(t) = \sum_{i=1}^{c} F_i(t),$$

corresponding to the objective function to be minimized. Our population fitness corresponds to the within cluster scatter S_W of the clustering specified by the population. The well known equation relating the within cluster and between cluster scattering,

$$S = S_W + S_B,$$

can be related to the above Evolution Strategy fitness functions as:

$$S(t) = \sum_{j=1}^{n} \|\mathbf{x}_j(t) - \bar{\mathbf{y}}(t)\|^2 = \sum_{i=1}^{c} F_i(t) + \sum_{i=1}^{c} \|\mathbf{y}_i(t) - \bar{\mathbf{y}}(t)\|^2,$$

where $S(t)$ remains constant as far as the same data sample is considered, and $\bar{\mathbf{y}}(t)$ denotes the centroid of the entire data sample considered at time t: $\mathbf{X}(t) = \{\mathbf{x}_1(t), \ldots, \mathbf{x}_n(t)\}$. We expect that the Evolution Strategy will implicitly react through the above equation balancing the minimization of the population fitness, from the local optimization of individual cluster representatives, and the maximization of the between cluster scattering. This justifies our working hypothesis that the local optimization of individual cluster distortions will eventually lead to the global optimization of the entire set of cluster centres.

10.3.2 Mutation operator

The mutation operator is a random perturbation that follows a normal distribution of zero mean and whose covariance matrix is estimated from the data in the cluster associated with the individual to be mutated. There are three algorithm design questions to answer at this point:

1. Which individuals will be mutated?
2. How many mutations will be allowed? and,
3. What information will be used to compute mutations?.

Our proposed Evolution Strategy performs a guided selection of the individuals subjected to mutation. The set of mutated parents is composed of the individuals whose local distortion is greater than the mean of the local distortions in its generation. More formally:

$$M(t) = \left\{ i \,\middle|\, F_i(t) \geq \bar{F}(t) \right\}.$$

Regarding the number of mutations, the algorithm approaches as much as possible to a fixed number of mutations m, so that the number of mutations per individual $m_i(t)$ will depend on the size of $M(t)$, that is: $m_i(t) = \lceil m/M(t) \rceil$. Regarding the information used to generate mutated individuals, we use the local covariance matrices of the sample partition associated with each individual. We apply a deterministic approximation to the theoretical random mutation operator in order to avoid the variability introduced by random sampling. Mutations are computed along the axes defined by the eigenvectors of the estimated local covariance matrix:

$$\hat{\Sigma}_i(t) = \frac{1}{n-1} \sum_{j=1}^{n} (\mathbf{x}_j(t) - \mathbf{y}_i(t)) (\mathbf{x}_j(t) - \mathbf{y}_i(t))^T \delta_{ij}(t).$$

Let $\Lambda_i = \operatorname{diag}\left(\lambda_j^i; j = 1,2,3\right)$ and $\Phi_i = \operatorname{diag}\left[\mathbf{e}_j^i; j = 1,2,3\right]$ denote, respectively, the eigenvalue and eigenvector matrices of $\hat{\Sigma}_i(t)$. Then the set of mutations generated along the axis of $\mathbf{e}_j^i$ is:

$$P_{ij}''(t) = \left\{ \mathbf{y}_i \pm \alpha_k \lambda_j^i \mathbf{e}_j^i; k = 1, \ldots, m_j^i(t), i \in M(t) \right\},$$

$$m_j^i(t) = round\left(\frac{m_i(t)\,\lambda_j^i}{2\sum_{k=1}^{3}\lambda_k^i}\right),$$

$$\alpha_k = 1.96\frac{k}{m_j^i(t)}.$$

The set of individuals generated by mutation is

$$P''(t) = \bigcup_{i,j} P_{ij}''(t).$$

10.3.3 Selection operator

Finally, following the $(\mu + \lambda)$-strategy, to define the selection of the next generation individuals we pool together parents and children $Q = P(t)$. Selection can not be based on the original individual fitness functions $F_i(t)$ because they do not have information about the interaction effects introduced by the mutation generated individuals. The optimal approach to the implementation of the selector operator consists in computing the fitness of all the possible populations of size c extracted from $P''(t) \cup P(t)$. That means to compute the population fitness functions a number of times given by the combinatorial expression:

$$\binom{|P''(t) \cup P(t)|}{c}.$$

This computational burden largely questions the feasibility of applying this approach in any real time application. Therefore, we have tested two alternative selection operators of lesser complexity. We will describe them in the order of decreasing complexity and optimality.

Selection operator 1

One way to reduce the complexity combinatorial growth of the selection operator is to try to explore the solutions in order, performing a greedy search. The greedy selection procedure results in a complexity that roughly grows quadratically with c and λ. More precisely it requires the computation of the population fitness function $c(c + \lambda + 1)$ times. The procedure tries to select the cluster representatives in order of decreasing distortion. Given a set of currently selected cluster centers $\{\mathbf{y}_{i_1}, \ldots, \mathbf{y}_{i_k}\} \subset P''(t) \cup P(t)$, with $k < c$, we select the next cluster center as the one that added to the previous selected ones produces the smaller distortion: the minimum of the $(c + \lambda - k)$ distortions that can be computed based on the sub-populations that can be formed adding one cluster representative (individual) to the k already selected. More formally, this selection operator $\mathcal{S}^1$ can be described as follows:

$$P(t+1) = \mathcal{S}^1(P''(t) \cup Q) = P^*(t) \subset (P''(t) \cup Q),$$

where the set $P^*(t) = \{\mathbf{y}_{i_1}, \ldots, \mathbf{y}_{i_c}\}$ is constructed iteratively by selecting the indices applying the following recursive expression (note that we drop the time index for simplicity):

$$i_k = \arg \min_{\substack{i=1,\ldots,c+\lambda \\ i \notin I_{k-1}}} \left\{ \sum_{l \in I_{k-1} \cup \{i\}} \sum_{j=1}^{n} \|\mathbf{x}_j - \mathbf{y}_l\|^2 \delta_{jl}^{I_{k-1} \cup \{i\}} \right\},$$

where $I_k = I_{k-1} \cup \{i_k\}$, and $I_0 = \emptyset$ are the sets of indices considered at each step of the iteration, and the membership function is dependent on this set:

$$\delta_{jl}^{I} = \begin{cases} 1 & l = \arg \min_{i \in I} \left\{ \|\mathbf{x}_j - \mathbf{y}_i\|^2 \right\} \\ 0 & \text{otherwise} \end{cases}$$

Selection operator 2

Let be $m' = P''(t)$ the number of individuals effectively generated by mutation. The fitness function used for selection of an individual is the distortion when the sample is codified with the codebook given by $P''(t) \cup Q - \{\mathbf{y}_i\}$, more formally:

$$F_i^S(t) = \sum_{\substack{k=1 \\ k \neq i}}^{c+m'} \sum_{j=1}^{n} \|\mathbf{x}_j(t) - \mathbf{y}_k(t)\|^2 \delta_{kj}^S(t),$$

$$\delta_{kj}^S(t) = \begin{cases} 1 & k = \arg \min_{\substack{l=1,\ldots,c+m' \\ l \neq i}} \left\{ \|\mathbf{x}_j(t) - \mathbf{y}_l(t)\|^2 \right\} \\ 0 & \text{otherwise} \end{cases}.$$

The selection operator $\mathcal{S}^2$ selects the c best individuals according to the above fitness:

$$P(t+1) = \mathcal{S}^2(P''(t) \cup Q) = \{\mathbf{y}_i \in P^*(t); i = 1, \ldots, c\},$$

$$P^*(t) = \left\{ \mathbf{y}_{i_1}, \ldots, \mathbf{y}_{i_{c+m'}} \middle| i_j < i_k \Rightarrow F_{i_j}^S(t) > F_{i_k}^S(t) \right\}.$$

The computation requirements of this selection operator are quadratic in the number of cluster representatives, and they can be made linear using the simple programming trick of precomputing the two nearest cluster representatives. However, this selection operator is clearly suboptimal. The experimental works try to asses the trade-off of its suboptimality versus its computational efficiency. This second definition of the selection operator involves the fitness of the whole population with the addition of the mutations generated. This makes the algorithm sensitive to the number of mutations generated. A large number of mutations decrease the discriminatory power of $F_i^S(t)$. The number of allowed mutations must be carefully chosen. In the experimental results it has been allowed as many mutations as individuals in the population.

10.3.4 The adaptive application of the ES

The adaptive application of the Evolution Schema involves the mapping of the two discrete time axes: the data time parameter τ and the generation t of the Evolution-based Adaptive Strategy. The most conventional approach would allow for several generations between data samples, so that :

$$\tau = \left\lfloor \frac{t}{T} \right\rfloor + 1,$$

where T is the number of evolution generations allowed between input data samples. In the context of Color Quantization of image sequences, T is the number of generations computed between presentations of image samples. The initial condition corresponds to the initial color representatives provided for the first image, and the adaptation starts upon the sample from the second image. A distinctive feature of our experiments below is that we impose a one generation per frame framework, that is $T = 1$.

10.4 Experimental Results

The experimental data has been described in Appendix .

As a benchmark non adaptive Clustering algorithm we have used a variation of the one proposed by Heckbert [141] as implemented in MATLAB following [283]. This algorithm partitions the RGB cube using an exhaustive minimum variance search. It is almost optimal, and its complexity is proportional to the discretization of the color space [283].

In the application of the Evolution-based Adaptive Strategy described in the previous section to Dynamic Color Quantization, the data samples at each time instant were sets of pixels picked randomly from the image. This image subsampling was aimed to approach as much as possible the real time constraints. The algorithm was applied in the one-generation time schedule, starting on the sample of the second image, and using as initial population $P(1)$ the Heckbert palette of the first image. The populations $P(t)$ are the color palettes used to perform the Color Quantization. The results of the Evolution-based Adaptive Strategy are always shown together with the benchmark curves of figure , in order to show its adaptive behavior. The historical sequence of our experiments started in fact with the selection operator $\mathcal{S}^2$, so in the following it is the default selection operator unless stated otherwise.

In the experiments reported in this chapter we have used samples of 1600 pixels to perform the adaptive computations, and, unless stated otherwise, the distortion results correspond to the Color Quantization of the whole images. We have selected the task of Color Quantization to 16 colors as representative of the general class of image segmentation tasks based on the color information. Color Quantization to 256 colors is representative of compression tasks. The experimentation with these two number of color representatives shows that the algorithms are sensitive to the number of clusters searched.

As a general inspection of Figures 10.1 to 10.3 will confirm, the qualitative performance of the algorithms (their error relative to the optimal application of the Heckbert algorithm) decreases as the number of clusters searched increases. This result must be hold in mind when trying to design adaptive algorithms that look for the natural number of clusters (color representatives).

The first set of results refer to the application of the Evolution Strategy with the theoretical random mutation operator. These results are shown in Figure 10.1, and consist of the distortion of the Color Quantization of the 1600 pixels image samples. We have performed 30 repetitions of the adaptive application of the Evolution Strategy. We give in the figure the average and 95% confidence interval of the results of these repetitions. It can be seen that the random mutation operator introduces a high uncertainty on the quantization results. This uncertainty is greater in the images that show the greater distribution variation relative to their predecessor in the sequence. It can be also appreciated that the confidence intervals are more large in the case of 16 colors than in the case of 256 colors.

The random mutation operator produces some very bad results, sometimes much worse than the Time Invariant applicaton of the Heckbert algorithm. That is, the random mutation operator gives a significative probablity of having responses well far from the desired adaptive one. We propose the deterministic formulation of the mutation operator to avoid this uncertainty. The results of the application of the Evolution Strategy with the deterministic mutation operator on the experimental sequence are shown in Figure 10.2 given by the curves of asterisks (*). Also shown in the figure are the results of the best replica found with the application of the random mutation operator, denoted by the curve of zeroes (o). In this and subsequent figures, the distortion results refer to the Color Quantization of the whole images. The figure shows that the deterministic operator gives a good approximation while reducing greatly the computational requirements. The Evolution Strategy with the deterinistic mutation operator performs adaptively almost all the time. As can be expected from the one generation schedule, it is not able to adapt to very big variations in the pixel distributions, such as it is the case in the transition from images #10 to #11 (refer to the data visualizations in Appendix). However it shows a quick recover after this sudden transition of distributions.

Figure 10.3 compares the results obtained with the Simple Competitive Learning (SCL) and the Evolution Strategy (ES) with a deterministic mutation operator. For the case of 16 colors it can be seen that their behavior is quite similar. However the Evolutionary Strategy shows a quicker recover after sudden changes, improving over the Simple Competitive Learning after them (frames #12 and #13). The response of the SCL is smoother and has a kind of momentum that gives a slower but better recovery (frames #14 and #15). In the case of the 256 colors the responses are similar. Both approaches seem to be sensitive to the quality of the response (relative to that of the Heckbert algorithm) when the number of color representatives (clusters) increases. Both of them seem to behave adaptively most of the time, if the population changes are smooth enough.

We have performed an exploration of the sensitivity of the deterministic Evolution-based Adaptive Strategy to the ratio of sample size to the number of colors. Obviously, sample size influences the computational requirements, so that small samples are preferred. The tradeoff is the degradation in the response obtained due to the loss of information. Figure 10.4 shows the distortion of the Color Quantization of the entire images with palettes of 16 colors computed by the Evolution-based Adaptive Strategy varying the size of the sample from 400 to 25600 pixels, which means a variation of the ratio sample:codebook from 6:1 up to 1600:1.

It can be appreciated in Figure 10.4 that increasing the sample size improves the response of the Evolution-based Adaptive Strategy, approaching that of the Time Varying Min Var algorithm. An optimal tradeoff between efficiency and computation requirements could be identified with a sample size of 1600 pixels (a sample:codebook ratio of 100:1). There is, however, an strange effect for the biggest sample size. The Evolution-based Adaptive Strategy gives an anomalous response for image #15 and recovers its adaptive behavior afterwards. It must be noted the suprisingly quick recovering. This unexpected degradation of the response may be related to the definition of the significance measure employed by the selection operator $\mathcal{S}^2$ applied in the experiments up to now. The significance measure decreases its discriminative power as the number of color representatives searched increases. It can be easily seen that approaches in the average as c grows. The number of samples per cluster will decrease in the average, so that cluster will be near-empty for large c. That means that the ability of Selection Operator $\mathcal{S}^2$ to discriminate good individuals decreases accordingly. This sensitivity could explain the anomaly in Figure 10.4(d).

To test this hypothesis we have formulated the Selection Operator $\mathcal{S}^1$. We propose it as a complexity intermediate solution between the linear but suboptimal Selection Operator $\mathcal{S}^2$ and the infeasible optimal selection operator. The idea behind the next experiment is that if the Evolution-based Adaptive Strategy with Selection Operator 1 recovers the anomalies, the responsibility for them would not lie in the deterministic mutation operator, but in the incorrect choice performed by the Selection Operator $\mathcal{S}^2$. This would allow the safe proposition of the Evolution-based Adaptive Strategy with the deterministic mutation operator as a reduced complexity Dynamic Color Quantization algorithm.

Figure 10.5 shows the results of the final experiment that compare the response that both selection operators give in the application of the Evolution-based Adaptive Strategy (with the deterministic mutation operator) in the case of 16 colors. There is a general improvement of the response in all sample sizes tested. The Selection Operator $\mathcal{S}^1$ improves in the case of bad sample size selection, the strange effect detected for sample size 25600 disappears, and it performs better in the case of very small (400) samples.

These results have two meanings. The first one is the expected conclussion that Selection Operator $\mathcal{S}^1$ improves the convergence of the Evolution-based Adaptive Strategy, a natural result. However, the quadratic growth of its complexity is a serious impediment for its practical application. Selection Operator

$\mathcal{S}^2$ is more sensitive to the size of the sample and the number of mutations, due to the definition of the significance. However, it is very efficient computationally, and can be of practical use for real time applications, if properly tuned. On top of that, we remind the reader that Color Quantization is an instance of the general Clustering problem, where much bigger problems can be posed. The second, is the confirmation that the deterministic mutation operator can be applied without introducing serious degradations of the algorithm response. The anomalies seem to be due to the selection operator used.

10.5 Conclusions

We have proposed an Evolution Strategy for the adaptive computation of color representatives for Color Quantization that can be very efficiently implemented and reach almost real time performance for highly variable color populations. We have tested it on an experimental sequence of images. Some general conclussions can be drawn from our experiments. The first is that the algorithms tested perform as desired. They profit on the previous time solutions to compute fast adaptations to the present time data. The second is the sensitivity of the adaptive algorithms to the number of clusters or color representatives searched. This sensitivity is demonstrated by the relative degradation (in front of the optimal application of the Heckbert algorithm) of the responses. This sensitivity must be taken into account when trying to design adaptive algorithms that look for the natural number of color representatives.

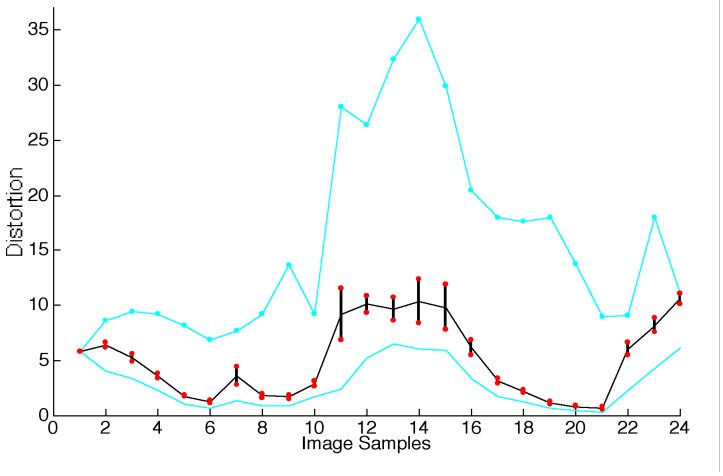
We have been able to propose a deterministic mutation operator that retain the adaptive nature of the Evolution Strategy proposed, while avoiding the uncertainty introduced by the random mutation operator. Our Evolution Strategy performance is comparable to that of some well known Competitive Neural Networks, validating it as an appropriate adaptive algorithm.

Most of the works done on Clustering deal with stationary data, assumed to come from a time invariant population. In this paper we deal with non-stationary data that comes from a population whose characteristics change with time. We have given working definitions of Non-stationary Clustering and Adaptive Vector Quantization. We have found that the problem of Color Quantization of image sequences is an instance of Adaptive Vector Quantization, that we have termed Dynamic Color Quantization. We have designed a Dynamic Color Quantization experiment that shows the characteristics that are specific of Non-stationary Clustering problems: a sequence of population distributions that show an unpredictable smooth (bounded) change.

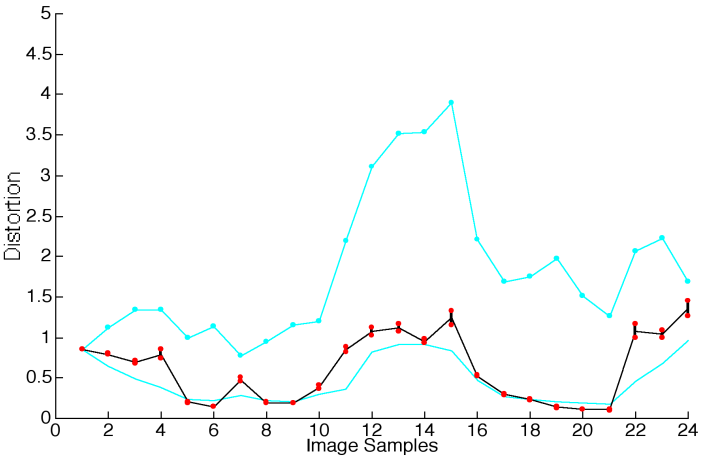
We propose an Evolution-based Adaptive Strategy for the adaptive computation of the cluster representatives at each time, and we have applied it to the computation of the color representatives for the Color Quantization of the experimental image sequence. The design of this algorithm has a main computational constraint: it must approach real time performance as much as possible, with the lowest variance induced by the algorithm itself. This lead us to formulate a deterministic version of the mutation operator, something very unusual in

the Evolutionary Computation literature. However, the algorithm remains an stochastic algorithm whose source of randomness lies in the data points themselves. We also enforce a one-pass adaptation schedule of the application of the Evolution-based Adaptive Strategy, that means that only one generation is computed from each real world time instant. Each time instant a sample of the data was considered. For Dynamic Color Quantization, we take a small sample of the image pixels, compute one generation of the Evolution-based Adaptive Strategy and use the resulting population to color quantize the entire image. The optimal selection strategy is infeasible, due to its large computational cost. This has forced us to propose two greedy suboptimal selection operators of linear and quadratic complexities, respectively. The Evolution-based Adaptive Strategy with the linear complexity suboptimal selection operator and the deterministic mutation operator performed adaptively for almost all the cases. An anomaly appeared for a large sample case. We tested the quadratic complexity selection operator combined with the deterministic mutation operator on the problematic case. The improved response backs the use of the deterministic mutation as a feasible approach, loading the responsability for the anomalous behavior to the selection operator . Therefore, the definition of the selection operator does influence more than we expected the response of the algorithm. The deterministic mutation operator does not introduce biases that could produce degradations of the algorithm.

Our work shows also the general feasibility of the so called Michigan approach, which can be applied to a wide variety of problems, besides the original classifier systems. There is, however an unavoidable tradeoff of complexity. The Michigan approach simplifies the individuals, the global population fitness functions introduces complexity in the definition of the selection operator. The Pittsburg approach maps the whole problem into each individual, but the independence of fitness functions makes the definition of the selection operator trivial. Further work must be addressed to explore this tradeoff in a diversity of problems.

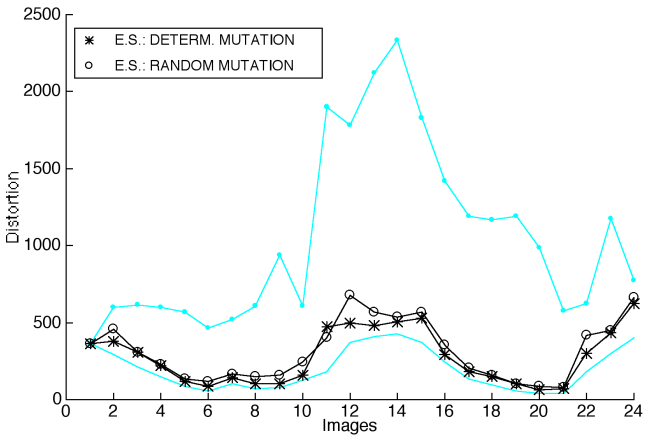


(a)

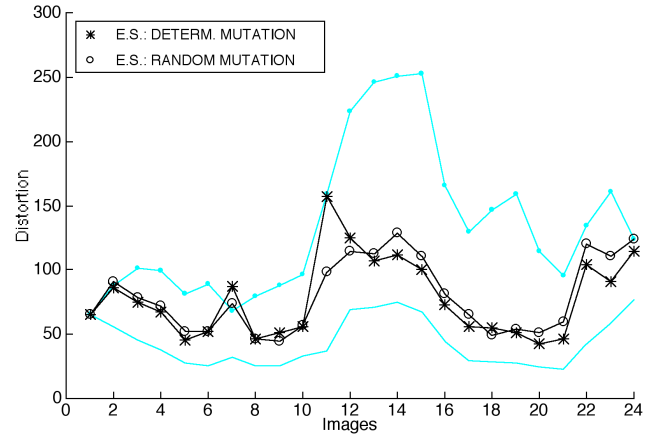


(b)

Figure 10.1: Mean distortion results and 95% confidence intervals of the application of the Evolution Strategy with the random mutation operator and the second selection operator $\mathcal{S}^2$ upon image samples of size $n = 1600$ (a) with $c = 16$, $m = 16$. (b) with $c = 256$, $m = 256$.

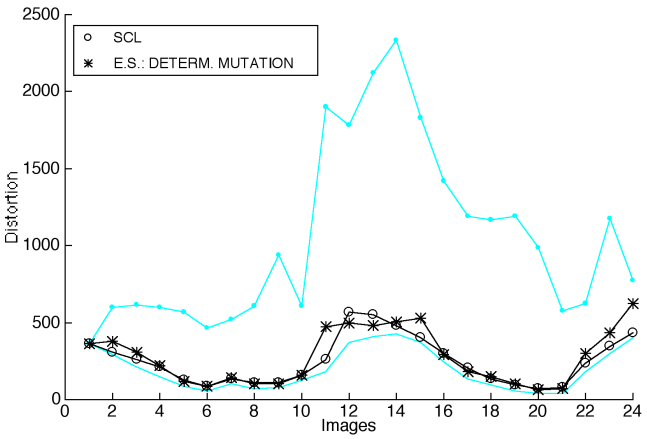


(a)

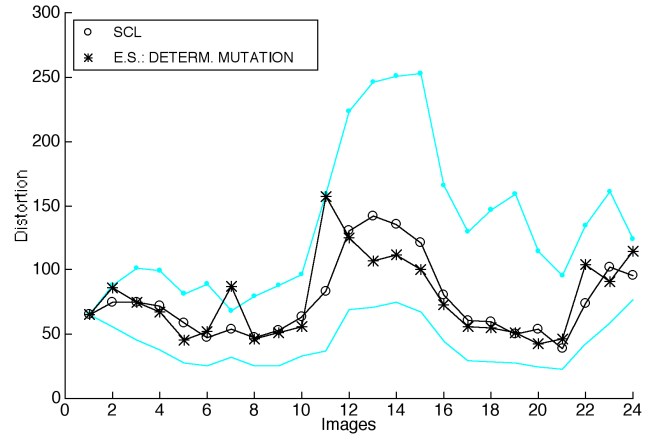


(b)

Figure 10.2: Distortion results on the image sequence from the Color Representatives computed by the Evolution Strategy with the best codebooks found after 30 replica of its application using the second selection operator $\mathcal{S}^2$, the random mutation operator, and the ones found with the deterministic operator. (a) $c = 16$ and (b) $c = 256$



(a)



(b)

Figure 10.3: Distortion results on the image sequence from the Color Representatives computed by the Simple Competitive Learning (SCL) and the Evolution Strategy with a deterministic mutation operator and the second selection operator $\mathcal{S}^2$, over image samples of size $n = 1600$. (a) $c = 16$ and (b) $c = 256$.

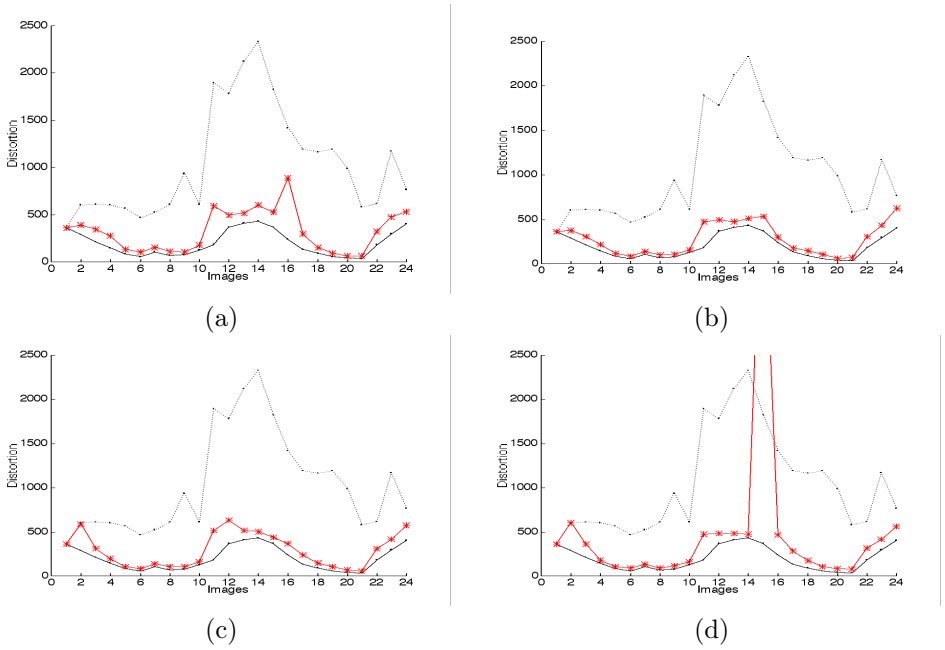
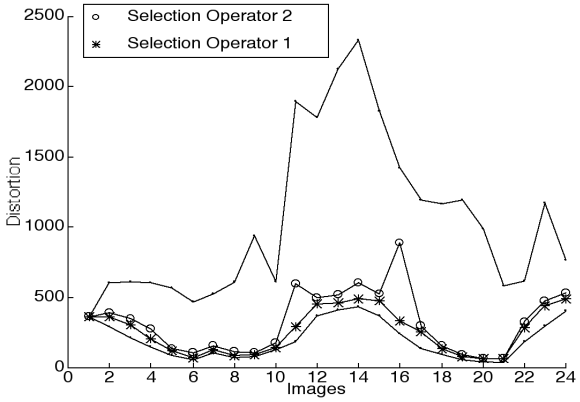
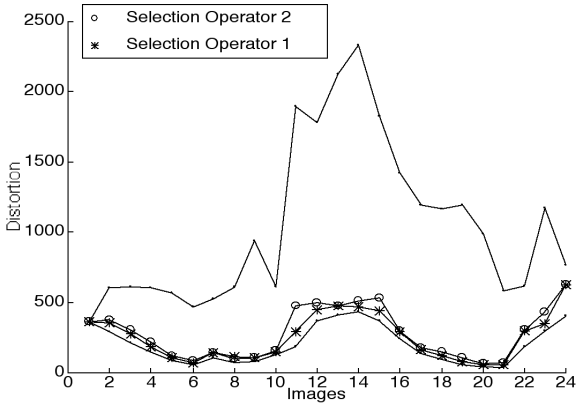


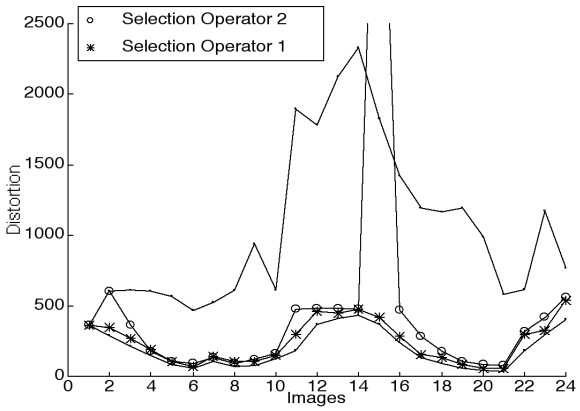
Figure 10.4: Sensitivity of the Evolution-based Adaptive Strategy with selection operator $\mathcal{S}^2$ and deterministic mutation operator to the size of the sample, $c = 16$, $m = 16$. (a) 400, (b) 1600, (c) 6400, (d) 25600 pixels.



(a)



(b)



(c)

Figure 10.5: Comparative results of the application of the deterministic Evolution-based Adaptive Strategy with Selection Operator 1 and 2. $c = 16$, $m = 16$. Samples of size 400 (a) , 1600 (b) and 25600 (c).

Chapter 11

Vector Quantization Bayesian Filtering based on an Evolutionary Strategy

In this chaper we use codebooks obtained from the Evolution-based Adaptation Strategy for Vector Quantization (VQ) of Chapter 10 for VQ Bayesian Filter (VQBF) (Appendix B) applied to noise removal and segmentation of a high-resolution T1-weighted Magnetic Resonance Image. We compare our approach with other more conventional smoothing filters. The results show that VQBF performs a smoothing that preserves region boundaries and small details. It does not show the strong boundary diffusion and displacement that are common to smoothing filters. Border detection on the filtered images is also presented.

Section 11.1 gives an introduction to the chapter. In Section 11.2, we present the VQ Evolution-based Adaptation Strategy. Section 11.3 presents the results on the application to an image and comparison with other conventional approaches. Finally, Section 11.4 gives some concluding remarks.

11.1 Introduction

Vector Quantization (VQ) [97, 148, 156] is the process of replacing the signal vectors by representatives chosen from a set named codebook. Its main application is in signal compression, although sometimes it is used as a feature extraction method for classification or as a signal filtering method [65]. This chapter reports experimental results on the use of VQ for filtering puroses. We follow an uncommon approach in the way we decompose and process an image, the VQ Bayesian filter (VQBF) which has been formalized in Appendix B. Given a codebook, VQBF is a convolution-type of operation which process each pixel in two steps:

- determine the codevector that encodes the vector given by a pixel and its

neighborhood, and

- substitute the pixel by the central element in the codevector.

The encoding process is a Maximum A Posteriori (MAP) classification and, thus, VQBF performs a kind of Bayesian image processing [**?**, 282]. In this Baysian context the VQBF distinguishing features are:

- the probabilistic model is given by the codebook,
- the model estimation consists in the search for the optimal codebook; and
- the process does not involve complex and lengthy relaxation procedures (i.e.: simulated annealing [**?**]), only the search for the nearest codevector.

VQBF approach, therefore, needs to solve the general problem of codebook design, which is a kind of clustering problem. Between the possible techniques that can be used to compute the codebook there are some methods based on evolutionary algorithms [8, 15, 18, 27, 32, 41, 47, 151, 163, 188, 189, 198, 31, 136]. In this chapter, we apply an Evolutionary Strategy [106] which is a variation of the one already presented in Chapter 10. We recall here its main features:

- vector real valued individuals,
- the genetic operator is mutation, and
- mutation is based in individuals local information.
- the population is the codebook, each individual a codevector, which induce a Voronoi partition over the input space, and a clustering of the sample.
- The mutation operator is guided by the estimated covariance matrices of the clusters.
- We have not defined any cross or recombination operators.
- The selection operator extracts the next generation population from the pool of parents and offspring generated by mutation. Experiments in this chapter use two types of selection operators, one that selects a fixed preset number of individuals for the next population, and other one that tries to determine the optimal number of individuals.

11.2 The Evolutionary Strategy specific features

The algorithmic structure of the ES is the same as in Algorithm 10.1, which we do not need to reproduce here. The population at generation t is denoted $P(t) = \{\mathbf{y}_i(t); i = 1, \ldots, c\}$. The local fitness of each individual is, its local distortion relative to the sample considered in this generation:$F_i(t) = \sum_{j=1}^{n} \|\mathbf{x}_j(t) - \mathbf{y}_i(t)\|^2 \delta_{ij}(t)$. The fitness of the population as a whole can be evaluated as $F(t) = \sum_{i=1}^{c} F_i(t)$. The data sample used is $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$.

The mutation operator is the same described in Section 10.3 of Chapter 10 with the only difference that we consider here an annealing process on the width of the spread mutation generation parameter:

$$\alpha_k = C(t)\frac{k}{m_j^i(t)},$$

where $C(t)$ decreases to zero smoothly, thus reducing the scale of the mutation perturbations. We assume that there is a monotonic improvement in the solution in each generation. Finally, we define two selection operators following the $(\mu+\lambda)$-strategy with an elitist constraint to ensure convergence: the selected population is accepted only if it improves on the previous one. We pool together parents and children generated by mutation $Q = P(t)$. The first selection operator considered is identical to $\mathcal{S}^2$ defined in Section 10.3. The second selection operator will be denoted $\mathcal{S}^3$ for notational consistency. This operator is designed to look for the optimal number of codevectors.

Let us denote $F^s(t)$ the fitness of population $P''(t) \cup P(t)$. Now, we define $F_i^s(t)$ as a linear combination of local distortion and entropy increment for each individual $\mathbf{y}_i$. We previously normalize both terms to the interval [0, 1]. Depending on the relative weight η, the selection operator gives more priority to the local distortion or to the entropy increment:

$$F_i^s(t) = D_i^2(t) + \eta \triangle H_i(t),$$

where the local distortion $D_i^2(t)$ is identical to the individual fitness defined in Section 10.3 to formulate selection operator , that is:

$$D_i^2(t) = \sum_{\substack{k=1 \\ k \neq i}}^{c+m'} \sum_{j=1}^{n} \|\mathbf{x}_j - \mathbf{y}_k(t)\|^2 \delta_{kj}^S(t),$$

$$\delta_{kj}^S(t) = \begin{cases} 1 & k = \arg \min\limits_{\substack{l=1,\ldots,c+m' \\ l \neq i}} \left\{\|\mathbf{x}_j - \mathbf{y}_l(t)\|^2\right\} \\ 0 & \text{otherwise} \end{cases}.$$

The entropy of each individual $H_i(t)$ is computed along the same strategy followed to compute the local distortion: we compute the entropy of population $P''(t) \cup P(t) - \{\mathbf{y}_i(t)\}$ for each $\mathbf{y}_i(t) \in P''(t) \cup P(t)$. The individual entropy increment is measured by $\triangle H_i(t) = H(t) - H_i(t)$, where $H(t)$ is the entropy of the whole population. Let n_i the number of data items classified into the i-th individual decision region, then we calculate $H_i(t)$ as:

$$H_i(t) = \sum_{\substack{k=1 \\ k \neq i}}^{c+m'} \frac{1}{n_i} \log \frac{1}{n_i}.$$

The selection operator $\mathcal{S}^3$ iteratively adds the best individual until a stopping condition is met. This stopping condition is defined over the relative increment of the accumulated fitness functions of the selected individuals that can be above a predetermined threshold T:

$$P(t+1) = \mathcal{S}^3(P''(t) \cup Q) = P^*(t),$$

$$P^*(t) = \left\{ \mathbf{y}_{i_1}, \ldots, \mathbf{y}_{i_z} \middle| i_j < i_{j+1} \Rightarrow F_{i_j}^S(t) > F_{i_{j+1}}^S(t) \;\&\; \right.$$

$$\left. \& \left| \frac{\sum_{m=1}^{j+1} F_m^s(t) - \sum_{m=1}^{j} F_m^s(t)}{\sum_{m=1}^{j} F_m^s(t)} \right| < T \right\}$$

11.3 Experimental results

In this section we present the visual results of the application of the VQBF based on the codebooks computed by the ES over a high resolution MRI image. The sample size used by the ES for the codebok design was the 10% of the original image. In experiments with selection operator $\mathcal{S}^2$ the number of generations is set to 30 and the number of codevectors is set to $m = c = 16$. In experiments testing the selection operator $\mathcal{S}^3$, the number of generations is set to 20 and the initial number of classes is the same as before. Other parameters for the evolution strategy are $T = 0.03$ (selection threshold) and $\eta = 0.5$ (more priority to local distortion than to entropy increment).

At the time, the end interest of these images is for medical-biological inspection, so the visual evaluation was the prime concern. Therefore, we present the visual results of the application of VQBF and several conventional approaches: the Median filter, the Gaussian smoothing, Gray level Morphological filters and the Wiener filter with noise self-estimation. To highlight the differences of the different filtering approaches, we show in the figure the equalization of the images after filtering. In all the cases we have considered neighborhoods of size 3x3, 5x5 and 7x7.

The approach has been tested over a Magnetic Resonance image obtained by the Unit of Magnetic Resonance of the Universidad Complutense. The original image is of 718x717 pixels and is shown in Figure 11.1(a). The objective of the work is to enhance the image with some denoising algorithm and to detect the infected region enclosed by a white square in Figure 11.1(a). The processing of the image must therefore, eliminate the Gaussian noise while preserving most of the structure of the image, specially in the interest region. To appreciate the denoising effects of the algorithms we perform the equalization of the original image (Figure 11.1(b)) and of the images after filtering shown in Figures 11.2 to 11.8.

The usual method to process noisy images before segmentation, when there is no known model of the distortion and the noise, is the application of smoothing filters. The results obtained by the application of the median filter, Gaussian

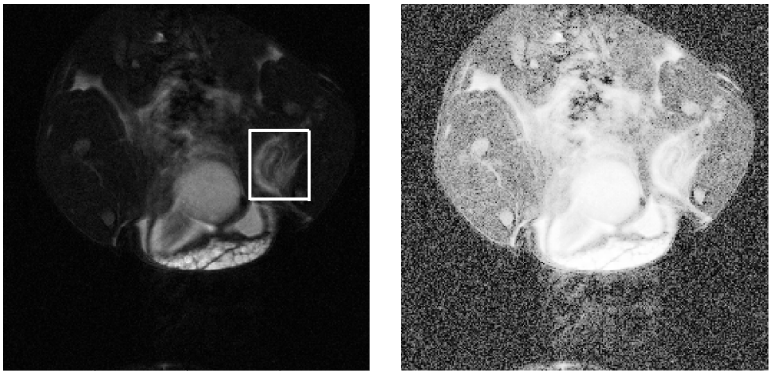
smoothing, gray level morphological filters and the Wiener filter appear, respectively, in Figures 11.2 to 11.6. The results of our VQBF with fixed number of classes are shown in Figure 11.7 and with variable number of classes in Figure 11.8. Figures 11.9 to 11.16 shows the borders detected in the filtered images by applying a Laplacian operator on the filtered image and thresholding it.

If we consider the results in terms of denoising and region segmentation, the general effect of conventional smoothing filters (Gaussian, Morphological, Median and Wiener) is a diffusion that distorts the region definition, blurring its boundaries. This negative effect increases as the size of the kernel increases. However, the VQBF shows a good denoising response while preserving the region definitions. Focussing into the infected region highlighted in Figure 11.1(a), it can be appreciated that the interest region is heavily blurred in Figures 11.2 to 11.6 while it is well preserved in Figures 11.7 and 11.8. Besides, VQBF shows no degradation by over blurring as the kernel size grows. The effect of our strategy to determine the optimal number of clusters can be appreciated comparing Figures 11.7 and 11.8, with Figures 11.15 and 11.16. Although the number of classes found is greater than in the constant size case, the visual results show some improvement, especially in the images of the detected borders. The search for the optimal number of classes produces the disappearance of the darker tissues in the image. However the infected region is well preserved in all the cases.

Focusing on the borders detected before and after filtering the image, the excellent properties of the VQBF are more clearly exhibited. Figure 11.9 shows the borders detected in the original image. The smoothing decreases the magnitude of the detected borders, displaces and diffuses them. The extreme bad result is for the Gaussian smoothing with kernel 7x7 whose detected borders have a very small magnitude that needs a very low threshold. Conventional filters either loose the interest region or preserve many noisy borders. As the codevector dimension increases, the VQBF border detection improves. The detection of the optimal number of classes gives the best results in terms of isolating interesting regions. Both instances of the 7x7 VQBF preserve the main boundaries, especially in the interest region.

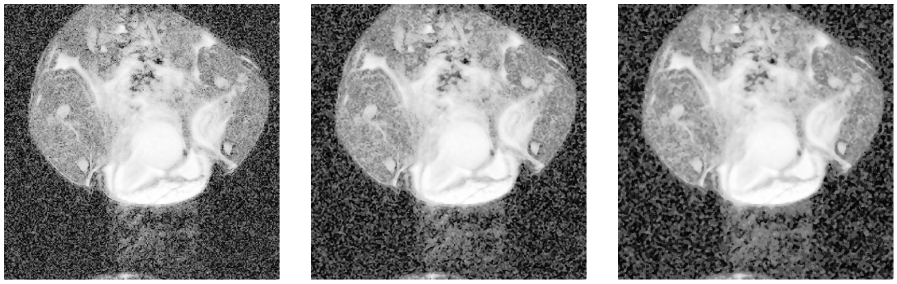
11.4 Conclusions

We have worked the application of an Evolution-based Adaptation Strategy to estimate a vector quantizer with fixed and variable number of classes that is applied as filtering mechanisms in the framework of VQBF. We have shown the results of those approaches against the results obtained by other conventional filtering and smoothing techniques widely used for noise removal. Ours approaches do not blur the image as the neighborhood size increases, and at the same time the noise is removed more efficiently as the neighborhood size increases. This is more evident in the parts of the image that correspond to empty space, where our approach removes almost all the noise. The border detection also shown that VQBF defined regions of interest with accuracy.



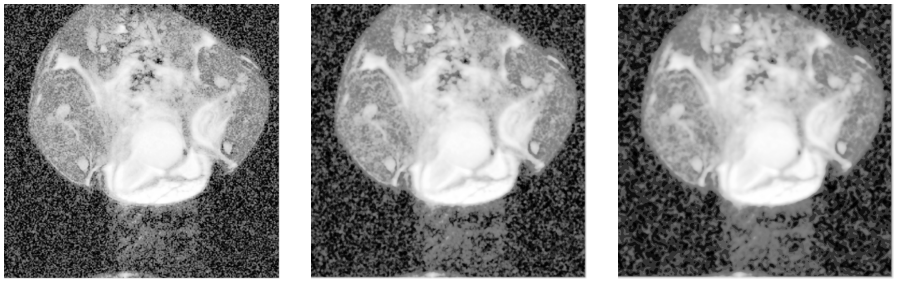
(a) (b)

Figure 11.1: Original image with the interesting region indicated by a white square (a). After equalization (b).



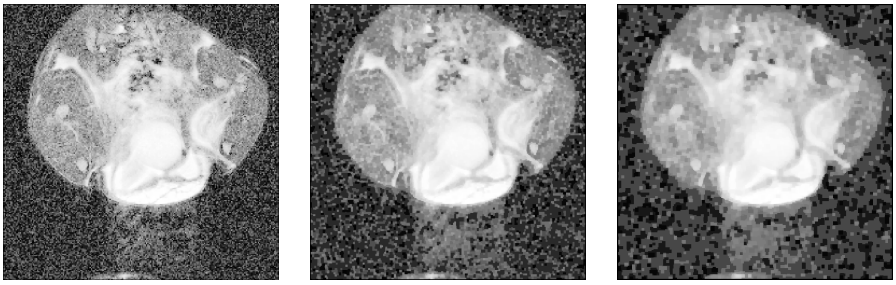
3x3 5x5 7x7

Figure 11.2: The results of with Median Filter with several neighborhood radius sizes after equalization.



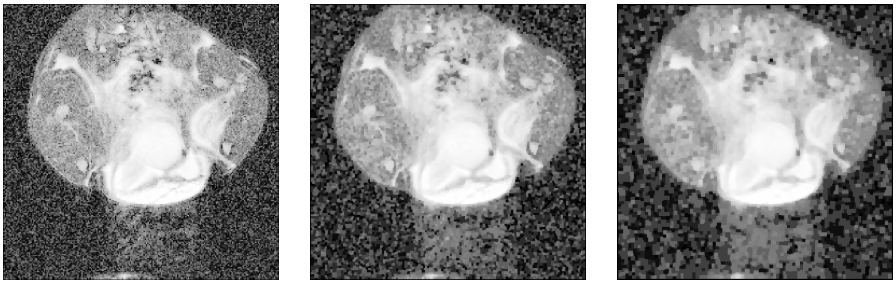
3x3 5x5 7x7

Figure 11.3: The results of with Gaussian Filter with several variance parameters after equalization.



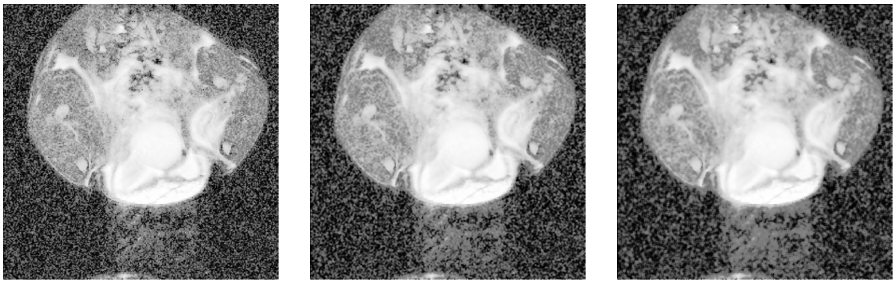
3x3 5x5 7x7

Figure 11.4: The results of with Opening+Closing Morphological Filter with several neighborhood radius, after equalization.



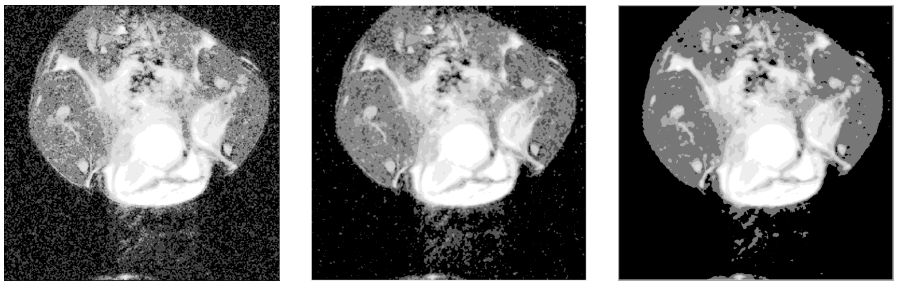
3x3 5x5 7x7

Figure 11.5: The results of with Closing+Opening Morphological Filter with several neighborhood radius, after equalization.



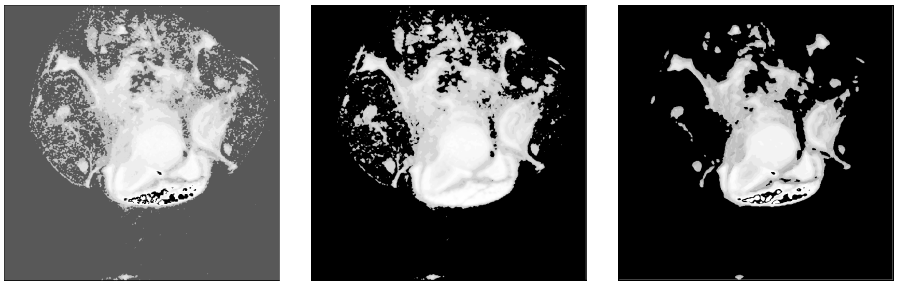
3x3 5x5 7x7

Figure 11.6: The results of with Wiener Filter with several neighborhood radius, after equalization.



3x3 5x5 7x7

Figure 11.7: The results of ES + VQBF filtering with several neighborhood radius using selection operator $\mathcal{S}^2$, with 16 codevectors, after equalization.



3x3 5x5 7x7

Figure 11.8: The results of ES + VQBF filtering with several neighborhood radius, after equalization. The number of classes is determined automatically by selection operator $\mathcal{S}^3$ (number of classes obtained from left to right: 24, 21, and 22).

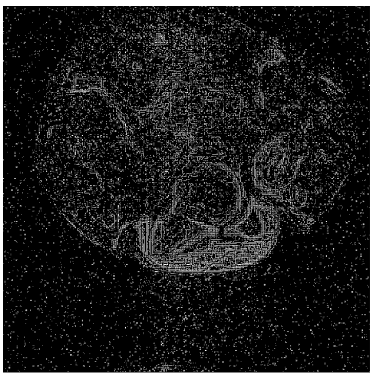
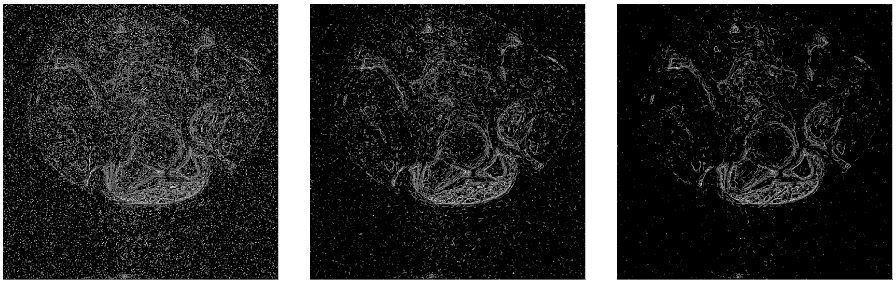


Figure 11.9: Borders detected in the original image (threshold 32).

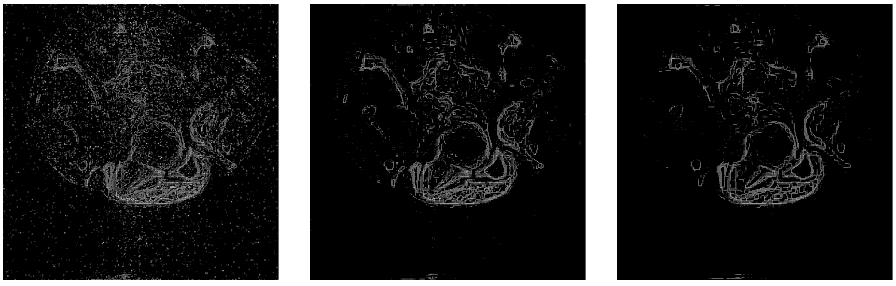


3x3 5x5 7x7

Figure 11.10: Borders of images filtered with Median Filter method and several neighborhood/radius sizes (threshold 16).

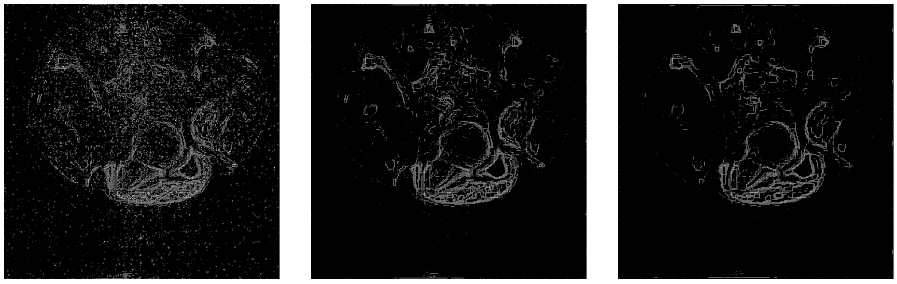
3x3 5x5 7x7

Figure 11.11: Borders of images filtered with Gaussian Filter method and several variance values (threshold 12).



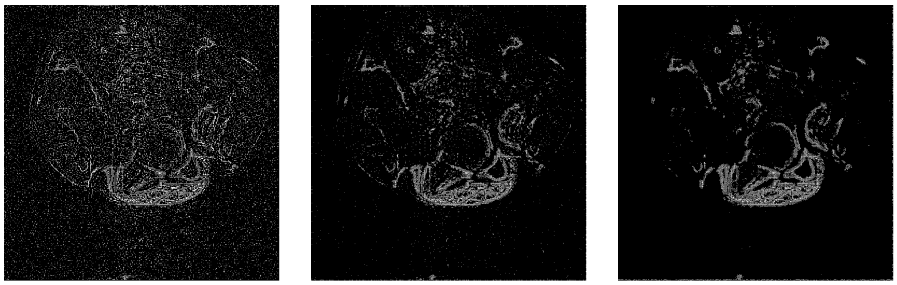
3x3 5x5 7x7

Figure 11.12: Borders of images filtered with Opening+Closing Morphological Filter method and several neighborhood radius sizes (threshold 32).



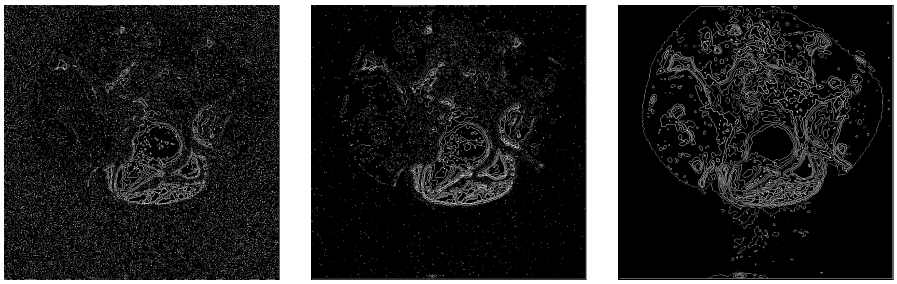
3x3 5x5 7x7

Figure 11.13: Borders of images filtered with Closing+Opening Morphological Filter method and several neighborhood radius sizes (threshold 32).



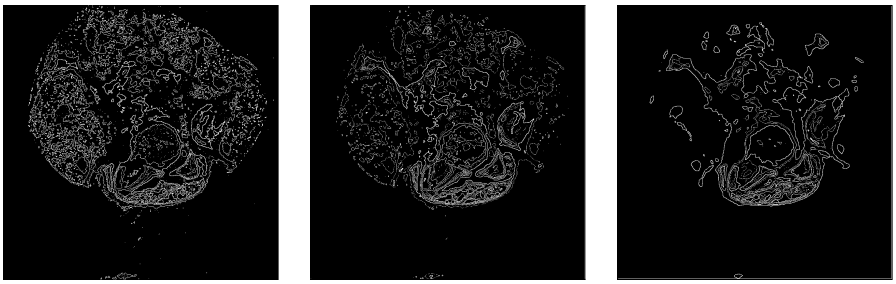
3x3 5x5 7x7

Figure 11.14: Borders of images filtered with Wiener Filter method and several neighborhood radius sizes (threshold 12).



3x3 5x5 7x7

Figure 11.15: Borders of images filtered with ES-VQBF method and several neighborhood radius sizes, using selection operator $\mathcal{S}^2$. The number of classes is constant $c = 16$. (threshold 32).



3x3 5x5 7x7

Figure 11.16: Borders of images filtered with ES-VQBF method and several neighborhood radius sizes, using selection operator $\mathcal{S}^3$. The number of classes is constant $c = 16$. (threshold 32).

Chapter 12

Occam Filters for VQBF number of classes

This chapter reports the application of an approach to the determination of the number of codevectors in a VQBF approach based on the idea of Occam filters. Occam filters use the fact that signal noise can be cancelled out by the signal loss produced by a lossy compression algorithm. In VQBF, the compression control parameter is the number of codevectors in the codebook. Tuning this paramater to obtain noise cancellation in the image is equivalent to try to determine the number of image block classes in the image.

Section 12.1 gives an introduction. Section 12.2 reviews the definition of Occam filters. Section 12.3 dicusses the application of Occam filters to VQ design. Section 12.4 presents some experimental results. Finally, Section 12.5 gives the chapter conclusions.

12.1 Introduction

Occam filters were proposed by Natarajan in [199, 200, 201]. It consists in the aplication of lossy compression algorithm as a signal filter to remove additive noise. The approach in founded in the noise cancellation induced by the compression/decompression process. It has been shown [201] that the approach works for general additive noise if the compression algorithm is *admissible*.

We work on the application of the Occam filter approach to the estimation of the *natural* codebook size in the VQ design process [97]. Other approaches to codebook size determination found in the literature consist in the addition of regularization terms to the clustering objective function minimized to perform the codebook design [49], or the formulation of heuristics for the growing and pruning of the codebook, sucha as the Growing Neural Gas [87]. The Occam filter approach gives a clear intuition on the meaning of the codebook size selection process, corresponding to a measure of the noise-free signal complexity. This interpretation is adequate for the unsupervised segmentation processes that we

perform, specifically on the MRI data.

The Occam filter approach involves estimation of the rate-distortion curve, mapping the distortion obtained after compression/decompression to the compression ratio. Therefore, codebook estimation must be repeated a large number of times. In this chapter we apply the SOM [168] in on-pass training schedule [110]. The experimental results refer to the segmentation of MRI data of a human embrio.

12.2 Occam filters

The idea of using compression algorithms as low-pass filters is not new, but Natarajan [199, 200, 201] gave it a more precise formulation. We consider the vector quantization referre to a given codebook of the image data as a lossy compression algorithm. If we try to tune the number of classes needed to obtain noise cancelation by compression signal loss, without signal degradation, the we are determining the optimal number of classes for data clustering. We reproduce in this section some of the original arguments for Occam filters in their more general view.

Conjecture 1. *When a lossy compression algorithm is applied to a noisy signal, fitting the signal loss to the noise magnitude, signal loss and noise tend to cancel each other instead of accumulate.*

We recall some notation:

- We consider Baire functions in the $[0,1]$ interval, whithout loss of generality.
- For a Baire function f and a natural number n, f_n is the sequence of samples of f uniformly separated by $1/n$. Specifically

$$f_n = \{f(0), f(1/n), f(2/n), ..., f((n-1)/n)\}. \tag{12.1}$$

- A sequence f_n is a vector in $[0,1]^n$, therefore, the difference between two sequences f_n and g_n is their vector difference. A norm in this space is denoted $\|f_n\|$. The norm l_2 is defined $\|f_n\|_2 = \frac{1}{n}\sum_{i=0}^{n-1}(f(i/n))^2$.
- Relative to a norm $\|.\|$, a lossy compression algorithm C is a programme taking as input a sequence f_n and a tolerance $\epsilon > 0$, producing as output a binary string binario s codifying the sequence g_n under the constraint $\|f_n - g_n\| < \epsilon$.
- A decompression algorithm D takes as input the binary string $s = C(f_n, \epsilon)$, producing the recovered sequence $g_n = D(s)$.
- The signal loss introduced by C (i.e. mean square error) is $d = \|f - g\| = E\left\{(f_i - g_i)^2\right\}$.

Algorithm 12.1 Coding/decoding filtering algorithm

input $\widehat{f}_n$
begin Let $\|v\|$ be the noise magnitude
Run compression $C\left(\widehat{f}_n, \|v\|\right)$
Run decompression D to obtain the filtered sequence g_n
end

- v is a random variable modeling noise, v_n is a sequence of n independent observations of v.
- $\widehat{f}_n = f_n + v_n$ is the noise corrupted sequence.
- Noise magnitude is defined as $\|v\| = \lim_{n\to\infty} \|v_n\|$.

Definition 2. The lossy compression algorithm C is *admissible* if the reconstruction error $g-f$ and the recovered signal g are uncorrelated, i.e. $(f-g)\cdot g = 0$ where $f \cdot g = E\{f_i g_i\}$.

Definition 3. The rate-distortion curve for a lossy compression algorithm C, denoted $R_C(f,d)$, is the compression ratio (average number of bits per input data) for signal source f as a function of distortion d.

Definition 4. The rate-distortion for source f is minimum over all possible lossy compression algorithms

$$R(f,d) = \min_C \{R_C(f,d)\}. \tag{12.2}$$

The fundamental convergence theorem of Occam filters [201] is as follows:

Theorem 5. *Let it be g the signal obtained by the compression and decompression of signal $f+v$ using an admissible compression algorithm C, with maxumum loss tuned to $\|v\|$. The residual noise of the reconstructed signal is bounded by*

$$\frac{\|f-g\|}{\|f\|} \leq \left(2+\sqrt{2}\right)\sqrt{\frac{R_C(f+v,\|v\|)}{-R'(v,\|v\|)\|f\|}}, \tag{12.3}$$

where $R'(v,\|v\|)$ is the left derivative of the rate-distortion function $R(f,d)$ relative to the distortion d evaluated in $d=\|v\|$.

Theorem's proof can be found in [201]. This theorem is the formalization of the intuition driving to the definition of Occam filters: if a noisy signal is compressed by an admisible compression algorithm with the loss tuned to the noise magnitude, then noise and loss tend to cancel out, with the signal loss extent depending on the noise incompresibility related to the signal. The general algorithm is presented in Algorithm 12.1.

The intuitive justification for the process is as follows:

1. Suposse that we can have access to the noise source, computing the rate-distortion curve for $C(v_n, \epsilon)$.
   (a) For $\epsilon > \|v\|$ the ratio will be high, because we can approximate noise by a constant inside this tolerance.
   (b) For $\epsilon < \|v\|$ compression ratio will be small and decreasing with ϵ.
2. Supposse that we can have access to the clean original signal, computing the rate-distortion curve for $C(f_n, \epsilon)$
   (a) For high values of ϵ the compression ratio will be high.
   (b) For small values of ϵ the compression ratio will be low.
3. When we compute the ratio-distortion curve for the noisy signal $C\left(\widehat{f}_n, \epsilon\right)$
   (a) For $\epsilon > \|v\|$ signal dominates noise and the curve follows the original signal rate-distortion curve.
   (b) For $\epsilon < \|v\|$ noise dominates signal and the curve follows the noise rate-distortion.
   (c) FOr $\epsilon = \|v\|$ la curve has an elbow that can be detected as the maximum of the second derivative.

12.2.1 On the noise magnitude estimation

Practical application of Occam filters depend on the noise magnitude $\|v\|$ estimation. [200] proposed method first builds an estimation of the rate-distortion curve, second finds the inflexion point corresponding to the maxima of the second derivative of this curve. Finally, the distorsion coordinate of this inflexion point is the estimation of the noise magnitude $\|v\|$. This identification follows from the idea that the distortion ratio curve can be approximated by two linear functions, defined in two disjoint regions of the distortion domain

$$R_C(f + v, d) \approx \alpha_1 d; d \geq \|v\|, \tag{12.4}$$

and

$$R_C(f + v, d) \approx \alpha_2 d; d < \|v\|. \tag{12.5}$$

For $d \geq \|v\|$, contribution of the compression of the noise to the compression ratio is constant and small, and the slope α_1 is small. For $d < \|v\|$ contribution of the noise compression to the compression ratio is big, dominating the signal compression, thus slope α_2 is big.

In practical applications we found that the second derivative is very sensitive to variations in the estimation of the rate-distortion curve. We have estimated the slopes of the linear approximations, α_1 and α_2 from points in the curve that can be assumed with certainty to be in the appropriate regions. We select the inflexion point as the nearest point in the rate-distortion curve to the intersection between both linear approximations.

12.3 VQ and Occam filters

VQ [97] is defined for 3D image data as follows: we have data samples $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ with $\mathbf{x}_i \in \mathbb{R}^{d\times d\times d}$, that is, samples from a stochastic process defined in a real Euclidean space of dimension $d \times d \times d$, the vector quantizer is given by the codebook $\mathbf{Y} = \{\mathbf{y}_1, .., \mathbf{y}_c\}$ formed by codevectors $\mathbf{y}_i \in \mathbb{R}^{d\times d\times d}$, where c is the size of the codebook. The coding operation is defined as

$$\varepsilon(\mathbf{x}) = \arg\min\{\|\mathbf{x} - \mathbf{y}_i\|; i = 1, .., c\} \tag{12.6}$$

and the decoding operation is defined as

$$\varepsilon^*(i) = \mathbf{y}_i, \tag{12.7}$$

recovering the original space signal from the codification using the codebook. For the design of the codebook we use the Self Organizing Map (SOM) For compression applications, the image is decomposed into non-overlapping image blocks, each image block assumed as an independent vector. In VQBF codevectors are referenced around their central pixel:

$$\mathbf{y}_i = \left(y^i_{l,m,n}; -\frac{d}{2} \leq l, m, n \leq \frac{d}{2}\right). \tag{12.8}$$

For convolution like operators, the image is not decomposed into image blocks. Instead, we process a sliding window of size $d \times d \times d$ centered in each pixel

$$\mathbf{f}_{i,j,k} = \left[f_{i+l,j+m,k+n}; -\frac{d}{2} \leq l, m, n \leq \frac{d}{2}\right]. \tag{12.9}$$

VQ-BF filtering is defined as:

$$\widetilde{f}_{i,j,k} = y^{\varepsilon(\mathbf{f}_{ijk})}_{0,0,0}. \tag{12.10}$$

VQ-BF segmentation is computed as:

$$\omega_{i,j,k} = \varepsilon(\mathbf{f}_{ijk}). \tag{12.11}$$

Codevectors embody the probabilistic model of the pixel's neighboring window. As seen in the Appendix we can assume that the filtering and classification operators correspond, respectively, to the following MAP decisions:

$$p\left(f_{i,j,k} = y^{\omega}_{0,0,0} \,|\mathbf{f}_{ijk}\right) = \delta_{\omega,\varepsilon(\mathbf{f}_{ijk})}, \tag{12.12}$$

and

$$p\left(\omega_{i,j,k} = \omega \,|\mathbf{f}_{ijk}\right) = \delta_{\omega,\varepsilon(\mathbf{f}_{ijk})}. \tag{12.13}$$

Aplication of Occam filters at VQBF design is as follows:

1. Fix the definition of the codevector space,

2. Compute the rate-distortion curve of the VQ compression using non-overlapping image blocks,

3. Compute the inflexion point of the rate distortion curve, and

4. Apply the codebook of optimal size to the VQBF image processing.

Some specific difficulties of the application of the Occam filter to the VQ design: the control parameter of this process is the compression rate specified by the number of codevectors, therefore, application of VQ is not guided by distortion, it only gives a sample of the distortion that we can obtain with such number of codevectors. The uncertainty on the compression loss implies that the distortion-rate curve will be very noisy. Estimation of the noise magnitude based on the inflexion point is, therfore, subject to uncertainty. The estimation of the distortion-rate curve can be made more precise repeating the design of the VQ at each number of codevectors, increaing the computational cost. Besides, there are two parameters determining the compression ratio: number of codevectors and codevector dimensionality. As a consequence there is no single rate-distortion curve. In the search process for the optimal codebook size and dimensionality, we have computed several rate-distortion curves exploring these parameters. Finally, all the process is conditioned to the VQ compression algorithm being admissible, which is the subject of the next proposition.

Proposition 6. *Consider an input signal whose probability distribution can be modeled as a mixture of Gaussians. The VQ whose codebook is composed of the Gaussian distributions' means is an admissible compression algorithm.*

Proof: If the input signal follows a mixture of Gaussians, then, the compression loss is a Gaussian distributed random variable of zero mean. The expectation of $(f-g)\cdot g$ is trivially zero due to the independence of the error and the signal recovered after compression and decompression with the VQ

$$E\left\{(f-g)\cdot g\right\} = E\left\{(f-g)\right\}\cdot E\left\{g\right\} = 0. \tag{12.14}$$

12.4 Experimental results

The intended task is the unsupervised segmentation of a 3D MRI data of a human embryo. The volume is made of 128 slices of 128×256 pixels each. We denote the data $\left[f_{i,j,k}; i,j = 1,..,128; k = 1,..,256\right]$. Data contains some illumination artifacts due to the experimental nature of the coil employed. Figure 12.1 shows some slices after the range transformation for visualization to 256 grayscale.

The unsupervised segmentation consists in the VQBF labelling of the pixels on the basis of the pixels' windows in order to obtain

$$\left[\omega_{i,j,k}; i,j = 1,..,128; k = 1,..,256\right]. \tag{12.15}$$

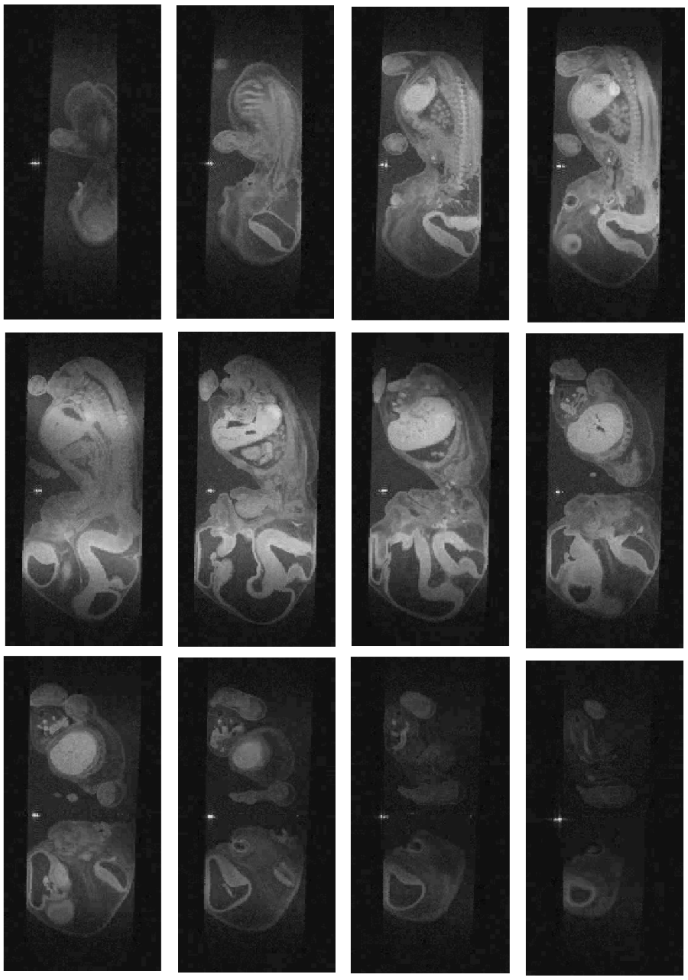


Figure 12.1: Some slices of the 3D data of an embryio used in the experiment of Occam filter detemination of the optimal number of classes.

In order to visualize the image regions identified by each class, we compute the binary images:

$$\left[f_{i,j,k}^{\omega^*}; i,j = 1,..,128; k = 1,..,256\right], \tag{12.16}$$

where $f_{i,j,k}^{\omega^*} = 1$ if $\omega_{i,j,k} = \omega^*$, and $f_{i,j,k}^{\omega^*} = 0$ otherwise for $\omega^* = 1,..,c$. We determined c by the Occam filter methodology. Each 3D data of a given class has been rendered as follows: we considered half volume rotating aroud the Z axis to show the identified structure from several vewpoints. Figure 12.2 shows some of the views obtained from segmentation using image blocks of size $5\times5\times5$ as codevector dimension. The orientations in the figure have been selected to highligtht the differences between classes. Class number is in the segmented region visualization.

Figure 12.3 shows rate-distortion curves and inflexion points (highlighted by small squares) computed for diverse codevector dimensions. The specific inflection point for codevectors of dimension $5\times5\times5$ is $c = 35$. In Figure 12.2, it can be appreciated that the first classes correspond to the saline solution containing the embryo. Classes 3 to 11 seem to be several layers between the embryo inners and the outside. From class 12 onwards, there are several regions of the embryo identified. Some volumes segmented are sparse, noise like. Others show strongly connected components. Classes 28, 29 and 30 identify two separate connected components that may correspond to the brain and abdominal regions. The class ordering may be explained by the topological preservation of SOM. The first codevectors correspond to the lower magnitude windows. Vector magnitude seems to increase with class number, so the last classes have the strongest signal.

The Occam filter has mitigated the unbalanced class problem introduced because the saline background solution has a big percentage of the volume. Codebooks with large dimension tend to assign many codevectors to this background. The Occam filter approach reduces the number of codevectors employed to model the background to three.

12.5 Conclusions

This chapter shows the results of applying the Occam filter approach to the optimal codebook size problem for a VQBF application. We find this number as the inflexion point of the rate-distortion curve of the image compression using VQ. To speed up computations we employ a sample of the entire image (volume). Once the codebook size is determined, the codebook is computed using all the image information. The SOM is used as a robust and fast codebook design algorithm. Finally, the codebook is used by a VQBF algorithm to perform the segmentation of the image. Visual results show a big spatial coherence of the detected classes with a small number of spurious classes.

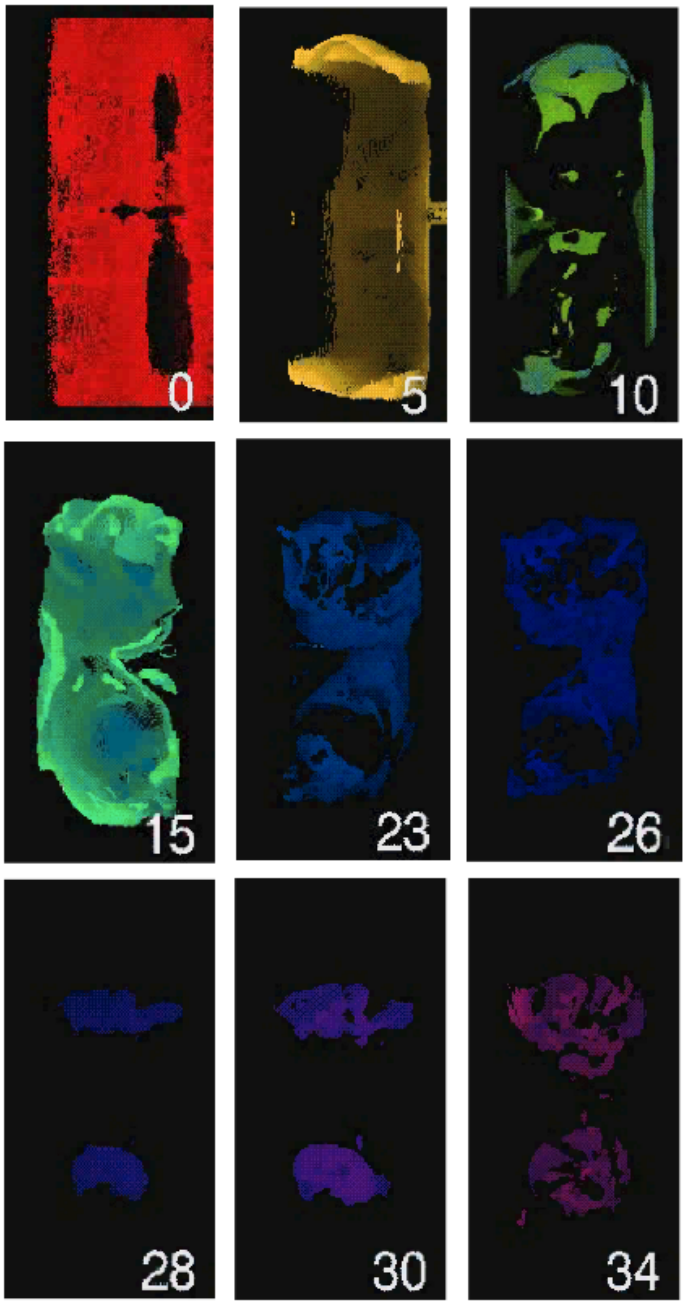


Figure 12.2: Visualization of the classes identified by the Occam filter plus VQBF. Image block size is $5 \times 5 \times 5$.

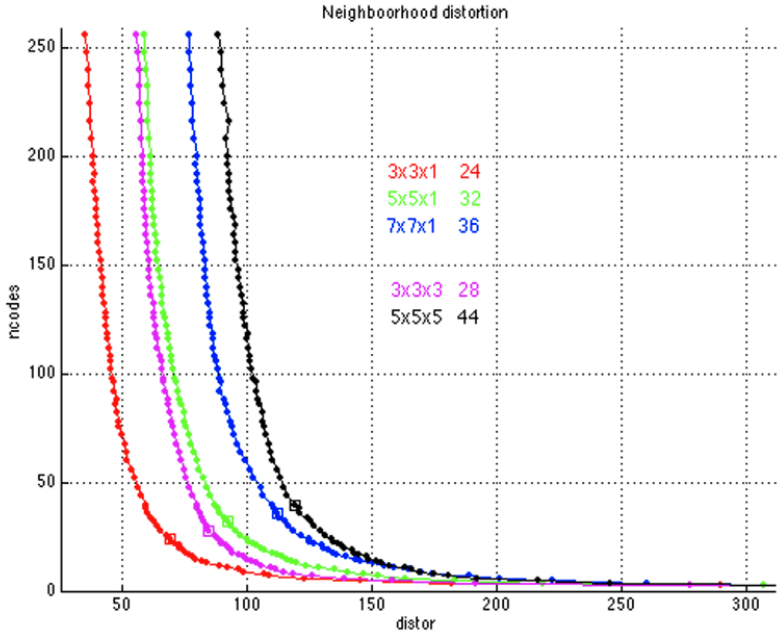


Figure 12.3: Rate-distortion curves computed by SOM in one-pass over the sample, with several codevector dimenisons.

Part IV

Supervised learning

Chapter 13

Supervised SOM for color face localization

In [127] we reported the design of face localization system decomposed in two stages. The first stage tries to localize the head region based on the analysis of the signatures of temporal difference images. It has been described in Appendix C. The second stage aims to provide confirmation of the head hypothesis through the color analysis of the head subimage. The color analysis is performed as a color quantization process. The color representatives are computed through an adaptive neural network technique, which is a supervised version of the well known Kohonen's Self Organizing Map (SOM). The contribution of the PhD candidate to the system [127] was the design of this specific SOM application and it is described in this chapter.

There have been some works that use color for the task of face localization [52, 110, 255, 262]. These works are based on a characterization of the *a priori* distribution of the skin color in the color space. They assume their color models to be invariant under capture conditions, illumination, camera distortions and so on. On the contrary, our proposition in this document is to apply an adaptive approach, so that face skin color representatives will be estimated from small samples of images. In the next sections we describe the computation of the color representatives and the experimental results.

13.1 Supervised vector quantization using SOM

The color process for face detection is a two class classification problem. Each pixel either corresponds to a face-skin color or not. This classification can be defined in terms of a Color Quantization of the images. We perform a supervised quantization where the SOM is applied independently either to face-skin pixels or to non-skin-face pixels. Two SOM networks are trained independent and simultaneously, in one pass over the sample data. The definitions of SOM are given in Section . The process can be summarized as follows:

1. Select a set of images for training, define the regions of interest that delineate the face region in the image. Select sample sets of pixels inside S^1 and outside S^0 the face regions in the images.

2. Train, in one pass over the sample set S^1, the SOM for the face-skin pixels $\mathbf{Y}^1 = \left\{\mathbf{y}_1^1, .., \mathbf{y}_c^1\right\}$, where c is the number of color clusters associated with face-skin.

3. Train, in one pass over the sample set S^0, the SOM for the non-face-skin pixels, $\mathbf{Y}^0 = \left\{\mathbf{y}_1^0, .., \mathbf{y}_{c'}^0\right\}$, where c' is the number of color clusters associated with non-face-skin.

4. For a pixel in a test image, find the nearest color representative in $\left(\mathbf{Y}^0, \mathbf{Y}^1\right)$. If the color representative belongs to $\mathbf{Y}^1$ the pixel is assigned to the face class.

Once a head subimage has been extracted applying the signature processing, the pixels are color quantized and classified as face or not-face pixels. The confirmation of the existence of a face is given based on the comparison of the percentage of face pixels with a decision threshold. As will be seen in next section, this threshold is not very high usually, because the skin region is usually relatively small in the images selected. A ratio of face pixels of 0.3 is too conservative and leads to many false rejections, because the head subimages found by the procedure described in Appendix C usually contain many other head features like neck and hair.

13.2 Experimental results

Regarding the numerical setting of the SOM for this task, we may say that the size of the samples for the experiments reported below were $\left|S^1\right| = 10500$; $\left|S^0\right| = 12500$; the number of units for both SOM instances was $c = c' = 16$, the initial neighborhood was$h_0 = 8$, the initial learning rate was $\alpha_0 = 0.1$ and the rate of convergence to the null neighborhood was $h_N = 1.1$.

Figure 13.1 shows some of the head subimages used to train the supervised SOM. Figure 13.2 shows the hand defined regions of interest that are used to extrac face color pixel train and test samples. Note that the eyes, eyebrows, lips and other features that do not share the skin color are also included in the region of interest. That means that some noise level in the class assignment must be accepted by the training algorithm. In Figure 13.3 we plot the per image accuracy of correct classification results of the pixel color classification on test images not included in the training set. In Figure 13.4 we plot the corresponding per image specificity (ratio of the number of pixels correctly assigned as face to the total number of pixels assigned as face) and sensitivity (ratio of the number of pixels correctly assigned as face to the total number of pixels hand-defined as face). These images come from different sequences that show variations of pose, scale and illumination. It can be appreciated that the supervised SOM

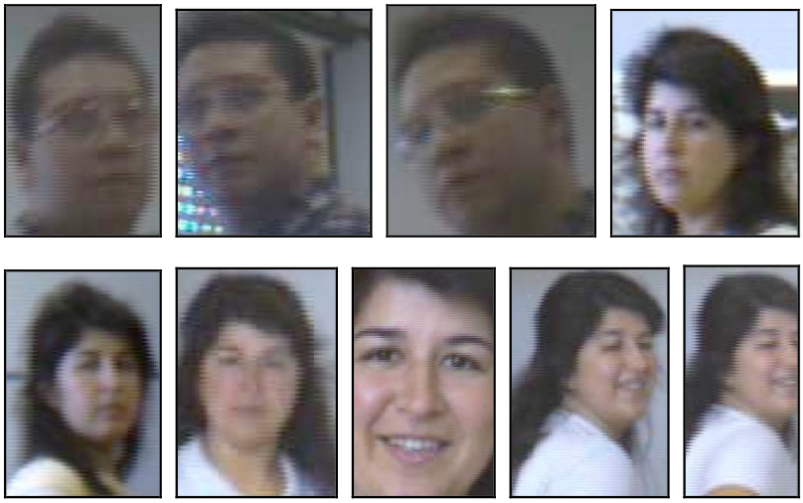


Figure 13.1: Some of the head subimages used to train the supervised SOM.

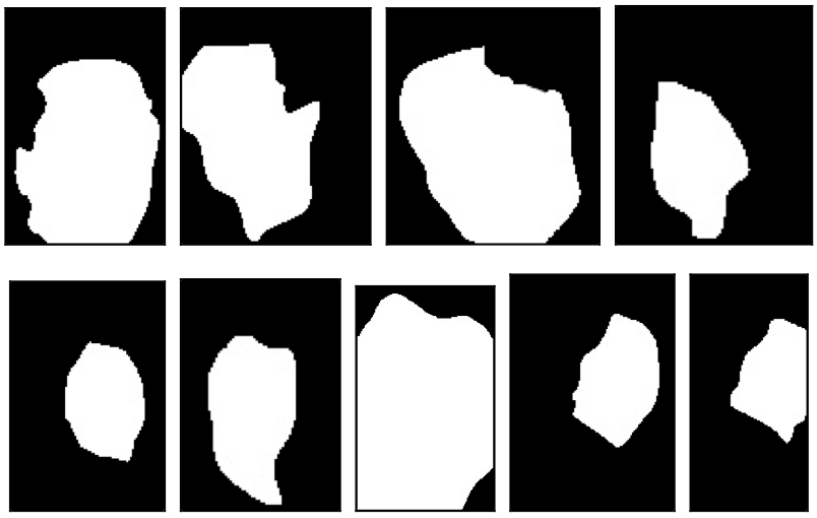


Figure 13.2: Manually selected face ROIs for the train images in Figure 13.1.

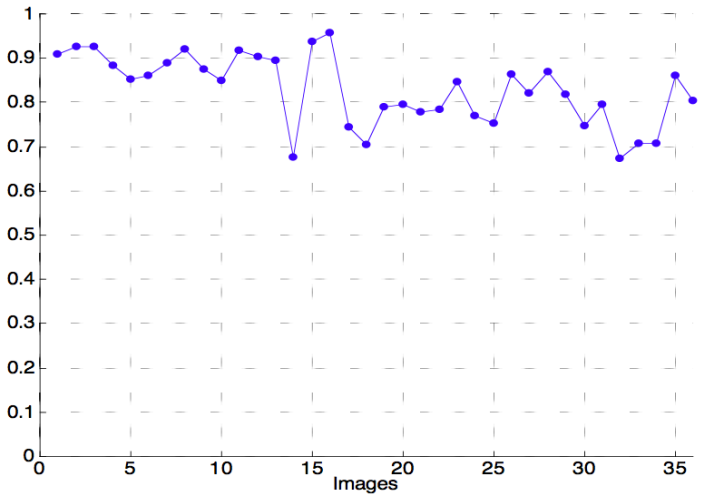


Figure 13.3: Results of the pixel color classification on test images not included in the training set: accuracy of correct clasification.

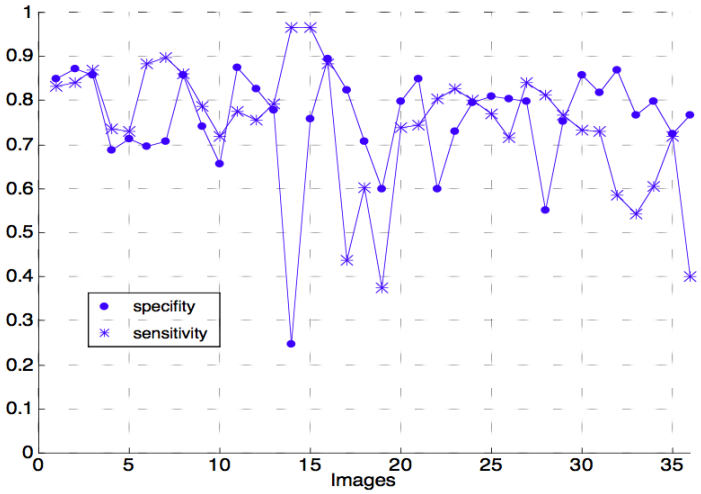


Figure 13.4: Results of the pixel color classification on test images not included in the training set: specificity and sensitivity.

Figure 13.5: Results of pixel classification on images selected by the signature algorithm.

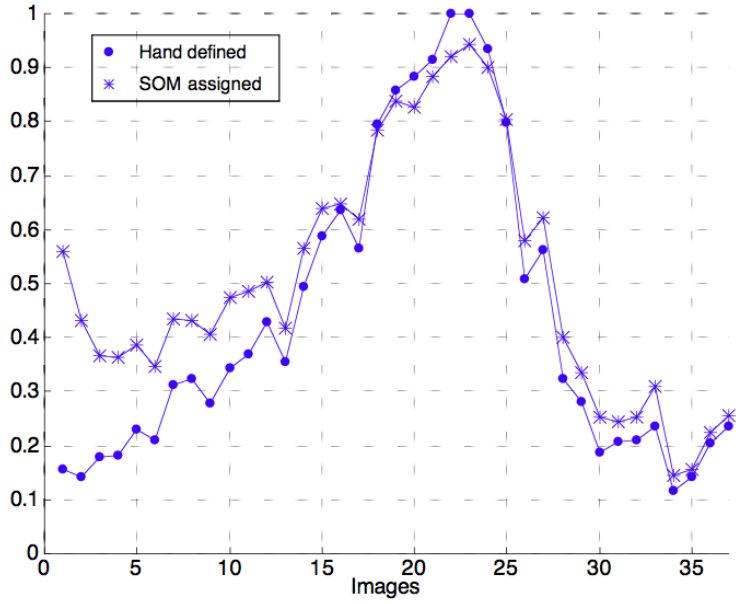


Figure 13.6: Ratios of the face area to the window area (in pixels) in the experimental sequence of images.

Figure 13.7: Some images rejected by the color segmentation with a threshold of 0.2 on the ratio of face pixels to head window size.

algorithm is very robust, and produces the correct pixel classification with great confidence, despite the simplicity of the numerical setting described at the end of the foregoing section. In Figure 13.5 we show some of the results of pixel classification in test head images selected by the signature algorithm. We have not introduced any spatial continuity constraint so that the validation of the face hypothesis is based on the number of pixels assigned to the face class. In Figure 13.6 we plot the ratio of the face area to the head window area, for the true face area as defined by the hand made regions of interest, and the face areas found by the neural classifier in the experimental sequence. A value of 0.2 for the threshold on this ratio of is a very conservative decision criteria for rejecting a head subimage. In the experimental data used for the present paper, this threshold corresponds to the rejection of images shown in Figure 13.7. These images show very small faces or face parts, that can be safely discarded as head subimages.

13.3 Conclusions

We have presented an algorithm for face localization in image sequences, based on the combination of a signature based process and a color based process. The first process tries to find rough estimations of the head, while the second provides confirmation of the head hypothesis and a more precise localization of the face region. The head localization proceeds by the computation of difference images for motion isolation. The motion image vertical projections serve to isolate individuals, whilst the horizontal projections serve to find the rough position of the head. Color processing is adaptively fitted by the application of a supervised version of Kohonen's SOM. A small sample of images can be used to fit the pixel classifier to the scenario, illumination conditions and skin face color. The color

processing decides to confirm the head hypothesis when the ratio of face pixels are above a threshold. In the experiments a ratio of 20% face pixels gives good results.

The combined algorithm shows a strong robustness against scaling effects, background cluttering, cloth textures and uncontrolled illumination. The computational cost of the algorithm is not affected by the changes in scale, because it does not involve the computation of a pyramid of images at different resolution levels. Still further confirmation of the face hypothesis can be obtained by other means [137, 222, 237, 238, 243, 255] so that the process described here can help to reduce the complexity of these other methods. The approach described is suited for still cameras, which is not the ideal setting for mobile robots. A possible way to perform the processes while the robot is moving is the computation of the motion images and motion fields from color quantized images, whose color representatives could be adaptively updated to increase the precision of the segmented motion images.

Chapter 14

VQBF for MRI tissue segmentation

This chapter gathers some results of the application of VQBF to the supervised segmentation of 3D Magnetic Resonance Images (MRI). Our contribution to the work reported in [70] by researchers of the IRM of the UCM was the technical support on the tuning of the algorithm for its application of VQBF to the medical/biological images. VQBF performs an unsupervised preprocessing of the image aiming to improve the performance of the supervised classifier performing the final image segmentation, built as a conventional feedforward artificial neural network trained with the error backpropagation algorithm, a Multilayer Perceptron (MLP). The hybrid system is composed of the VQBF filtering layer, trained with an unsupervised algorithm, and a classification layer consisting of the MLP. To test the generalization of the approach, the MLP is trained over the data from a slice containing the central region to be extracted, testing it on the remaining slices of the MRI volume. The goal is to obtain the 3D region corresponding to the tissue of interest. Results reported in this chapter refer to an experimental model of an accute inflamatory process. The results show high correlation between the manual segmentation, the histopatological data and the results of the automatic segmentation of the combined VQBF + MLP system.

Section 14.1 contains a brief state of the art review. Section 14.3 comments on the experimental images. Section 14.2 presents the hybrid system. Section 14.4 recalls the definition of the performance indices used to report validation results. Section 14.5 give the experimental results. Finally, Section 14.6 gives the conclusions of this chapter.

14.1 State of the art

Medical imaging techniques have shown high capability for differential diagnosis and the evaluation of the response to alternative therapies to some pathologies. MRI has an astounding flexibility and diagnostic capacity. Its combination

with digital image processing techniques, and automated pattern recognition increases precision in the quantification of lesions and extraction of their characteristics [155, 166], resulting in a reduction of the analysis time, operator bias, with increasing consistency in the identification of tissue types in different images. A long term goal is to establish an automated methodology for the segmentation and volumetric measurement of regions in images. Besides, objective methods are specially useful when the decision needs consensus among several physicians [155]. In this sense, artificial neural networks (ANN) [44, 142, 214] and statistical pattern recognition methods [76, 28], are appropriate tools to build automatic medical image analysis tools. ANN have been recognized as tools for decision assistance [155], allowing to build classifiers based on quantitative features extracted from the medical images: i.e., mammography diagnosis [284], anatomical brain segmentation [190, 98]. Medical image segmentation is achieved by the classification of image pixels, assigning each pixel to concrete structure or tissue.

A big problems in medical image are: low resolution and contrast, blurred boundaries and partial volume effects, variability in sizes and elastic (non-rigid) deformations of the objects (tissues, structures....). Live tissue show non linear deformations and the response to the visualization instrument may vary greatly among machines and subjects. Therefore, most systems have some interactive element, either to tune some parameter or to perform some initial selection, that can be used to train the system. For instance, active contours [194] need the specification of an initial contour that must be close to the region of interest. Such initialization must be sustained by some *a priori* knowledge, and the process is highly sensitive to initial conditions [251]. These automated image analysis methods are desired in many contexts, for instance in the monitorization and follow up of some diseases, or the impact of pharmacological treatments on tissue morphology, physiology and biochemical properties [175]. Such procedures can help to speed up the eavaluation of a mechanism and pharmakinetic, pharmadynamic and security profiles of a drug in preclinc experiments with animal models [239]. They can allow to perform longitudinal studies on the same individual, reducing economical cost and improving experimental precision, because reduces the need to combine observations on different individuals of the diverse phases of the process, avoiding premature sacrifice of the individuals for histopathological observation.

The hybrid system used in this chapter uses VQBF for image preprocessing and MLP for image classification. VQBF requires training of a codebook. For such task we use the SOM [168, 171]. SOM has been used for MRI processing, both multispectral and functional data. Hybrid systems using SOM and MLP have been applied to pathologies such as the osteosarcoma [216]. Fuzzy clustering has been applied to MRI segmentation [63, 242, 22]. Most works in MRI segmentation have been on multispectral data [252, 147]. However, the increasing acquisition time and the need for image registration justifies the research into the segmentation of mono-spectral images.

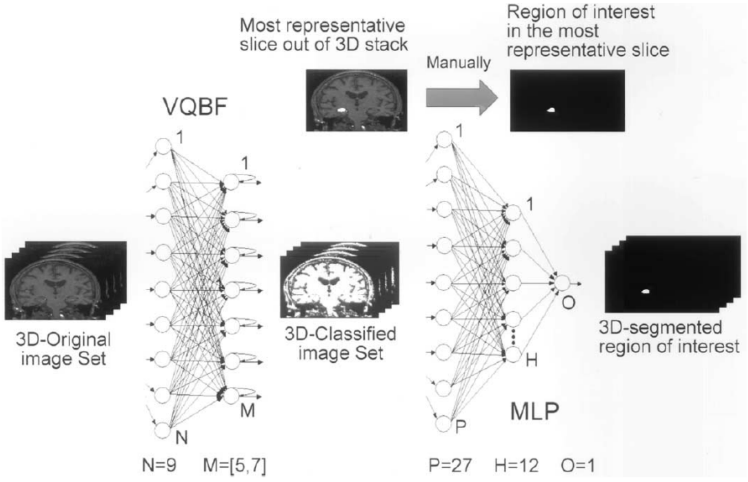


Figure 14.1: Manual and automatic segmentation procedure.

14.2 Semi-automated segmentation

The semi-automated process applies two kinds of artificial neural network to isolate and classify 3D structures, allowing the measurement of their volumes. The original data are preprocessed by a VQBF as described in Appendix B. Subsequent semi-automated region of interest (ROI) identification is achieved by the training and application of a MLP. The result of the MLP application is a volume of binary valued voxels, the values 1 giving a prediction of the ROI identification. In some applications, a new VQBF application is done on the results of the MLP-based ROI identification, i.e.: to discriminate edematose and necrotized tissues in the inflamed region [240]. Figure 14.1 shows an schema of the whole process. The complete dataset is subject to a VQBF classification process obtaining a 3D classified dataset. The most representative slice is selected and the ROI corresponding to the tissue of interest is manually drawn. This ROI and the corresponding slice in the VQBF-classified data is used to train a MLP which classifies each voxel taking as input a window around it in the VQBF-classified data. The MLP is then applied to all slices in the VQBF-classified data giving a prediction of the ROI. For validation, independent manual drawing of the entire tissue done by expert operators is available.

14.2.1 VQBF unsupervised segmentation

VQBF consists in the VQ of sliding windows centered on the processed voxels. Voxel neighborhoods are defined in three dimension: the triplet (nX, nY, nZ) specifies the neighborhood size in each dimension, where the Z axis corresponds to the slice number. The codebook used by VQBF has been extracted from the data volume applying a SOM with the following unit specific neighboring

function

$$v(t) = \frac{\eta(t)(\rho(t)+1)}{d}, \tag{14.1}$$

where $\eta(t)$ is the learning velocity, decreasing exponentially from 0.5 to 0.01, $\rho(t)$ is distance factor decreasing in the learning process from 1 down to 0.001, finally d is the distance in the index space between the winner neuron and the one being updated.

Data preprocessing consists in the classification of each voxel according to its neighborhood, given by a cube centered in the voxel [115, 107, 124]. The result is a new 3D dataset with the same spatial dimensions as the original volume. The voxel value depends on the number of classes used to categorize the tissues in the MRI data. VQBF parameters are as follows: neighborhood size $3\times3\times1$, number of SOM iterations: 3000, sample size: 3000 input windows randomly selected in the original 3D data. These parameters allow a reasonable computing time with the available resources. The neighborhood size has good edge preservation as discussed in Appendix B. Therefore, the input to the VQBF can be visualized as a layer with $N = 9$ units, and the output as a layer with $M = 5$ units. The number of output VQBF classes was decided by the expert opinion of a pathologist. Codevectors were consistently associated with the following classes [240]: background, healthy muscle, abceses, inflamated muscle and miscelanea tissues with high T2 signal in the lesion perifery, including subcutaneous grease. We did not attempt the automated determination of the number of classes.

14.2.2 Supervised identification of the ROI

MLP is a well known supervised ANN [142, 214], consisting of a feedforward network trained with the error gradient backpropagation algorithm. In our case, the MLP was a three layer network: input, hidden, and output layers. These layers are completely connected, no pruning algorithm has been applied. The input data to the MLP come from the VQBF-classified volume. The output layer consists of a single binary unit, given the probability of the voxel belonging to the ROI. The input layer consists of P units whose values are the VQBF classes of the voxels inside the window, plus two spatial values (X_p y Y_p) computed as:

$$\begin{aligned} X_p &= C_x (X_i - X_c)^2, \\ Y_p &= C_y (Y_i - Y_c)^2, \end{aligned} \tag{14.2}$$

where (X_i, Y_i) are the voxel coordinates in the slice, and (X_c, Y_c) and (C_x, C_y) are, respectively, the center of mass and the spatial standard deviation of the ROI obtained by manual segmentation of the training slice. We have tested several hidden layer configurations, finding that the best results were obtained with 12 units. This value was obtained in the empirical evaluation as the best tradeoff between computational cost, classification precision and generalization of the segmentation results. For MLP training, a human operator selects the most characteristic slice from the 3D data, providing a manual segmentation of the ROI. The binary image provided by this segmentation will be used as the

ground truth for the MLP training, which is performed on the corresponding VQBF-classified slice. The trained MLP is then applied to the remaining slices in the VQBF classified data. Training is performed using a gradient rule with a momentum term [142]. To avoid saturation in the unit transfer functions, first we sort in ascending orde the labels of the class representatives obtained by the VQBF in the ROI of the selected slice. Learning velocity is set to 0.45 and the weight of the momentum term is set to 0.01. The unit transfer function is a sigmoid function with parameter 0.5.

Besides the normalized coordinates given by equation 14.2, MLP inputs consist in the class values in 5×5 neighborhood. In any case, the neighborhood size is not a critical parameter. We have found that 3×3 and 7×7 neighborhoods produce similar results. Training is performed for a fixed number of iterations. After training, the MLP is applied to the remaining slices obtaining a binary valued volume.

14.3 Experimental images

The serial studies on mice ($N = 16$) after intramuscular innoculation of *Aspergillus fumigatus* have been realized with each animal in diverse days of the acute infection period, from days 0 to 14 after innoculation. Image acquisition was performed with an spectrometer Bruker Biospec 47/40 (Ettlingen, Germany) with tailored birdcage resonator. Animals were put in prone position, with similar positioning in each acquisition: the two rear legs inserted in the coil side by side. After an exploratory sequence, the acquisition consisted as fast 3D T_2 weighted ($256 \times 256 \times 32$) of axial images with TR/TE of 2000/67.5 ms and field of view $40 \times 40 \times 22$ mm. The proposed method was applied to the quantification of the inflamed muscle, and the necrosis in lesions with some abcess in the case of cronical acute inflamation. Only images from days 3, 7, and 14 after innoculation have been used for validation and train. Histopathological details can be found in [70].

14.4 Statistical analysis

To otain the ground truth for image segmentation, two independent human operators performed the complete manual segmentation of the ROI in some images in the original dataset, three animal models. The temporal segmentation between the segmentations is never less than one hour, to minimize the subjective error of manual delineation relative to the anatomical references. To minimize memory effects, delination is performed on non consecutive slices.

Segmentation performance results statistical significance was assesed using ANOVA and correlation methods. Significative difference between the ROI slice areas and total ROI volume for automated and manual segmentation. For all comparisons, Student-test on the means ($p < 0.01$) and F-test on the variances ($p < 0.01$) were performed to test for equality. However, comparison between

manual results and computer assisted segmentation can be misleading, because the absolute volumes and areas can be similar despite the actual voxels are not included. For this reason, a simplified ROC analysis is performed [294, 196], evaluating the following accuracy indices:

1. Similitude (aka repeatibility) measuring the overlapping of manual and automatically segmented areas [48, 162, 232, 75]:

$$S = \frac{|A \cap B|}{|A \cup B|} \tag{14.3}$$

2. Kappa similarity index (K_i) [48, 69, 17, 299]:

$$Ki = \frac{2\,|(A \cap B)|}{|A + B|} \tag{14.4}$$

3. The true positive ratio (TPF), which gives a sensitivity measure corresponding to the detection probability

$$TPF = \frac{|A \cap B|}{|B|} \tag{14.5}$$

4. False positive ratio (FPF) which is related to the false alarm probability, giving an specificity measure

$$FPF = \frac{|A - B|}{|B^c|} \tag{14.6}$$

14.5 Experimental results

The general methodology is illustrated in Figure 14.2. Figure 14.2(A) shows a central slice of the MRI data volume across the middle of the lesion. Figure 14.2(B) shows the result of VQBF classification. The affected region is well segmented including some additional tissues containing lesions. The manual delination of the ROI giving the classification ground truth is given in Figure 14.2(C). The data in Figure 14.2(B) (input patterns) and Figure 14.2(C) (output pattern) were used to train the MLP which was afterwards applied to the remaining slices of the volume, included the training slice. Figure 14.2(D) shows the result of performing a second VQBF-classification on the original data masked by the results of the MLP. The automatic counting of the voxels of each class, multiplied by the voxel volume gives the estimation of the infected tissue volume. For statistical analysis, ten central slices were selected in three animal models. Using each one as the training slice for the MLP we compute the number of voxels corresponding to the detected infected region. We applied ANOVA to compare the results with the manual segmentation, trying to determine some bias due to the selection of the training slice. There were no significative differences for any of the animal models ($p < 0.05$) neither on

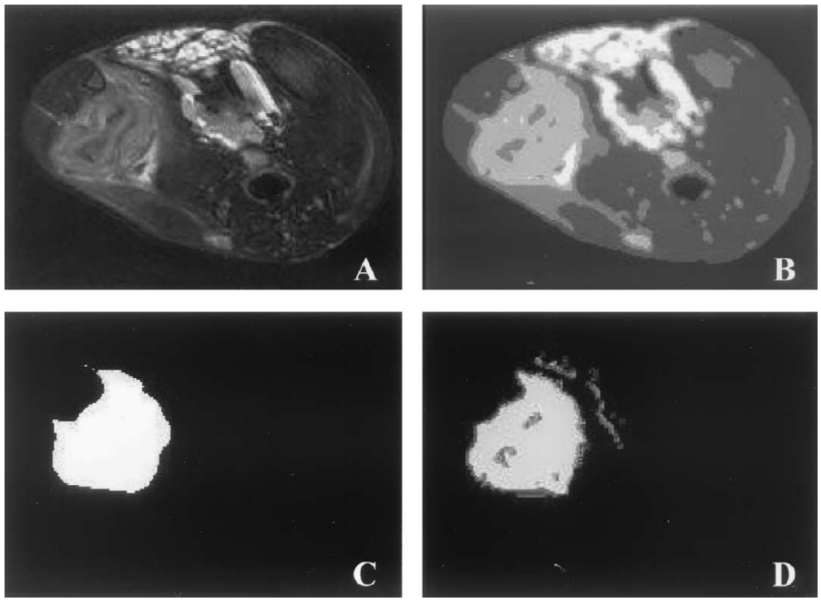


Figure 14.2: Axial slices of the mouse seven days after innoculation. (A) Original T2-weighted slice. (B) VQBF-classified image, grayscale correspond to class label. (C) Ground truth for the slice in (A). (D) VQBF classification on the results of the MLP trained in this data volume for the slice in (A).

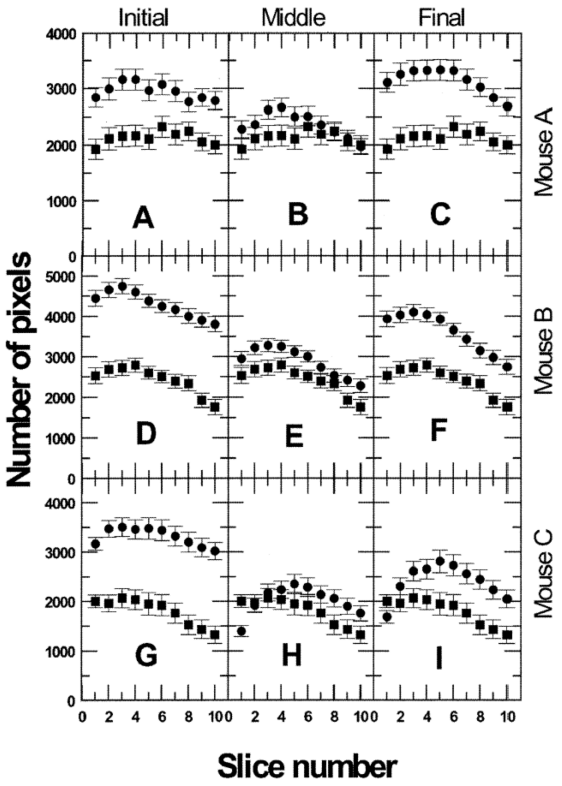


Figure 14.3: Average number of voxels corresponding to the inflamated region per slice. Square means the manual ground truth, circle means the automated segmentation.

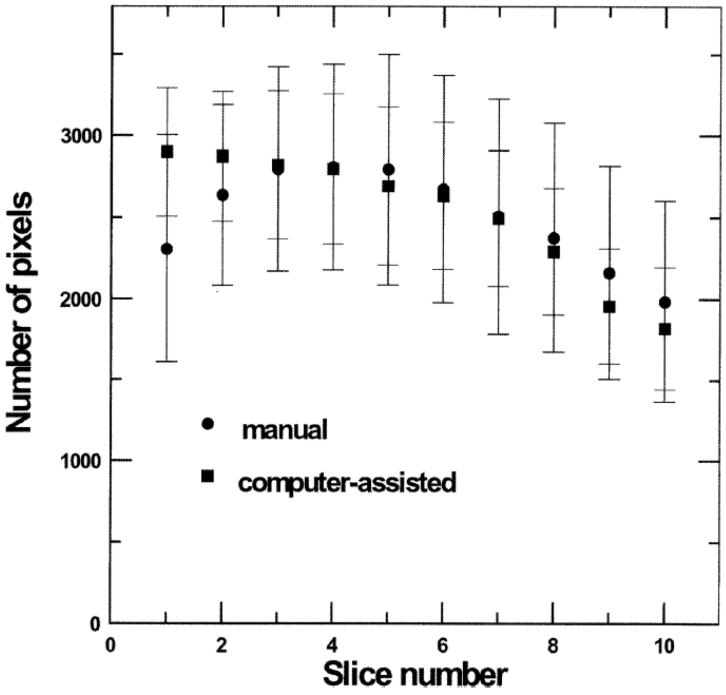


Figure 14.4: Average number of voxeles classified as infection in the ground truth (square), and the automated system when the slice for train and test is the same.

Day		3	3	3	7	7	7	14	14	14		
Mouse		A	B	C	D	E	F	G	H	I	M	SD
Slice 1	H	63.1	63.5	88.7	67.9	91.5	64.4	44.1	76.3	80.0	71.1	13.9
	ANN	68.9	82.1	76.6	71.1	89.0	71.6	35.8	71.6	77.6	71.6	14.0
Slice 2	H	64.2	78.9	65.9	56.9	92.9	50.4	46.3	80.7	51.6	65.3	15.0
	ANN	72.8	86.6	65.4	65.7	88.9	53.0	45.4	79.3	38.4	66.2	16.8

Table 14.1: Percentage of the inflamated muscle del musculo inflamado measured by the histopahological analysis (H), and by the computer assisted method (ANN). Summary of the percetages of inter-lesion tissues in nine animal models, two histological slices per each, after innoculation with *A. Fumigatus*. Day corresponds to the number of days after innoculation.

the measured areas nor on the manual segmentations by the human operators. There was some statistical significative differences among the results for the different animal models.

Figure 14.3 shows the comparison of the average number of voxels in the ground truth inflamated area (solid squares) with the ones found by the automated method (solid circles). Each column of plots in the figure corresponds to the use of three different slice sets in the MLP training. Each row corresponds to the results on a different animal model. The *three central* slices are used for the MLP training in plots A, D, G, the *four central* slices in plots B, E and H, and the *three extreme* slices in plots C, F and I. Plots A, B and C correspond to an animal model studied six days after innoculation with *A Fumigatus*. Plots D, E and F correspond to an animal model after seven days, and plots G, H and I correspond to an animal model after seven days of innoculation. As expected, difference between the ground truth and the automated segmentation decrease when training is performed on the central set of slices. On the other hand, when training is performed on the extreme slices the automated method does not agree with the ground truth. Also the number of slices used can have a dramatical effect. When the best slice set is used, the greatest discrepancies between the manual ground truth and the automated method are found in the extreme slices of the lesion, where the lesion is less prevalent, less defined and poorly delimited, as can be seen in Figure 14.4 obtained from the whole set of sixteen animal models. The number of lesion voxels per slice is shown for the manual and automated segmentations. Correlation between ground truth and automated segmentation are above 0.9 most of the times (there is a single animal model below this figure) with $p < 0.01$. Correlation between the number of voxels inside the inflamated lesion is 0.78 with high statistical significance (α <0.001).

Table 14.1 gives a comparison of the percentage of inflamated and necrotic tissue found by the automated method and the histopathological analysis of the dissecte tissues. In this study only nine animals and two slices of each

animal were processed, due to the difficulty in establishing the correspondence of the histopathological sample and the MRI slice. The mean and standard deviations in Table 14.1 show that both measurements are in agreement, with a high correlation coefficient 0.87 ($p < 0.01$), giving a high reliability to the inflamated tissue detected by the automated method.

14.6 Conclusiones

The first step in the proposed method is a VQBF classification, based on the texture representatives found by a VQ design method in monospectral MRI data. The VQBF produces a smooth segmentation of the images into tissues classes with good preservation of perceived tissue boundaries. VQBF edge preservation is comparable to other approaches such as anysotropic filtering [95]. The SOM used for the VQ design has two interesting properties: (i) is very robust against initial conditions, and (ii) codevectors found tend to be ordered, due to topological preservation of SOM. This ordering is important for subsequent processes. The role of VQBF is to reduce signal variability across individual data volumes [190], giving a voxel classification that enhances the ensuing classification processes.

Second step is the supervised training of MLP using a given ground truth. This supervised classifier overcomes translation and small deformations due to animal positioning and anatomical evolution in time. For instance, the localization and shape of the inflamatory lesion changes along the longitudinal study of the animal model.

The inflamatory processes in the animal models show complex mixtures of tissues with poor delimitation, difficult to segment with conventional methods [240]. The proposed algorithm gives good results in the identification of the total lesion. In summary, the process is rather robust agains the strong variations of the target region between slices and data volumes, with minimal human intervention.

Chapter 15

High Order Boltzmann Machines

This chapter reports the results obtained with the application of High Order Boltzmann Machines without hidden units to construct classifiers for some problems that represent different learning paradigms. The Boltzmann Machine learning algorithm remains the same regardless of the discrete or continuous nature of the domain of the variables. High Order Boltzmann Machines (HOBM) are characterized by the absence of hidden units. When HOBM are restricted to classification problems the estimation of the connection statistics needed by the learning algorithm can be done efficiently, without the computational cost of simulated annealing. In this setting, the learning process can be sped up several orders of magnitude with no appreciable loss of quality of the results obtained.

Section 15.1 contains an introdution to the chapter. Section 15.2 gives a quick revision of Boltzmann Machines. Section 15.3 introduces our notation for HOBM, and the diverse versions of the learning algorithms. Section 15.4 introduces the test problems, the definitions of the machines applied to each problem, and the results obtained trying several high order topologies. Section 15.5 gives the conclusions of this chapter.

15.1 Introduction

The Boltzmann Machine is among the first artificial neural network architectures proposed in the literature [2, 1]. However, training them was difficult due to the computational cost and the difficulty to tune the several parameters involved in the estimation of the connection statistics used for weight updating in the learning process. Weight updating in the Boltzmann Machine is based on the difference of the activation probabilities of the connections in the so-called clamped and free phases. Simulated annealing is required to compute these statistics in the general case.

Boltzmann Machines training can be much easier than was previously thought whenever two restrictions are considered.

- The first is the avoidance of hidden units, using high order connections to model the high order correlations of the input. High order connections have been referred sometimes as product or sigma-pi units [77, 220, 179]. In our work, the weights of the high order connections are computed using the same learning algorithm of conventional (order 2) connections.

- The second is the restriction to classification problems. This domain of problems is very broad including most of the application areas in which intelligent systems are applied.

Under these restrictions there is no needed to perform simulated annealing for the estimation of the connection statistics because in the clamped phase there are no degrees of freedom, while in the free phase, the asymptotic state of the output units can be easily computed as the search for the output unit with highest gain. Moreover, the learning measure, the Kullback-Leibler pseudo-distance, is convex for Boltzmann Machines without hidden units [1, 7, 6, 5], therefore

- learning is trivially robust against initial conditions, so that the initial weights can be arbitrarily set (in our works we always set the initial weights to zero),

- there is no need to realise several instances of the learning to estimate the average learning performance or to make a broad search for the best initial conditions

Despite these simplifications, we found that the quality of the results is comparable to other neural architectures, and the number of learning cycles needed is much less than in the classical approach.

Previous attempts to reduce the computational burden of learning in Boltzmann Machines include the application of mean field approximations [217, 218, 145, 33, 34] to the estimation of the stationary distribution of the network. Other authors [244] have studied a particular class of topologies, tree-like topologies, suitable to the exact computation of the activation statistics using a decimation technique. However, the class of topologies appropriate for classification can not be easily cast into the class of tree-like topologies.

15.1.1 High Order connections

High order connections are the mean to obtain full modelling power while preserving the computational simplifications implied by the absence of hidden units. Previous to our own work, we only know of sparse references [248, 144, p.211] to High Order Boltzmann Machines. These references only point to their definition, without further exploration of their capabilities.

Besides, there are references to neural network topologies with high order connections. In [215, 257] connections of order 3 are defined to obtain classifiers of two dimensional patterns that are invariant to translation, scaling and rotation. The weights of these connections are computed *a priori* based on the geometrical characteristics of the problem. In [220] Pinkas addresses the modelling of the resolution of propositional expressions by the relaxation of recurrent networks, giving algorithmic transformations of the conventional topology with hidden units into a high order topology, and vice versa. Taylor and Coobes [261] present an extension of Oja's rule that is capable of adapting the weights of higher order neurons to pick up higher order correlations from a given data set. Mendel and Wang [195] show the use of high-order statistics (cumulants) in the identification of Moving Average Systems. Although they don't use high-order connections, the cumulants are used as complementary characterisations of the system identified via neural networks. Other works (i.e. [143]) propose mixed topologies that include hidden units and high order connections trained with Backpropagation. Karlholm [160] presents a study of recurrent associative memories with exclusively short-range connections, using high order couplings (up to order 3) to increase the associative memory capacity. The main aim of his study is to asses the effect of short coupling ranges in the capacity and pattern completion ability of the networks, and little attention is paid to the effect of using high-order connections. High-order connections are still the subject of recent research works such as[260, 78, 79, 296, 269] .

15.1.2 Boltzmann Machines with non-binary units

Learning with Boltzmann Machines was mostly restricted in the literature to problems defined over $\{0, 1\}$ variables represented by binary units. We have considered machines that include non binary units. Generalised discrete units can take values in arbitrary integer intervals. Continuous units can take values in arbitrary real intervals. In both cases, the learning algorithm is a straightforward generalisation of the binary case. It suffices to consider the mean activation level of the connections, instead of the activation probabilities. The use of generalised discrete units and continuous units allows for big reductions of the number of units used to codify the problem, and, therefore, of the network complexity.

Our approach to the generalisation of the Boltzmann Machine learning paradigm is not related to previous attempts to introduce recurrent networks with discrete or continuous units. In [135], Gutzmann describes a continuous state Boltzmann Machine to solve combinatorial optimisation problems. The states of the units are restricted to the $[0, 1]$ interval. Also, some authors [12, 13, 217, 218, 145, 11, 33, 165] use the interpretation of the probability of the unit being in state 1 as a kind of continuous state. Networks with multivalued units generalising the dynamics of the Hopfield network, based on the Potts theory, have been also proposed in the setting of combinatorial optimisation [219, 100]. In a similar vein, Lin and Lee [183] propose a generalisation of Boltzmann dynamics for the case of multivalued spin like units whose states are orientations in the plane, with application to solve the navigation problem based on the definition of an artifi-

cial magnetic field. Parra and Deco [213] deal with the training of the so-called rotor neurons. The states of rotor neurons are continuous multidimensional vectors of norm 1. The authors propose an expression of the Boltzmann Machine learning algorithm for this continuous multidimensional case. A Directional-Unit Boltzmann Machine (DUBM) with complex valued units is proposed in [293]. The weights of the DUBM are also complex values. The authors define a quadratic energy function on the network configurations and a generalisation of the Boltzmann learning algorithm for the DUBM.

15.2 Boltzmann Machines

Boltzmann Machines are recurrent networks [1, 2] with binary units and symmetric weights. Each configuration of units in the network represents a state with a global energy or consensus. We follow the notation and definitions of Aarts [1], where the Boltzmann Machine is defined as a maximiser of a consensus function (versus energy minimisation in other references). The binary units considered take $\{0, 1\}$ values (versus $\{-1, +1\}$ in other references). The network operation is a stochastic process in which states of higher consensus are preferred. Once the network reaches equilibrium, the probability of finding it in a particular global state (configuration) obeys the Boltzmann distribution.

More formally the structure of a Boltzmann Machine can be describe by a triplet (U, L, W) where $U = \{u_i\}$ is a set of binary $\{0, 1\}$-valued units. In the conventional Boltzmann Machine the set of connections that defines the topology of the network is a set of pairs of units $L \subseteq U \times U$ including the bias connections of units with themselves. The set of weights associated with the connections is denoted by $W = \{w_{ij}; (u_i, u_j) \in L\}$. A global configuration of the machine is denoted by $\mathbf{k} \in \{0, 1\}^{|U|}$, and $k(u_i)$ denotes the state of the unit u_i in the global configuration $\mathbf{k}$. The consensus function

$$C(\mathbf{k}) = \sum_{(u_i, u_j) \in L} w_{ij} k(u_i) k(u_j),$$

gives a measure of the desirability of the global configuration $\mathbf{k}$.

Boltzmann Machines perform a global maximization of the consensus function [1]. The dynamics of the Boltzmann Machine are given by simulated annealing, governing the transition between different states of the units. Simulated annealing basically simulates a sequence of Markov chains with one control parameter (the temperature). Markov chain state transition probabilities drive the state changes such that stationary probability distributions of the configurations are Boltzmann distributions on the values of its corresponding consensus function. The sequence of Markov chains is defined such that as the temperature parameter descends towards zero, the associated stationary distributions assign an increasingly higher probability to configurations with the highest values of the consensus function. In the theoretical limit the probability for the process to be on a state of global maximum consensus is one. This is the appeal of

Boltzmann machines for the statement and solution of combinatorial optimisation problems. In practice, if the process is in state $\mathbf{k}$ the simulation proceeds through the generation of a neighbour configuration $\mathbf{k}'$ and its acceptance with probability

$$A_{\mathbf{k}\mathbf{k}'}(c) = \left[1 + \exp\left(\frac{C(\mathbf{k}) - C(\mathbf{k}')}{c}\right)\right],$$

leading to the stationary Boltzmann distribution of the global configurations:

$$q_{\mathbf{k}}(c) \propto \frac{1}{Z}\exp\left(\frac{-C(\mathbf{k})}{c}\right)$$

with

$$Z = \sum_{\mathbf{k}} \exp\left(\frac{C(\mathbf{k}) - C(\mathbf{k}')}{c}\right),$$

with where c is the temperature parameter and $\mathbf{q}_{\mathbf{k}}(c)$ denotes the stationary probability for the configuration $\mathbf{k}$ at temperature c. The partition function Z is needed to normalise the Boltzmann distribution. After reaching equilibrium, the parameter c is decreased by a small step.

15.2.1 Learning in Boltzmann Machines

For the purpose of learning the set of units is divided into three disjoint subsets: input, output and hidden units. The learning process consists of a series of cycles each of them with two different phases. In the first phase the examples to be learnt are *clamped* into the input and output units and the machine is simulated until reaching a state stationary probability distribution denoted by $\mathbf{q}^+(c)$. In the second phase, all units are free to adjust their state. The free state stationary probability distribution is denoted by $\mathbf{q}^-(c)$. It is a common practice to clamp the input units in the free phase, leaving the hidden and output units free to adjust their state. This practice is specially meaningful in classification problems, so we will adhere to it. The objective of the learning algorithm is to adjust the connection weights so that $q^-(c)$ approaches $q^+(c)$. The difference between both probability distributions is measured by the Kullback-Leibler divergence

$$D\left(\mathbf{q}^+(c), \mathbf{q}^-(c)\right) = \sum_{\mathbf{k}} q^+_{\mathbf{k}}(c) \ln \frac{q^+_{\mathbf{k}}(c)}{q^-_{\mathbf{k}}(c)}$$

where $q^+_{\mathbf{k}}(c)$ and $q^-_{\mathbf{k}}(c)$ are the desired (clamped) and actual (free) probabilities of the visible units being in state $\mathbf{k}$. The learning algorithm can be expressed as finding the weights that minimise D. The probabilistic interpretation of learning in Boltzmann Machines is the search for the log-linear model that best fits the distribution of the data [36, 93, 177, 14]. In the conventional Boltzmann Machine hidden units are introduced to capture high order interactions between variables. The minimisation of the Kullback Leibler divergence is performed

applying gradient descent on the weights. This gradient is of the form

$$\frac{\partial D\left(\mathbf{q}^{+}\left(c\right),\mathbf{q}^{-}\left(c\right)\right)}{\partial w_{ij}}=-\frac{1}{c}\left(p_{ij}^{+}-p_{ij}^{-}\right),$$

where p_{ij}^{+} and p_{ij}^{-} are the probabilities of the connection (u_i, u_j) being activated under the clamped and free stationary distributions respectively:

$$p_{ij}^{+}=\sum_{\mathbf{k}}q_{\mathbf{k}}^{+}\left(c\right)k\left(u_i\right)k\left(u_j\right),$$

$$p_{ij}^{-}=\sum_{\mathbf{k}}q_{\mathbf{k}}^{-}\left(c\right)k\left(u_i\right)k\left(u_j\right).$$

Formal derivation of the learning rule and convergence results can be found in [1]. The gradient descent approach to minimise $D\left(q^{+}\left(c\right),q^{-}\left(c\right)\right)$ is to change the weights according to:

$$\triangle w_{ij}=\alpha\left(\hat{p}_{ij}^{+}-\hat{p}_{ij}^{-}\right),$$

where $\hat{p}_{ij}^{+}$ and $\hat{p}_{ij}^{-}$ are estimations of p_{ij}^{+} and p_{ij}^{-} respectively.

15.2.2 Hidden units versus high order connections

High order interactions involve more than two variables. In classification problems, high order interactions appear when there are joint correlations of several input variables with one output variable.

Hidden units are introduced to model high order interactions between variables. However, the Kullback Leibler divergence for Boltzmann Machines with hidden units is not convex, therefore the learning process can fall in local minima corresponding to suboptimal models. Moreover, the estimation of the connections' activation probabilities is a very delicate step involving the tuning of a number of parameters, such as the appropriate temperature, the appropriate annealing schedule to reach the stationary distribution, and the length of the simulation of the stochastic behaviour for the gathering of statistics over this stationary distribution.

High order connections provide a straightforward model for high order interactions avoiding the need of introducing hidden variables. The advantage of high order connections over hidden units is twofold. First, high order connections can be clearly interpreted as identifying high order interactions, whereas in the case of hidden units this interpretation is more obscure. Second, hidden units introduce spurious interactions, whereas in the case of high order connections all the interactions introduced in the model correspond to high order correlations found the data.

It has been shown [1, 13]that, for Boltzmann Machines without hidden units, the Kullback-Leibler divergence $D\left(q^{+}\left(c\right),q^{-}\left(c\right)\right)$ is a convex function with a single global minimum. Therefore, learning is robust against bad initial conditions and gradient descent ensures reaching the global optimum. Moreover,

the dynamics of the Boltzmann Machine can be simplified, avoiding the need to perform simulated annealing to estimate the clamped and free distribution. In the clamped phase there are no degrees of freedom. The computation of $\hat{p}_{ij}^{+}$ can be done clamping the patterns in order and taking the corresponding statistics. Moreover, this computation can be made once for all at the beginning of the learning process. The absence of hidden units implies that in the free phase the output units are the only degrees of freedom of the system. Restricting the domain of application to classification problems allows further computational simplifications. Assume the convention of codifying the classes with orthogonal binary vectors. Classification of an input pattern is achieved when the output vector is composed of zeroes, with only one unit set to one: the unit that represents the most likely class for the pattern. That implies that the Boltzmann Machine output layer is a "winner-take-all" structure[1]. Once an input pattern is fixed, the asymptotic behaviour of the Boltzmann Machine driven by the simulated annealing algorithm will be to set to one the output unit with the maximum gain. Therefore, the computation of $\hat{p}_{ij}^{-}$ can be done clamping each input pattern, searching for the output unit with the maximum gain (corresponding to the maximum a posteriori probability class) and accumulating the statistics of the activation of the connections. There is no need to explicitly build up the "winner-take-all" structure of the connections of the output layer, because the maximum gain unit can be found by direct search.

15.3 High Order Boltzmann Machines

A High Order Boltzmann Machine with binary units is also described by a triplet (U, L, W), where $U = \{u_i\}$ is the set of binary units, L the set of connections between the units (the network topology) and W are weights associated with the connections. In a High Order Boltzmann Machine (HOBM) a connection $\lambda \in L$ can connect more than two units. Connections are no longer pairs of units but arbitrary subsets of U:

$$\lambda = \left\{u_{i_1}, u_{i_2}, \ldots, u_{i_{|\lambda|}}\right\} \subseteq U.$$

That is, $L \subseteq \mathcal{P}(U)$: The set of connections is a subset of the power set of U. The order of a connection is the number of units connected by it: $O(\lambda) = |\lambda|$. We say that the order of the HOBM is that of the connection with maximum order: $O(U, L, W) = \max\{O(\lambda); \lambda \in L\}$. The topology of a conventional Boltzmann Machine can be visualised as a graph, whereas the topology of the High Order Boltzmann Machine is visualised as an hypergraph [24]. The weights W can be formulated as a mapping that associates each connection with a real number $W : L \to \mathbb{R}$. The consensus function of the HOBM is a straightforward generalisation

[1]Mutually inhibitory connections in the output layer are needed to ensure that the system configuration with the maximum gain output unit set to one corresponds to the global maximum of the consensus function.

of the consensus function of the conventional Boltzmann Machine:

$$C(\mathbf{k}) = \sum_{\lambda \in L} w_\lambda \prod_{u \in \lambda} k(u),$$

where $k(u)$ is the state of unit u in the global configuration $\mathbf{k} \in \{0,1\}^{|U|}$. The HOBM dynamics is the search for the global maximum of the consensus function. The asymptotic distribution of the configurations $\mathbf{q_k}(c)$ follows the Boltzmann distribution based on the consensus function. Learning is the minimisation of the Kullback-Leibler divergence between the distributions of the data and the High Order Boltzmann Machine configurations. The gradient takes the form:

$$\frac{\partial D\left(\mathbf{q}^+(c), \mathbf{q}^-(c)\right)}{\partial w_\lambda} = -\frac{1}{c}\left(p_\lambda^+ - p_\lambda^-\right),$$

where p_λ^+ and p_λ^- are the probabilities of the connection λ being activated under the clamped and free stationary distributions respectively:

$$p_{ij}^+ = \sum_{\mathbf{k}} q_{\mathbf{k}}^+(c) \prod_{\lambda \in L} k(u),$$

$$p_{ij}^- = \sum_{\mathbf{k}} q_{\mathbf{k}}^-(c) \prod_{\lambda \in L} k(u).$$

Formal derivation of the gradient descent learning and convergence properties for the HOBM with binary units (and without hidden units) can be found in[7, 6, 5]. An alternative and more general geometrical proof of the convexity of the learning error for high order neural networks without hidden units can be found in [12].

We have used two weight updating rules. The first is the straightforward application of the gradient descent:

$$\triangle w_\lambda = \alpha\left(\hat{p}_\lambda^+ - \hat{p}_\lambda^-\right),$$

where $\hat{p}_\lambda^+$ and $\hat{p}_\lambda^-$ are estimations of p_λ^+ and p_λ^-, respectively. The learning rate parameter is set to $\alpha = 1$. The second involves a momentum term:

$$\triangle_t w_\lambda = \alpha\left(\hat{p}_\lambda^+ - \hat{p}_\lambda^-\right) + \mu \triangle_{t-1} w_\lambda,$$

with the learning rate set to $\alpha = 1$ as above, and the momentum coefficient set to $\mu = 0.9$.

The estimation of activation probabilities is performed as follows. In the clamped phase, each of the training patterns is set at the input/output units. The activation state of each connection is recorded. (In the binary $\{0,1\}$ case a connection is active if and only if all the extreme units are set to 1). The activation probability of the connections in the clamped phase is computed as the mean activation state of the connections. These clamped probabilities are computed only once at the beginning of the learning process. The free phase is

a series of learning cycles. In each cycle, the input components of each pattern are set on the input units. The response of the network is computed searching for the maximum gain output unit, which is set to 1, while the other output units are set to 0. (Remember we deal only with orthogonal binary output vectors). The activation state of each connection is recorded, and the mean activation state after the presentation of all the training patterns is taken as the activation probability in the free phase. This weight updating schedule is often called batch or off-line adaptation. The weights are updated according to the rule employed and then a new learning cycle is started. The initial weights are always set to zero.

15.3.1 High Order Boltzmann Machines with generalised discrete units

If we consider problems with discrete valued variables, be them categorical or integer valued, we need to introduce discrete units in the fomulation of the Boltzmann Machine adding the specification of the state space of each unit. The Boltzmann Machine with generalised discrete units is described by a quadruple (U, R, L, W) where U, L, W have the same meaning of binary Boltzmann Macines, and $R = \{R_i; i = 1, \ldots, |U|\}$ where R_i is the state space of unit u_i. The configuration space is now the product of the unit state spaces, so that $\mathbf{k} \in R_1 \times R_2 \times \ldots \times R_{|U|}$. The product interpretation of the connection activations is maintained, so that the consensus function preserves its form:

$$C(\mathbf{k}) = \sum_{\lambda \in L} w_\lambda \prod_{u \in \lambda} k(u),$$

taking into account that $k(u_i) \in R_i$. The basic dynamics of the HOBM with discrete units is the search for the global consensus maxima. The Kullback-Leibler divergence between the clamped and free distributions

$$D\left(\mathbf{q}^+(c), \mathbf{q}^-(c)\right) = \sum_{\mathbf{k}} q^+_{\mathbf{k}}(c) \ln \frac{q^+_{\mathbf{k}}(c)}{q^-_{\mathbf{k}}(c)}$$

is well defined taking into account that the configuration space is discrete and finite and that the configuration stationary distributions are well defined as Boltmann distributions of the consensus function. The learning is defined as the minimisation of the Kullback-Leibler divergence, whose gradient can be deduced [120] to be of the following form:

$$\frac{\partial D\left(\mathbf{q}^+(c), \mathbf{q}^-(c)\right)}{\partial w_\lambda} = -\frac{1}{c}\left(a^+_\lambda - a^-_\lambda\right),$$

where a^+_λ and a^-_λ are the mean activations of the connection λ under the clamped and free stationary distributions respectively:

$$a^+_{ij} = \sum_{\mathbf{k}} q^+_{\mathbf{k}}(c) \prod_{\lambda \in L} k(u),$$

$$a_{ij}^{-} = \sum_{\mathbf{k}} q_{\mathbf{k}}^{-}(c) \prod_{\lambda \in L} k(u).$$

A formal derivation of the expression of the gradient and convergence conditions can be found in [120]. The convexity of the Kullback-Leibler for the High Order Boltzmann Machine with discrete state units without hidden units has been proved [120] following the same reasoning employed in [5] for the binary state units. The learning rules used in the experiments reported in this paper are similar to the ones used in the binary case:

$$\triangle w_{\lambda} = \alpha \left(\hat{a}_{\lambda}^{+} - \hat{a}_{\lambda}^{-} \right),$$

where $\hat{a}_{\lambda}^{+}$ and $\hat{a}_{\lambda}^{-}$ are estimations of a_{λ}^{+} and a_{λ}^{-}, respectively. The momentum learning rule is of the form:

$$\triangle_t w_{\lambda} = \alpha \left(\hat{p}_{\lambda}^{+} - \hat{p}_{\lambda}^{-} \right) + \mu \triangle_{t-1} w_{\lambda}.$$

The learning rate and momentum parameters are set to $\alpha = 1$ and $\mu = 0.9$, respectively. The estimation of the mean activation levels is performed in a way similar to the described above for the binary case. The adaptation of the weights is also performed in batch mode. The initial weights were always set to zero.

15.3.2 High Order Boltzmann Machines with Continuous units

The introduction of continuous units does not change much the notation and definitions. The unit state space units is $R_i \subseteq \mathbb{R}$ and the global configuration space is a real high dimensional space $\mathbf{k} \in \mathbb{R}^{|U|}$. The consensus function, that defines the underlying log-linear probabilistic model, has the same expression than in the case of generlized discrete units:

$$C(\mathbf{k}) = \sum_{\lambda \in L} w_{\lambda} \prod_{u \in \lambda} k(u).$$

We do not impose any normalisation restriction on the states, neither we assume any probabilistic or geometric interpretation of the states. The Kullbak-Leibler divergence that drives the learning process is of the form:

$$D\left(\mathbf{q}^{+}(c), \mathbf{q}^{-}(c)\right) = \int_{\mathbf{k}} q_{\mathbf{k}}^{+}(c) \ln \frac{q_{\mathbf{k}}^{+}(c)}{q_{\mathbf{k}}^{-}(c)}.$$

The gradient of the Kullback-Leibler divergence can be shown to be

$$\frac{\partial D\left(\mathbf{q}^{+}(c), \mathbf{q}^{-}(c)\right)}{\partial w_{\lambda}} = -\frac{1}{c}\left(a_{\lambda}^{+} - a_{\lambda}^{-}\right),$$

where a_{λ}^{+} and a_{λ}^{-} are the mean activations of the connection λ under the clamped and free stationary distributions respectively:

$$a_{ij}^{+} = \int_{\mathbf{k}} q_{\mathbf{k}}^{+}(c) \prod_{\lambda \in L} k(u),$$

$$a_{ij}^{-} = \int_{\mathbf{k}} q_{\mathbf{k}}^{-}(c) \prod_{\lambda \in L} k(u).$$

The study of the convexity of the Kullback-Leibler measure, and of the convergence conditions of the learning algorithm, have not been rigorously done in this case. The experimental work reported in this chapter serves as an empirical confirmation of the validity of this generalisation to continuous units. We do not need to tackle with the very difficult problem of realising a simulated annealing in a continuous space, because the input units are always clamped. The estimation of mean activity levels and weight adaptation can be done following the same procedure used in the binary and generalised discrete case. The learning rules are also the same.

15.4 Experimental results

In this section we give a detailed account of the application of HOBM to some selected classification problems described in Appendix . We have applied both binary and generalised discrete unit machines to the Monk's problems. Machines with continuous input units have been applied to the Sonar and Vowel recognition problems. In each case we have tested several high order topologies (without hidden units) to explore the behaviour of the learning algorithm. From a methodological point of view, we use the training and test data sets as provided in the original databases. Training has been performed until either oscillation or saturation on the training set were detected. Oscillation in the training error implies that the topology lacks parameters to fit the problem. Saturation implies that the tested topology has enough (or too many) parameters to fit the data. The tables of results show the percentage of correct classification on both the train and test sets. The results on the train set indicate the kind of stopping of the learning process. The table entries with a percentage of correct classification on the training set well below 100% indicate that the tested topology has not enough parameters to exactly fit the data. In such cases, learning was stopped when oscillations were detected. It must be noted, however, that this does not imply bad generalisation. On the contrary in some cases, the best results on the test set (generalisation) were obtained with topologies that did not saturate on the training set.

15.4.1 HOBM with binary units for the Monk's problems

The set of binary units, used to codify the Monk's problems, is defined as follows:

$$U^{16} = U_1 \cup U_2 \cup U_3 \cup \{u_o\},$$

$$U_1 = \{u_{ij}; i \in \{1,2,4\}, j \in \{1,2,3\}\},$$
$$U_2 = \{u_{ij}; i \in \{3,6\}, j = 1\},$$
$$U_3 = \{u_{ij}; i = 5, j \in \{1,\ldots,4\}\}.$$

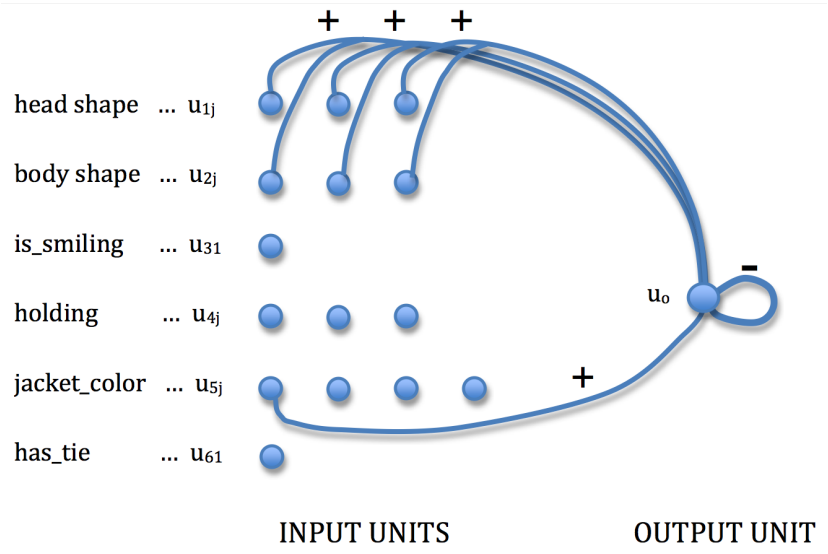


Figure 15.1: A priori topology of the binary Boltzmann Machine for the $\mathbf{M}_1$ problem.

Categorical variables are modelled by a set of binary units. Binary variables are modelled by a single unit. The unit u_o models the classification output. The mapping of the data patterns $\mathbf{x}$ into the states of the machine is as follows:

$$k\left(u_{ij}\right) = \begin{cases} 1 & x_i \text{ takes its } j-\text{th value} \\ 0 & otherwise \end{cases}$$

A priori topologies can be deduced from the logical definition of the problems and the logical interpretation of the connections as extended AND operators. Figure 15.1 shows a graphical representation of the *a priori* topology for the $\mathbf{M}_1$ problem. It consists of three connections of order 3 (with positive weights), one connection of order 2 (with positive weight) and a bias (with negative weight) for the output unit. The formal definition of the *a priori* topologies for the three problems, together with the a priori weights that guarantee their solution, can be found in [120]. The existence of *a priori* topologies tells us two things. First, the HOBM is able to model the problem. Second, they give some hints about the complexity of the network topologies that can be used for each problem. The order of the a priori topology tells us that topologies of lesser order probably will be unable to fit the data. The orders of the a priori topologies are 3, 4 and 3 for $\mathbf{M}_1$, $\mathbf{M}_2$ and $\mathbf{M}_3$, respectively.

The experimental work done on the Monk's problems with binary unit HOBM is an exploration of the sensitivity of the learning algorithm to various topologies. The first experiment was the training of the *a priori* topologies. Subsequent experiments were the training of rather general topologies of increasing order. We call densely connected topology of order $\mathbf{r}$ to an HOBM topology

	$\mathbf{M}_1$			$\mathbf{M}_2$			$\mathbf{M}_3$		
	%train	%test	cycles	%train	%test	cycles	%train	%test	cycles
Best result		100			100			100	
A priori topology	100	100	2	100	97	186	95	100	200
$\mathbf{L}^3$	100	90	12	100	80	128	100	91	30
$\mathbf{L}^4$	100	94	6	100	73	31	100	93	24
$\mathbf{L}^5$	100	88	5	100	71	22	100	91	23
$\mathbf{L}^6$	100	88	5	100	72	22	100	94	17
$\mathbf{L}^7$	100	88	5	100	73	23	100	94	17

Table 15.1: Results with binary Boltzmann Machines for the a priori topologies and densely connected topologies. Weight updating by the simple gradient rule.

	$\mathbf{M}_1$			$\mathbf{M}_2$			$\mathbf{M}_3$		
	%train	%test	cycles	%train	%test	cycles	%train	%test	cycles
Best result		100			100			100	
A priori topology	100	100	3	100	87	66	54	53	200
$\mathbf{L}^3$	100	99	20	100	83	52	100	92	32
$\mathbf{L}^4$	100	92	15	100	73	33	100	90	24
$\mathbf{L}^5$	100	88	15	100	73	24	100	90	23
$\mathbf{L}^6$	100	88	15	100	73	25	100	94	17
$\mathbf{L}^7$	100	87	15	100	73	34	100	94	17

Table 15.2: Results with binary Boltzmann Machines for the a priori topologies and densely connected topologies. Weight updating by the momentum rule.

in which the set of connections contains all the significative connections up to order **r**. Significative connections are those that include the output unit, and that do not connect units representing alternative values of the same pattern component. More formally, the set of connections of the densely connected topology of order **r** is given by:

$$L^r = \left\{ \lambda \subset U^{16} \,|(|\lambda| \leq r) \wedge (u_o \in \lambda) \wedge (u_{ij} \in \lambda \Rightarrow i_{ik} \in \lambda; k \neq j) \right\}.$$

Tables 15.1 and 15.2 summarise the results of the application of the binary unit HOBM to the Monk's problems using the simple gradient rule (Table 15.1) and the momentum rule (Table 15.2). In each case, we give the success on the train and test data, and the number of learning cycles performed. In successive rows we give

- the best result in the reference report [263],

- the results obtained with the *a priori* topology and
- the results with densely connected topologies of increasing order.

Peering through the tables reveals that the number of learning cycles needed is small, therefore learning is fast. In general, the training set was always fitted. All the tried topologies had enough parameters to model the problem. The topologies of order greater than that of the *a priori* topology show clear symptoms of overfitting. The problem $\mathbf{M}_2$ appears as the most difficult, with really poor results on the test data. The cause of this poor behaviour seems to be that all pairs of patterns with Hamilton distance 1 belong to different classes, which makes very difficult the generalisation of the learned distribution when general topologies are used. However, there is an *a priori* topology for this problem that gives very good results. Finally, it is difficult to ascertain the superiority of one of the weight updating rules. For problem $\mathbf{M}_1$ the momentum rule gives excellent results, whereas for problem $\mathbf{M}_3$ it is fairly outperformed by the simple gradient rule. (For the *a priori* topology the momentum rule gets stuck because of the noise in the training set).

15.4.2 HOBM with generalised discrete units for the Monk's problems

The set of units employed to model the patterns is $U = \{u_i; i = 1, \ldots, 6, u_o\}$. The unit state spaces are: $R_1 = R_2 = R_4 = \{0, ..., 2\}$, $R_3 = R_6 = R_o = \{0, 1\}$ and $R_5 = \{0, \ldots, 3\}$. The mapping of the patterns into the unit states is $k(u_i) = j - 1$ if variable x_i=(j-th value in its range).

A priori topologies could be designed searching for the weights that give the consensus maxima for the desired configurations, which could be derived from the logical statement of the problem. We have not found any a priori topologies for problems $\mathbf{M}_1$ and $\mathbf{M}_3$. However, we have found one for problem $\mathbf{M}_2$. In this topology, the positive weight connections are all the connections of order 5 that include the output unit. The negative weight connections are all the connections of order 6 that include the output unit, and the bias connection of the output unit. The formal definition appears in [120]. Figure 15.2 shows a sketched graphical representation of the a priori topology for the $\mathbf{M}_2$ problem using discrete units. In this figure black hexagons represent high order connections and small circles represent units. For clarity, only two of the whole set of connections of order 5 and 6 of each order are shown. Appropriate weights of the connections are 2 for the order 5 connections ($w_{i,j,k,l,o} = 2$ in the figure), -30 for the order 6 connections ($w_{i,j,k,l,m,o} = -30$ in the figure) and $w_o = -1$ for the output bias.

The experimental work is an exploration of the sensitivity of the learning algorithm to various topologies. For $\mathbf{M}_2$ the first experiment was the training of the a priori topology. In the experiments we have tested densely connected topologies of order $\mathbf{r}$. The set of connections in these topologies contains all the connections of order $\mathbf{r}$ or less that include the output unit. A formal definition

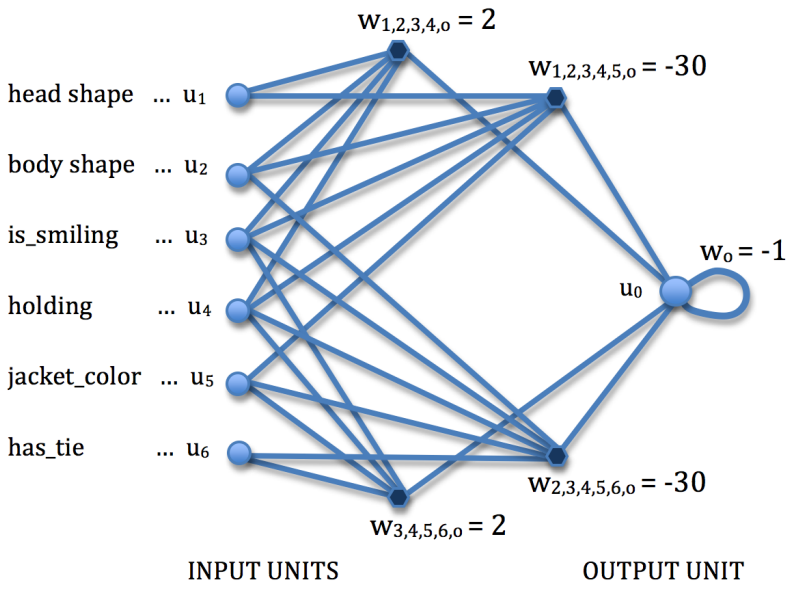


Figure 15.2: Sketch of the a priori topology and weights of a machine with generalised discrete units for the M2 problem.

of the set of connections of the densely connected topology of order **r** is:

$$L^r = \{\lambda \subset U \,|(|\lambda| \leq r) \wedge (u_o \in \lambda)\}$$

Tables 15.3 and 15.4 shows the results for the Monk's problems with High Order Boltzmann Machines that include generalised discrete units using the simple gradient and momentum rules for weight updating. The number of learning cycles is relatively small. Focusing on the problem $\mathbf{M}_2$, the *a priori* topology gives very good results. Relatively good results are obtained with a topology of order 5. Given that the *a priori* topology is of order 6, we did not be expect to obtain good results with topologies of lesser order. In fact, the results on the training set of topologies of order less than 6 (for the $\mathbf{M}_2$ data) show that the topologies are unable to fit the data. For this problem the results of the generalised discrete units improve greatly upon the HOBM with binary units. The results obtained for the $\mathbf{M}_1$ and $\mathbf{M}_3$ problems are bad, but better than expected, given the lack of a known *a priori* topology giving us the certainty that the desired data distribution can be modelled by the machine. For the $\mathbf{M}_1$ and $\mathbf{M}_3$ problems it is quite doubtful that HOBM with generalised discrete units can fit the desired distribution. The appropriateness of applying HOBM with binary or discrete units seems to be a matter of opportunity. The suggestion for applications would be to try first the simplest model (generalised discrete units) and, in case of failure, modelling critical input features with binary units. The learning algorithm remains the same in any case.

	$\mathbf{M}_1$			$\mathbf{M}_2$			$\mathbf{M}_3$		
	%train	%test	cycles	%train	%test	cycles	%train	%test	cycles
Best result		100			100			100	
A priori topology				100	97	20			
$\mathbf{L}^3$	70	67	200	62	66	500	68	71	145
$\mathbf{L}^4$	87	76	163	74	69	500	93	82	133
$\mathbf{L}^5$	91	83	200	98	90	500	98	88	491
$\mathbf{L}^6$	91	84	200	100	91	80	98	85	206
$\mathbf{L}^7$	90	76	200	100	91	80	98	83	238

Table 15.3: Results with Boltzmann Machines that include generalised discrete units for the a priori topologies and densely connected topologies. Weight updating by the simple gradient rule.

	$\mathbf{M}_1$			$\mathbf{M}_2$			$\mathbf{M}_3$		
	%train	%test	cycles	%train	%test	cycles	%train	%test	cycles
Best result		100			100			100	
A priori topology				100	98	40			
$\mathbf{L}^3$	74	73	112	68	67	500	84	81	161
$\mathbf{L}^4$	90	75	134	84	77	500	97	92	233
$\mathbf{L}^5$	95	77	129	100	90	186	98	91	78
$\mathbf{L}^6$	95	77	200	100	91	49	96	84	200
$\mathbf{L}^7$	96	80	106	100	91	49	96	85	200

Table 15.4: Results with Boltzmann Machines that include generalised discrete units for the a priori topologies and densely connected topologies. Weight updating by the momentum rule.

15.4.3 HOBM with continuous [0,1] units for the Sonar problem

The set of units employed to model the variables is $U = \{u_i; i = 1, \ldots, 60, u_o\}$. The unit state spaces are: $R_1 = .. = R_{60} = [0..1]$ and $R_o = \{0, 1\}$. The mapping of the patterns into the unit states is

$$k(u_i) = x_i; i = 1, \ldots, 60,$$

$$k\left(u_o\right) = 1 \text{ if } x_o = \text{metal}$$

The experimental work is an exploration of the sensitivity of the learning to various topologies. No *a priori* topology for this problem can be formulated because there is no logical definition. Two kinds of general topologies were tested:

- The *densely connected topologies* of order **r**, in which the set of connections contains all the connections of order r or less that include the output unit. Formally:
$$L^r = \left\{\lambda \subset U \left| (|\lambda \leq r|) \wedge (u_o \in \lambda) \right.\right\}.$$
- the "in line" topologies, which are densely connected topologies with the additional restriction that the input units in the connection are consecutive. Formally:
$$L_I^r = \left\{\lambda \in L^r \left| (r > 2 \Rightarrow (u_i \in \lambda \Rightarrow (u_{i-1} \in \lambda \vee u_{i+1} \in \lambda))) \right.\right\}.$$

Tables 15.5 and 15.6 show the results obtained for this problem with the simple gradient and momentum rules for weight updating. The results are comparable to those obtained by Gorman and Sejnowski . Sometimes a peak result is followed by a decay of the learning results. When this occurs we have given the peak and the result. The momentum rule shows better convergence (is faster) and better results than the simple gradient rule. The "in line" topologies appear to be well fitted to this problem, probably due to the sequential nature of the data. The correlation between inputs near in time and the situation of the peaks of the signal seem to be the relevant characteristics for the classification of the signals, and they are well captured by the "in line" topologies. It can be concluded that the learning algorithm works well with input continuous units.

15.4.4 HOBM with continuous units for the vowel recognition problem

The set of units for this problem is $U = \{u_i; i = 1, \ldots, 10, u_{oj}; j = 1, \ldots, 11\}$, the input units u_i have state spaces R_i included in the interval $[-5, 5]$, the output units u_{oj} have range $R_{oj} = \{0, 1\}$. The input units take the value of the input component $k(u_i) = x_i$. The mapping of values to the output units makes $k(u_{oj}) = 1$ if x_o takes its j-th value. Again, the experimental work consists of the exploration of the sensitivity of the learning to the topology. The topologies

Topology	cycles	%train	%test
$\mathbf{L}^2$	500	73	69
$\mathbf{L}^3$	447	100	87
$\mathbf{L}^4$	202	100	87
$\mathbf{L}_I^3$	500	86	86
$\mathbf{L}_I^4$	500	89	84
$\mathbf{L}_I^5$	500	88	83
$\mathbf{L}_I^6$	500	91	82
$\mathbf{L}_I^7$	430	93	88
	500	76	71
$\mathbf{L}_I^8$	500	93	86
$\mathbf{L}_I^9$	470	94	89
	500	67	64
$\mathbf{L}_I^{10}$	500	93	87
$\mathbf{L}_I^{11}$	470	93	88
	500	77	73

Table 15.5: Results on the sonar signal recognition using the simple gradient rule.

Topology	cycles	%train	%test
$\mathbf{L}^2$	500	94	76
$\mathbf{L}^3$	77	95	89
$\mathbf{L}^4$	48	99	89
$\mathbf{L}_I^3$	205	97	83
$\mathbf{L}_I^4$	97	100	88
$\mathbf{L}_I^5$	97	100	88
$\mathbf{L}_I^6$	180	98	85
$\mathbf{L}_I^7$	103	100	87
$\mathbf{L}_I^8$	110	97	86
$\mathbf{L}_I^9$	101	99	85
$\mathbf{L}_I^{10}$	136	99	88
$\mathbf{L}_I^{11}$	95	91	82

Table 15.6: Results on the sonar signal recognition using the rule with momentum.

Topology	cycles	%train	%test
$\mathbf{L}^2$	400	48	37
$\mathbf{L}^3$	230	78	52
$\mathbf{L}^4$	100	88	46
$\mathbf{L}^5$	30	89	46
$\mathbf{L}^6$	30	90	41

Table 15.7: Results on the vowel recognition problem. Simple gradient rule

used for the experiments are the *densely connected topologies* of order **r** whose set of connections contain all the connections of order **r** or less with only one output unit in each connection. Formally:

$$L^r = \{\lambda \subset U \,|(|\lambda| \leq r) \wedge (u_{oj} \in \lambda) \wedge (\forall k \neq j \,(u_{ok} \notin \lambda))\}\,.$$

And the *in line* topologies that are densely connected topologies with the additional restriction of the input units being consecutive. Formally:

$$L_I^r = \{\lambda \in L^r \,|(r > 2 \Rightarrow (u_i \in \lambda \Rightarrow (u_{i-1} \in \lambda \vee u_{i+1} \in \lambda)))\}\,.$$

Tables 15.7 and 15.8 show the results of the experiments using the simple gradient and momentum rules. In Table 15.7 we have tested only densely connected topologies, whereas in Table 15.8 we have tested also "in line" topologies. The results are comparable to those given by Robinson . In particular we have even found a better result with the "in line" topology of order 3 and the momentum rule. This result is also comparable to the best reported in [61].

The results in Tables 15.7 and 15.8 show that the proposed learning algorithm for HOBM with continuous input units is quite robust. It performs well even in if the convexity of the Kullback-Leiber distance is doubtful. We always start from zero weights, assuming the convexity of this distance. Bad generalisation (poor results on the test set) seems to be inherent to the data, as the reference works give also poor results. The overfitting effect can be clearly appreciated as the order of the topologies grows. Excellent results are obtained with low order topologies. From the sonar experiment and this one, the momentum rule seems to be more appropriate for continuous inputs.

15.5 Conclusions

HOBM without hidden units applied to classification problems allow for simplifications of the learning process that speed up it by several orders of magnitude, making of practical interest this kind of neural networks. The results obtained are comparable to those found in the reference works, obtained with other techniques or other neural network architectures. We have also found that a small

Topology	cycles	%train	%test
$\mathbf{L}^2$	100	62	43
$\mathbf{L}^3$	80	98	54
$\mathbf{L}^4$	40	96	46
$\mathbf{L}^5$	60	100	45
$\mathbf{L}^6$	50	100	45
$\mathbf{L}_I^3$	120	87	58
$\mathbf{L}_I^4$	120	87	57
$\mathbf{L}_I^5$	120	84	51
$\mathbf{L}_I^6$	250	97	55
$\mathbf{L}_I^7$	250	90	53
$\mathbf{L}_I^8$	250	95	53
$\mathbf{L}_I^9$	150	96	52

Table 15.8: Results on the vowel recognition problem. Momentum rule.

number of learning cycles are needed if the order of topology is high enough (there are enough parameters to model the problem). We have also found that in many cases excellent results are obtained with relatively low order topologies. That means that in most cases of practical interest High Order Boltzmann Machines of moderate size may be of use.

HOBM without hidden units allow the easy generalisation of the learning algorithm to networks with generalised discrete and continuous units. The learning algorithm remains essentially the same, regardless of the kind of units used. The main benefit of the use of generalised discrete units is the reduction of the network complexity, and further speedup of the learning and application processes. The experimental results reported in this chapter show that the effect of the change of codification, from binary to the generalised discrete units, can have quite different effects depending on the problem at hand. In fact, the results can be contrary to the intuition that the use of generalised discrete units implies a loss of modelling power. The results obtained for the $\mathbf{M}_2$ problem show that problems difficult to modelise with binary units can be appropriately modelised with discrete units.

Chapter 16

Relevance Dendritic Computing

This chapter presents the last works performed by the PhD candidate. They represent an open line of research with promising preliminar results. They aim to the enhancement of the so-called Dendritic Computing (DC) in order to obtain improved generalization results. We have embedded it in the Sparse Bayesian framework which leads to the proposition of Relevance Vector Machines (RVM) , therefore we call Relevance Dendritic Computing the resulting approach.

The structure of the chapter is the following. Section 16.1 gives an introductory motivation. Section 16.2 reviews the baseline dendritic approach. Section 16.3 reviews the Sparse Bayesian Learning for linear models. 16.4 defines the Relevance Dendritic Computing. Section 16.5 provides experimental results comparing RDC and RVM over a couple of data sets. Section 16.6 gives our conclusions and avenues for further research.

16.1 Introduction

Dendritic Computing (DC) [21, 226, 228, 229, 230] was introduced as a simple, fast, efficient biologically inspired method to build up classifiers for binary class problems, which could be extended to multiple classes. Notice that DC falls in the general class of Lattice Computing [118] algorithms. Specifically the Single Neuron Lattice model with Dendrite Computation (SNLDC), has been proved to compute a perfect approximation to any data distribution [227, 230]. However it suffers from over-fitting problems: performance of cross-validation experiments is very poor. In a recent work [62] we focused on the results over an specific database that we have studied in previous works [91, 245, 246, 247]. We found that SNLDC showed high sensitivity but very low specificity in a 10-fold cross-validation experiment, with an average low accuracy.

To improve the SNLDC generalization, [21] proposed to compute the optimal rotation of each of the hyperboxes by some optimization method at each step

of the training algorithm. This procedure is computationally very expensive and does not guarantee optimal generalization of classification performance. It depends on the local distribution of the data, as a local kernel transformation whose parameters must be fitted locally. More recently, [62] has proposed the application of a kernel transformation [250] followed by dimension reduction process realized by the Lattice Independent Component Analysis (LICA) [119] as an appropriate feature extraction for SNLDC classifiers, improving over other feature extraction processes. The composite transformation is the Kernel-LICA approach for SNLDC.

Sparse Bayesian Learning [264, 265, 266] is a general Bayesian framework for obtaining sparse solutions to regression and classification tasks. This approach obtains dramatically simpler models than other approaches. A popular instance of this approach is the Relevance Vector Machine (RVM), which is trains a prediction model that is functionally identical to the one used by the Support Vector Machines (SVM) but obtaining much parsimonious representations, i.e. using much less relevant vectors than the equivalent performance SVM. The RVM and SVM underlying model is linear, though it can benefit from the kernel trick to obtain solutions to non-linear problems. Following the notation introduced in [265], in supervised learning we are given a sample of input vectors $\{\mathbf{x}_n\}_{n=1}^N$ along with corresponding targets $\{t_n\}_{n=1}^N$ which might be real values or class labels. The conventional prediction model is given by a linear function of the form:

$$y(\mathbf{x};\mathbf{w}) = \sum_{i=1}^{M} w_i \psi_i(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}), \tag{16.1}$$

where the output is a weighted sum of M basis functions. Basis functions are fixed and might be non-linear, without affecting the intrinsic linear nature of the model.

16.2 Dendritic Computing

A single layer morphological neuron endowed with dendrite computation based on lattice algebra was introduced in [230]. Figure 16.1 illustrates the structure of a single output class single layer Dendritic Computing system, where D_j denotes the dendrite with associated inhibitory and excitatory weights $\left(w_{ij}^0, w_{ij}^1\right)$ from the synapses coming from the i-th input neuron. Assume that we are given a collection of m pairs of patterns and class labels $\left(\mathbf{x}^\xi, c_\xi\right)$, $\mathbf{x}^\xi \in \mathbb{R}^d, c_\xi \in \{0,1\}$. The response of the j-th dendrite to the ξ-th input vector is as follows:

$$\tau_j\left(\mathbf{x}^\xi\right) = p_j \bigwedge_{i \in I_j} \bigwedge_{l \in L_{ij}} (-1)^{1-l} \left(x_i^\xi + w_{ij}^l\right), \tag{16.2}$$

where $l \in L_{ij} \subseteq \{0,1\}$ identifies the existence and inhibitory/excitatory character of the weight, $L_{ij} = \emptyset$ means that there is no synapse from the i-th input

neuron to the j-th dendrite; $p_j \in \{-1, 1\}$ encodes the inhibitory/excitatory response of the dendrite. The complete neuron activation is computed as:

$$\tau\left(\mathbf{x}^{\xi}\right) = \bigwedge_{k=1}^{j} \tau_k\left(\mathbf{x}^{\xi}\right);\ \xi = 1, \ldots, m. \tag{16.3}$$

To obtain the output classification prediction, a hard-limiter or Heaviside function is applied:

$$\hat{c}^{\xi} = f\left(\tau\left(\mathbf{x}^{\xi}\right)\right),$$

where

$$f(x) = \begin{cases} 1 & x > 0 \\ 0 & x \leq 0 \end{cases}.$$

We could also apply a logistic sigmoid function to the activation funcion:

$$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}, \tag{16.4}$$

which coud be interpreted as the posterior probability of class 1.

It has been shown [230] that classification models based on dendritic computation have powerful approximation properties. In fact, they showed that this model is able to approximate any compact region in higher dimensional Euclidean space to within any desired degree of accuracy. However, it must be noted that the neuron activation function of equation (16.3) has not derivatives defined. Therefore, it is not possible to apply gradient based approaches to develop learning algorithms.

A constructive algorithm was provided in [230], which is specified in Algorithm 16.1. The algorithm starts building a hyperbox enclosing all pattern samples of class 1, that is, $C_1 = \{\xi : c_{\xi} = 1\}$. Then, the dendrites are added to the structure trying to remove misclassified patterns of class 0 that fall inside this hyperbox. In step 6 the algorithm selects at random one such misclassified patterns, computes the minimum Chebyshev distance to a class 1 pattern and uses the patterns that are at this distance from the misclassified pattern to build a hyperbox that is removed from the C_1 initial hyperbox. In this process, if one of the bounds is not defined, $L_{ij} \neq \{0, 1\}$, then the box spans to infinity in this dimension.

16.3 Sparse Bayesian Learning for linear models

The general linear model of equation (16.1) is specialized to the function:

$$y(\mathbf{x}; \mathbf{w}) = \sum_{i=1}^{N} w_i K(\mathbf{x}, \mathbf{x}_i) + w_0, \tag{16.5}$$

where $K(\mathbf{x}, \mathbf{x}_i)$ is a kernel function which defines a basis function from each training set sample. This model is used in the Relevance Vector Machines

Algorithm 16.1 Dendritic Computing learning based on elimination

Training set $T = \left\{ \left(\mathbf{x}^{\xi}, c_{\xi} \right) \mathbf{x}^{\xi} \in \mathbb{R}^{d}, c_{\xi} \in \{0,1\}; \xi = 1, \ldots, m \right\}$,

1. Initialize $j = 1$, $I_j = \{1, \ldots d\}$, $P_j = \{1, \ldots, m\}$, $L_{ij} = \{0,1\}$,

$$w_{ij}^{1} = - \bigwedge_{c_{\xi}=1} x_i^{\xi};\ w_{ij}^{0} = - \bigvee_{c_{\xi}=1} x_i^{\xi},\ \forall i \in I$$

2. Compute response of the current dendrite D_j, with $p_j = (-1)^{\mathrm{sgn}(j-1)}$:

$$\tau_j \left(\mathbf{x}^{\xi} \right) = p_j \bigwedge_{i \in I_j} \bigwedge_{l \in L_{ij}} (-1)^{1-l} \left(x_i^{\xi} + w_{ij}^{l} \right),\ \forall \xi \in P_j.$$

3. Compute the total response of the neuron:

$$\tau \left(\mathbf{x}^{\xi} \right) = \bigwedge_{k=1}^{j} \tau_k \left(\mathbf{x}^{\xi} \right);\ \xi = 1, \ldots, m.$$

4. If $\forall \xi \left(f \left(\tau \left(\mathbf{x}^{\xi} \right) \right) = c_{\xi} \right)$ the algorithm stops here with perfect classification of the training set.

5. Create a new dendrite $j = j + 1$, $I_j = I' = X = E = H = \emptyset$, $D = C_1$

6. Select $\mathbf{x}^{\gamma}$ such that $c_{\gamma} = 0$ and $f \left(\tau \left(\mathbf{x}^{\xi} \right) \right) = 1$.

7. $\mu = \bigwedge_{\xi \neq \gamma} \left\{ \bigvee_{i=1}^{d} \left| x_i^{\gamma} - x_i^{\xi} \right| : \xi \in D \right\}$.

8. $I' = \left\{ i : \left| x_i^{\gamma} - x_i^{\xi} \right| = \mu, \xi \in D \right\}$; $X = \left\{ \left(i, x_i^{\xi} \right) : \left| x_i^{\gamma} - x_i^{\xi} \right| = \mu, \xi \in D \right\}$.

9. $\forall \left(i, x_i^{\xi} \right) \in X$

   (a) if $x_i^{\gamma} > x_i^{\xi}$ then $w_{ij}^{1} = -x_i^{\xi}$, $E_{ij} = \{1\}$

   (b) if $x_i^{\gamma} < x_i^{\xi}$ then $w_{ij}^{0} = -x_i^{\xi}$, $H_{ij} = \{0\}$

10. $I_j = I_j \bigcup I'$; $L_{ij} = E_{ij} \bigcup H_{ij}$

11. $D' = \left\{ \xi \in D : \forall i \in I_j, -w_{ij}^{1} < x_i^{\xi} < -w_{ij}^{0} \right\}$. If $D' = \emptyset$ then goto step 2, else $D = D'$ goto step 7.

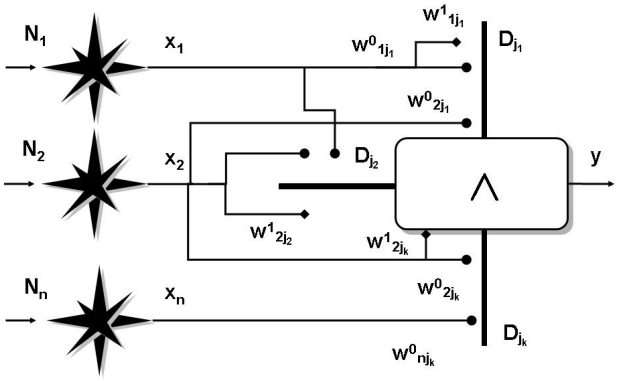


Figure 16.1: A single ouput single layer Dendritic Computing system.

(RVM). We have a classification problem, where $\{\mathbf{x}_n, t_n\}_{n=1}^N$ are the training input-target class pairs, $t_n \in \{0, 1\}$. The logistic function of equation (16.4) is applied to the linear model to obtain a prediction of the probability of class 1, then, adopting a Benouilli distribution for $P(t\,|\mathbf{x})$ we write the training set likelihood as:

$$P(\mathbf{t}\,|\mathbf{w}) = \prod_{n=1}^{N} \sigma\left(y\left(\mathbf{x}_n; \mathbf{w}\right)\right)^{t_n} \left[1 - \sigma\left(y\left(\mathbf{x}_n; \mathbf{w}\right)\right)\right]^{1-t_n}. \tag{16.6}$$

A prior distribution on the weights is introduced to model our assumptions or preferences, the most popular is the zero-mean Gaussian prior distribution over $\mathbf{w}$:

$$p(\mathbf{w}\,|\boldsymbol{\alpha}) = \prod_{i=0}^{N} \mathcal{N}\left(w_i\,|0, \alpha_i^{-1}\right), \tag{16.7}$$

where $\boldsymbol{\alpha}$ is the vector of hyperparameters moderating the strength of the prior. An infinite value of the hyperprior implies that the weight is certainly zero valued. Hyperprior parameters are a measure of weight relevance. The distribution of the hyperparameters are conventionally [265] assumed to be Gamma distributions:

$$p(\boldsymbol{\alpha}) = \prod_{i=0}^{N} \text{Gamma}\left(\alpha_i\,|a, b\right), \tag{16.8}$$

where setting $a = b = 0$ we obtain non-informative hyperpriors with uniform (flat) distributions.

The estimation of the model parameters, learning, corresponds to the computation of the posterior distribution over all unknowns given the data $p(\mathbf{w}, \boldsymbol{\alpha}\,|\mathbf{t})$,

which can be done decomposing it as follows:

$$p\left(\mathbf{w},\boldsymbol{\alpha}\left|\mathbf{t}\right.\right)=p\left(\mathbf{w}\left|\mathbf{t},\boldsymbol{\alpha}\right.\right)p\left(\boldsymbol{\alpha}\left|\mathbf{t}\right.\right), \tag{16.9}$$

where

$$p\left(\mathbf{w}\left|\mathbf{t},\boldsymbol{\alpha}\right.\right)=\frac{p\left(\mathbf{t}\left|\mathbf{w}\right.\right)p\left(\mathbf{w}\left|\boldsymbol{\alpha}\right.\right)}{p\left(\mathbf{t}\left|\boldsymbol{\alpha}\right.\right)}\propto p\left(\mathbf{t}\left|\mathbf{w}\right.\right)p\left(\mathbf{w}\left|\boldsymbol{\alpha}\right.\right),$$

can not computed in closed form[1], but can be approximated by a Gaussian by the Laplace's method. First, find the 'most probable' weights $\mathbf{w}_{MP}$ for fixed hyperparameters $\boldsymbol{\alpha}$. This is done finding the maximum over $\mathbf{w}$ of the log of the un-normalized posterior:

$$\log p\left(\mathbf{w}\left|\boldsymbol{\alpha},\mathbf{t}\right.\right)\propto\log\left\{p\left(\mathbf{t}\left|\mathbf{w}\right.\right)p\left(\mathbf{w}\left|\boldsymbol{\alpha}\right.\right)\right\}\quad=$$

$$=\sum_{n=1}^{N}\left[t_n\log y_n+\left(1-t_n\right)\log\left(1-y_n\right)\right]-\frac{1}{2}\mathbf{w}^T\mathbf{A}\mathbf{w}, \tag{16.10}$$

where $y_n=\sigma\left(y\left(\mathbf{x}_n;\mathbf{w}\right)\right)$, $\mathbf{A}=\mathrm{diag}\left(\alpha_0,\alpha_1,\ldots,\alpha_N\right)$ is a diagonal matrix of hyperparameters. Laplace's method is a quadratic approximation of the log-posterior around its mode that approaximates it by a Gaussian centered at the mode, allowing to compute the Hessian matrix required for the Newton minimization method:

$$\nabla^2_{\mathbf{w}}\log p\left(\mathbf{w}\left|\boldsymbol{\alpha},\mathbf{t}\right.\right)\Big|_{\mathbf{w}_{MP}}=-\left(\boldsymbol{\Phi}^T\mathbf{B}\boldsymbol{\Phi}+\mathbf{A}\right), \tag{16.11}$$

where $\mathbf{B}=\mathrm{diag}\left(\beta_1,\beta_2,\ldots,\beta_N\right)$ is a diagonal matrix with $\beta_n=y_n\left(1-y_n\right)$, and $\boldsymbol{\Phi}$ is the $N\times\left(N+1\right)$ design matrix $\boldsymbol{\Phi}=\left[\boldsymbol{\Phi}\left(\mathbf{x}_1\right),\boldsymbol{\Phi}\left(\mathbf{x}_2\right),\ldots,\boldsymbol{\Phi}\left(\mathbf{x}_N\right)\right]$, with $\boldsymbol{\Phi}\left(\mathbf{x}_n\right)=\left[1,K\left(\mathbf{x},\mathbf{x}_1\right),K\left(\mathbf{x},\mathbf{x}_2\right),\ldots,K\left(\mathbf{x},\mathbf{x}_N\right)\right]^T$. The posterior covariance matrix and mean provided by the Laplace's method approximation is:

$$\boldsymbol{\Sigma}=\left(\boldsymbol{\Phi}^T\mathbf{B}\boldsymbol{\Phi}+\mathbf{A}\right)^{-1}, \tag{16.12}$$

$$\boldsymbol{\mu}=\mathbf{w}_{MP}=\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\mathbf{B}\mathbf{t}. \tag{16.13}$$

The hyperparameter posterior distribution $p\left(\boldsymbol{\alpha}\left|\mathbf{t}\right.\right)$ is conveniently approximated by a delta function at the most probable value $\boldsymbol{\alpha}_{MP}=\arg\max_{\alpha}\left\{p\left(\boldsymbol{\alpha}\left|\mathbf{t}\right.\right)\right\}$, assuming that this point estimate is *representative* of the posterior distribution. Relevance learning is the search for this hyperparameter posterior mode, maximizing

$$p\left(\boldsymbol{\alpha}\left|\mathbf{t}\right.\right)\propto p\left(\mathbf{t}\left|\boldsymbol{\alpha}\right.\right)p\left(\boldsymbol{\alpha}\right),$$

with respect to $\boldsymbol{\alpha}$. The iterative re-estimation [265] of the hyperparameters is given by the following equation:

$$\alpha_i^{\mathrm{new}}=\frac{\gamma_i}{\mu_i^2}, \tag{16.14}$$

[1]Because we can not compute $p\left(\mathbf{t}\left|\boldsymbol{\alpha}\right.\right)$.

where μ_i is the i-th posterior mean weight from equation (16.13), and

$$\gamma_i = 1 - \alpha_i \Sigma_{ii},$$

with Σ_{ii} the i-th diagonal element of the covariance matrix of equation (16.12), computed with the current $\boldsymbol{\alpha}$ values. The values $\gamma_i \in [0, 1]$ are interpreted as the measure of well determination of the weight w_i from the data. When $\gamma_i \approx 0$ then $\Sigma_{ii} \approx \alpha_i^{-1}$, α_i will be large and the weight will be highly constrained by the prior. When α_i is small and w_i fits the data, then $\gamma_i \approx 1$.

Starting from arbitrary uniform hyperparameter values, the learning algorithm proceeds by computing the most probable weights $\mathbf{w}_{MP}$ by minimization of the log-posterior using the Hessian matrix provided by Laplace's method for Newton's method. Then, posterior mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ are computed according to equations (16.13) and (16.12). The hyperparameters are updated by equation (16.14), and the process is repeated until some convergence condition is reached.

16.4 Relevance Dendritic Computing

We start discussing the application of the Sparse Bayesian Learning to the dendritic model of equation (16.3) by making one observation: although the weights in equation (16.3) appear to be defined as something unrelated to the data, in fact in Algorithm 16.1 the weights correspond to selected components of data input vectors. This observation allows us to rewrite the dendritic neuron activation in a pattern more similar to the linear model function of equation (16.5):

$$\tau(\mathbf{x}) = \bigwedge_{n=1}^{N} \lambda_n(\mathbf{x}, \mathbf{x}_n), \tag{16.15}$$

where $\lambda_k(\mathbf{x}, \mathbf{x}_n)$ assumes the role of a lattice-based kernel function which we need to define appropriately to fit into the pattern of equations (16.5) and (16.2). First notice that each vector component can be in different dendrites, meaning that (1) a lattice kernel function is not equivalent to a dendrite, and (2) that we must account for the p_j parameter in our model. Second, notice that the component of the input vector can be either in an excitatory/inhibitory synapse or absent from the equations. The absence is modelled by an infinite value which is the null element of the minimum operator. The lattice kernel of the n-th training sample is, therefore, defined as follows:

$$\lambda_n(\mathbf{x}, \mathbf{x}_n) = \bigwedge_{i=1}^{d} (x_i - x_{n,i})\, \pi_{n.i}, \tag{16.16}$$

where x_i and $x_{n,i}$ are the i-th components of vectors $\mathbf{x}$ and $\mathbf{x}_n$, respectively. The factor $\pi_{n.i} \in \{-1, 1, \infty\}$ models the contribution of the i-th component of the n-th sample training vector to the neural activation function, constituting the model weights to be learnt $\{\pi_{n,i}\} = \boldsymbol{\pi}$. Notice that the number of model

parameters is $N \times d$. A value $\pi_{n.i} = \infty$ means that the i-th component of the n-th sample training vector is irrelevant. When a component is irrelevant for all sample training data, then the whole dimension is irrelevant for the classification of the input. This change in the value null element of the underlying algebra implies that the Gaussian prior distributions of the weights as specified in equation (16.7) must be formulated over the inverses of the weights $\mathbf{w} \equiv \left\{\pi_{n,i}^{-1}\right\} = 1/\boldsymbol{\pi}$:

$$p\left(\mathbf{w}\,|\boldsymbol{\alpha}\right) = \prod_{n=1}^{N}\prod_{i=1}^{d}\mathcal{N}\left(\pi_{n,i}^{-1}\,|0, \alpha_{n,i}^{-1}\right).$$

Some questions may be posed on the model of equation (16.15), namely:

- It is always meaningful? It may be possible that some configurations of the parameters' values π correspond to expressions which are not computable? i.e. active weights may define intervals where the upper limit lies below the lower limit. If so, how can they be detected or avoided by search algorithms?

- The expression in equation (16.15), it is always equivalent to a dendritic model defined according to equation (16.3)? If not, how can we decide if the expression is a proper dendritic model? It seems that equation (16.15) is more general than equation (16.3). Can we prove that?

We assume the log-posterior of equation (16.10) to estimate both the model weights and their relevance. It does not require any specific probabilistic distribution of the model output, therefore is general enough to cover the dendritic neuron activation of equation (16.15). However, the log-posterior is not derivable respect to the weights. Therefore, the search for the "most probable" weights must be carried out by some Monte-Carlo method that will allow also to estimate the mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ of the posterior distribution in order to apply the hyperparameter estimation of equation (16.14). We do not need to compute the estimation of the full covariance matrix $\boldsymbol{\Sigma}$, because only the elements in the diagonal are useful for the estimation of the relevance hyperparameters, according to equation (16.14). Algorithm 16.2 summarizes the estimation process. We assume non-informative hyperpriors, therefore the hyperparameters are initialized as uniformly distributed random variables.

The search for the most probable weights $\boldsymbol{\pi}_{MP}$ must be done by a Monte-Carlo method, which, in essence, consists of the random generation of alternative parameter configurations and the (probabilistic) selection of the alternative configurations that improve the value of the log-posterior. This search is performed for fixed hyperparameters $\boldsymbol{\alpha}$. The Monte-Carlo method specified in Algorithm 16.3 is a variation of Simulated Annealing [164]. The random generation of the alternative configuration of parameter values $\boldsymbol{\pi}'(k)$ is performed selecting a subset of parameters and randomly changing their values. The temperature T allows to control the nature of the algorithm, which becomes a greedy local search for low temperatures. The process generates a sequence of parameter

Algorithm 16.2 The Relevance Dendritic Computing

1. Initialize uniform weight prior hyperparameters $\alpha_{n,i} = \frac{1}{Nd}$.

2. Search for the most probable weights $\boldsymbol{\pi}_{MP}$ minimizing the log-posterior of equation (16.10)

$$\log p\left(\boldsymbol{\pi}\,|\boldsymbol{\alpha}, \mathbf{t}\,\right) \propto \log\left\{p\left(\mathbf{t}\,|\boldsymbol{\pi}\,\right) p\left(\boldsymbol{\pi}\,|\boldsymbol{\alpha}\,\right)\right\}$$

$$= \sum_{n=1}^{N} \left[t_n \log y_n + (1 - t_n) \log\left(1 - y_n\right)\right] - \frac{1}{2}\mathbf{w}^T \mathbf{A}\mathbf{w},$$

   with $y_n = \sigma\left(\tau\left(\mathbf{x}_n\right)\right)$, by Monte-Carlo Methods. Obtain relevant estimations of $\boldsymbol{\Sigma}$ and $\boldsymbol{\mu}$ from Monte-Carlo generated data.

3. Apply relevance updating

$$\alpha_{n,i}^{\text{new}} = \frac{\gamma_{n,i}}{\mu_{n,i}^2},$$

   with

$$\gamma_{n,i} = 1 - \alpha_{ni}\Sigma_{ni,ni},$$

4. Remove irrelevant weights ($\alpha_{n,i} > \theta$) setting them to infinity

5. Test convergence. If not converged, repeat from step 2.

Algorithm 16.3 A Monte-carlo method for the maximization of the log-posterior.

Initialize randomly $\boldsymbol{\pi}(0)$
Set the initial temperature $T(0)$
$k = 0$
Repeat

- generate a random candidate configuration $\boldsymbol{\pi}'(k)$
- compute $E'(k) = \log p(\boldsymbol{\pi}(k), \boldsymbol{\alpha}|\mathbf{t})$
- $\triangle E = E'(k) - E(k)$
- compute $P_a(\triangle E, T) = e^{\triangle E/T}$, generate random $r \sim \mathcal{U}(0,1)$
- if $\triangle E > 0$ or $P_a(\triangle E, T) > r$ then $\boldsymbol{\pi}(k+1) = \boldsymbol{\pi}'(k)$; $E(k+1) = E(k)$.
- reduce T

Until convergence

values assignments that can be assumed as a sample of the posterior distribution, therefore we can use them to produce empirical estimations of the mean and covariance matrix of the parameters, specifically the diagonal elements.

16.5 Experimental results

For comparison we use the first version of SparseBayes code provided by Tipping[2] to train the RVM. It is more powerful than the later version, though it has difficulties with large databases. Our implementation of RDC has follow the same template. We are publishing our code in our research group site[3]. In the implementation used to obtain the results below we have used the estimated distribution mode instead of the mean in step 3 of Algorithm 16.2. The number of state transitions attempted in the chain is 100 times the number of actual relevant parameters, meaning that the chains become shorter as the algorithm discards irrelevant parameters, and the process speeds up.

The first experiment has been done on the demo two class data set provided by Tipping, originally was proposed by Ripley [225]. The data were designed as mixtures of normal distributions to have a best-possible error rate about 8%. We have used the same train-test partition proposed by Tipping, 250 training vectors of two dimensions and 1000 test vectors. Both classes have the same representation in both train and test partitions. Gaussian kernel and hyper-parameters for RVM are set as suggested by him. Table 16.1 summarizes the results on the test partition of the data set for both RVM and RDC. Notice that

[2] http://www.miketipping.com/index.php
[3] http://www.ehu.es/ccwintco/index.php/Relevance_Dendritic_Computing:_codes_and_examples

	accuracy	sensitivity	specificity	#rel. par.
RDC	0.89	0.86	0.92	2
RVM	0.90	0.87	0.92	6

Table 16.1: Results of RDC and RVM on the Rippley dataset

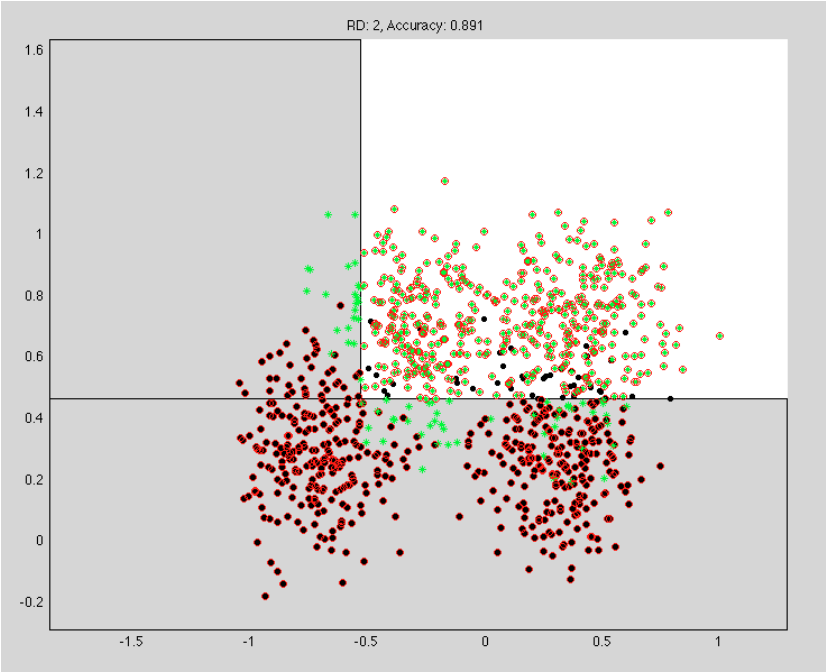


Figure 16.2: Visualization of the data space partition obtained by RDC

the relevant parameters of RDC are unidimensional components of train data vectors, while for RVM relevant parameters are train data vectors. Therefore, the model achieved by RDC is more parsimonious than the RVM model. The difference between both approaches can be better appreciated examining the visualization of the data space partition induced by RDC, shown in Figure 16.2, versus the visualization of the decision boundary obtained by the RVM, shown in Figure 16.3. In those figures, green points correspond to class 1 samples, black points to class 0 samples. Correctly classified test samples are enclosed in red circles. The RDC parsimonious solution obtains a square assigned to class 1, defined by two thresholds, each on one dimension. The RVM solution involves the linear combination of the kernel transformations of several relevant vectors. It is surprising that such different solutions computed over the train data set may give almost the same performance on the test data set.

The second experiment has been done on a specific data set produced in our

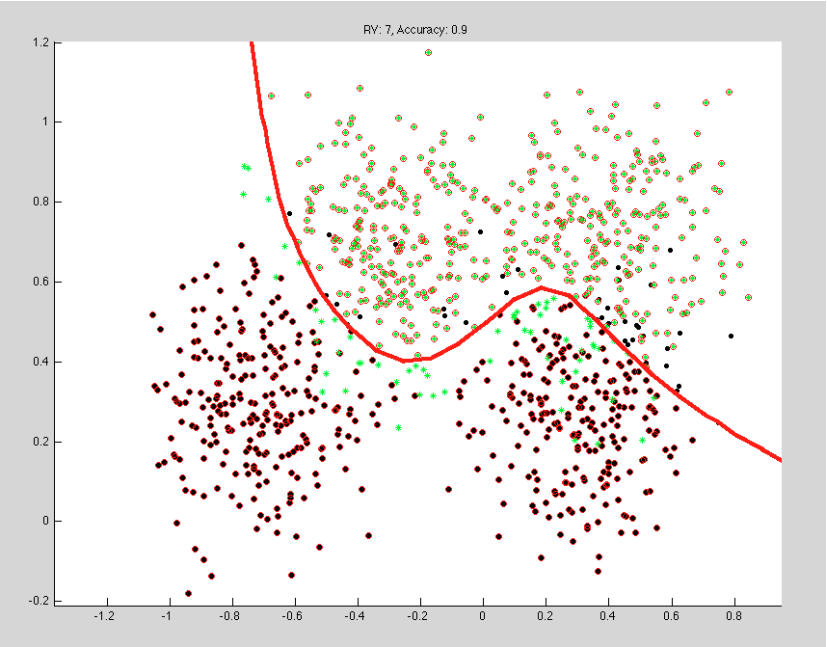


Figure 16.3: Visualization of the decision boundary obtained by RVM

research about machine learning application to Alzheimer's Disease computer aided diagnosis [129]. The aim was to obtain discriminant features from scalar measures of Diffusion Tensor Imaging (DTI) data, Fractional Anisotropy (FA) and Mean Diffusivity (MD), and to train and test classifiers able to discriminate Alzheimer's Disease (AD) patients from controls on the basis of features extracted from the FA or MD volumes. Feature selection was done computing the Pearson's correlation between FA or MD values at voxel site across subjects and the indicative variable specifying the subject class. Voxel sites with high absolute correlation are selected for feature extraction. The original dataset comes from an on-going study in Hospital de Santiago Apostol collecting anatomical T1-weighted MRI volumes and DTI data from healthy control subjects and AD patients. Specifically we have tested the RDC and RVM over the set of FA features obtained applying a 99.5 percentile on the Pearson's correlation ditribution, giving 121 image features ($d = 121$) for each subject either patient of control. We have performed a 10-fold crossvalidation experiment. Table 16.2 shows the summary of average results over the 10-folds. Again RDC performs close to RVM with much more sparse models.

FA features and a linear SVM classifier achieve perfect accuracy, sensitivity and specificity in a several cross-validation studies, supporting the usefulness of DTI-derived features as an image-marker for AD and to the feasibility of building Computer Aided Diagnosis systems for AD based on them.

	accuracy	sensitivity	specificity	#rel. par.
RDC	0.87	0.93	0.82	1
RVM	0.88	0.95	0.84	2.24

Table 16.2: Results of RDC and RVM on the FA features for AD patients and healthy controls dataset

16.6 Conclusions

We are concerned with the enhancement of generalization capabilities of single layer neuron model endowed with Dendritic Computing. We have examined the Sparse Bayesian Learning paradigm for linear models. In this chapter we we apply the Sparse Bayesian Learning framework in order to detect the most relevant data samples that provide greater generalization with the most parsimonious model, which we call Relevance Dendritic Computing. We propose a reformulation of the dendritic model in order to fit into the Sparse Bayesian Learning paradigm, tailoring the general Bayesian computation scheme to it. Strictly compared with RVM, RDC advantage is that it does not compute any matrix inversion, therefore it has no numerical conditioning problems. On the other hand, RDC needs to perform a Monte Carlo search and estimation on a very flat energy landscape, with atypical discrete valued parameters. We have performed comparative experiments against the RVM on two data sets. RDC finds comparable results with much more parsimonious models than RVM. We find these results encouraging to pursue further works in this topic, including exploration of alternative Monte Carlo sampling and optimization methods.

Part V

Appendices

Appendix A

Continuation and Graduated Non-Convexity Methods

Most times the error function minimized by the learning algorithms is a strongly non-linear and non-convex, meaning that solutions found are highly dependent on the initial conditions and the numerical method employed to find the minima of the error function. One of the perspectives considered along the work on this thesis has been to put the learning algorithms in the framework of numerical methods developed elsewhere to tackle with non-linear difficult problems. We review in this appendix two such approaches: Continuation and Graduated Non-Convexity methods. Both categories of methods are intimately related, but their development and language come from diverse disciplined and we have not found any attempt to harmonize them, therefore we will present them as separate approaches.

A.1 Continuation methods: motivation

Many computational problems can be stated as the search for solutions of non-linear systems. Linear systems either have one solution or an infinite number of equivalent solutions. Ill-posed linear systems of equations can be solved by least squares approaches, obtaining the best solution in terms of minimal error. However, non-linear systems may have many non-equivalent solutions, and the solutions found can be strongly dependent on the initial state of the resolution process. The problem of finding the best solution regardless of the initial condition is equivalent to the general problem of global minimization of non-convex functions. We review the basics of homotopy and numerical continuation methods, which we believe are patterns underlying many computational approaches in the Computational Intelligence domain. The main reference followed in this appendix is [291]. More on Homotopy can be found in [182, 281]. Continuation methods are presented in [10, 132, 224, 278, 280]. Some historical accounts can be found in [208, 278, 280].

A.2 Continuation methods as homotopies

We seek the solution of nonlinear systems of n variables and n equations identifying the points $\mathbf{x} = (x_1, x_2, ..., x_n)$ that solve the system. One general approach is to start with another systems of equations of which we already know the solution. This simplified system is "bent" into the original system. The mathematical equivalent to physical bending is the *homotopy* concept. The homotopy-based no-linear system solving approach starts with a simple system that is specially chosen for having an easy solution, i.e. is a linear system. Then, sets up a system of equations with an additional variable t, called the *homotopy parameter*, that yields the simple system at $t = 0$, while at $t = 1$ it yields the original non-linear system. To obtain the solution of the original system ir follows a path $\mathbf{x}^*(t)$ of the system's solution from $t = 0$ to $t = 1$, thereby solving the original problem.

Formally: Let $\mathbb{R}^n$ denote the Euclidean n-dimensional space. A vector function $\mathbf{F} : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ means that both $\mathbf{F}(\mathbf{x})$ and $\mathbf{x}$ have n components, i.e. $\mathbf{F}(\mathbf{x}) = (F_1(\mathbf{x}), ..., F_n(\mathbf{x}))$ and $\mathbf{x} = (x_1, ..., x_n)$, respectively, i.e. $\mathbf{F}(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^n$. We desire to solve the $n \times n$ system of nonlinear equations $\mathbf{F}(\mathbf{x}) = \mathbf{0}$, obtaining a solution $\mathbf{x}^\star = (x_1^\star, ..., x_n^\star)$. The homotopy approach to obtain $\mathbf{x}^\star$ is as follows.

1. First define a vector function $\mathbf{E} : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ such that $\mathbf{x}^0$ solves the system of equations $\mathbf{E}(\mathbf{x}) = \mathbf{0}$.

2. Define a special function, called a *homotopy function*, $\mathbf{H}(\mathbf{x}, t) : \mathbb{R}^{n+1} \rightarrow \mathbb{R}^n$ which has the original n variables plus an extra one, denoted t. Here $(\mathbf{x}, t) = (x_1, ..., x_n, t) \in \mathbb{R}^{n+1}$. The homotopy function $\mathbf{H}$ must be constructed so that $\mathbf{H}(\mathbf{x}, 0) = \mathbf{E}(\mathbf{x})$ and $\mathbf{H}(\mathbf{x}, 1) = \mathbf{F}(\mathbf{x})$. It follows that at $t = 0$, $\mathbf{H}(\mathbf{x}, 0) = \mathbf{0}$ has a known solution $\mathbf{x}^0$, while at $t = 1$, $\mathbf{H}(\mathbf{x}, t) = \mathbf{0}$ has solution $\mathbf{x}^\star$.

3. The continuation process starts at $\mathbf{x}(0) = \mathbf{x}^0$, the solution of $\mathbf{H}(\mathbf{x}, 0)$, and then solves $\mathbf{H}(\mathbf{x}, t)$ for increasing t until reaching $\mathbf{x}(1) = \mathbf{x}^\star$. In general, the solution obtained at each value of t will be helpful to compute the solution at the next step of the path follwing iteration. That is, the sequence of solutions $\mathbf{x}(t)$ specifies a path that we can follow from $t = 0$ to $t = 1$, thereby obtaining the solution $\mathbf{x}^\star$ of the original system $\mathbf{F}(\mathbf{x}) = \mathbf{0}$.

A.2.1 Varieties of homotopies

The shape of the path that will depend directly upon the homotopy function $\mathbf{H}(\mathbf{x}, t)$ selected. There are thre varieties of homotopies most frequent in the literature:

- *Newton homotopy*: $\mathbf{H}(\mathbf{x}, t) = \mathbf{F}(\mathbf{x}) - (1 - t)\mathbf{F}(\mathbf{x}^0)$. A simple method to start with. Pick an arbitrary point $\mathbf{x}^0$. Next, calculate $\mathbf{F}\left(\mathbf{x}^0\right)$, and then let $\mathbf{E}(\mathbf{x}) = \mathbf{F}(\mathbf{x}) - \mathbf{F}\left(\mathbf{x}^0\right)$.

- *Fixed-point homotopy*: $\mathbf{H}(\mathbf{x},t) = (1-t)(\mathbf{x}-\mathbf{x}^0)+t\mathbf{F}(\mathbf{x})$. As in the former type we can pick an arbitrary point $\mathbf{x}^0$ and $\mathbf{E}(\mathbf{x}) = \mathbf{x}-\mathbf{x}^0 = \mathbf{0}$.

- *Linear homotopy*: $\mathbf{H}(\mathbf{x},t) = t\mathbf{F}(\mathbf{x})-(1-t)\mathbf{E}(\mathbf{x}) = \mathbf{E}(\mathbf{x})+t\left[\mathbf{F}(\mathbf{x})-\mathbf{E}(\mathbf{x})\right]$ that subsumes the previous two choosing the appropiate function $\mathbf{E}(\mathbf{x})$.

A.2.2 Existence of solution paths

Given a homotopy function $\mathbf{H}:\mathbb{R}^{n+1}\longrightarrow\mathbb{R}^n$ we would like to obtains conditions for the existence of solution paths. Define $\mathbf{H}^{-1} = \{(\mathbf{x},t)\,|\mathbf{H}(\mathbf{x},t)=\mathbf{0}\}$ as the set of all solutions $(\mathbf{x},t)\in\mathbb{R}^{n+1}$ to the system $\mathbf{H}(\mathbf{x},t)=\mathbf{0}$. This set can consist of one or more paths but also may contain points not belonging to any particular path configuration. Figure A.1 illustrates the various possible distributions of points of $\mathbf{H}^{-1}$.

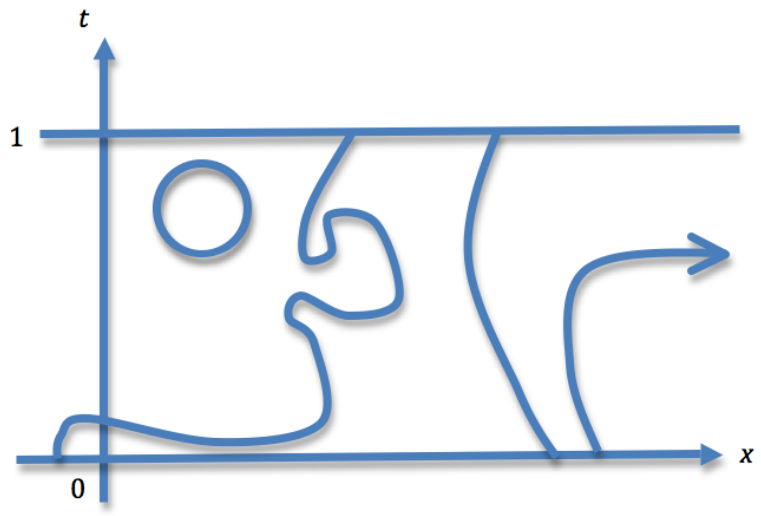
Denote the solutions for t fixed as the set $\mathbf{H}^{-1}(t) = \{\mathbf{x}\,|\mathbf{H}(\mathbf{x},t)=\mathbf{0}\}$. Therefore, $\mathbf{H}^{-1}(0)$ consists of all the start points $\mathbf{x}(0)$, and, similarly, $\mathbf{H}^{-1}(1)$ consists of all the solutions $\mathbf{x}^\star = \mathbf{x}(1)$.

The *implicit function theorem* gives some conditions ensuring that $\mathbf{H}^{-1}$ consists solely of solution paths. The Jacobian of the homotopy function, $\mathbf{H}'$ can be split conveniently into two parts $\mathbf{H}'(\mathbf{x},t) = \left(\mathbf{H}'_x(\mathbf{x},t), \frac{\partial\mathbf{H}}{\partial t}\right)$, where $\mathbf{H}'_x(\mathbf{x},t)$ is the $n\times n$ matrix composed of the derivatives $\frac{\partial\mathbf{H}_i}{\partial x_j}$ and $\frac{\partial\mathbf{H}}{\partial t}$ is the column vector composed of the derivatives $\frac{\partial H_i}{\partial t}$. Denote $\mathbf{H}'_x(\bar{\mathbf{x}},\bar{t})$ the Jacobian computed at point $(\bar{\mathbf{x}},\bar{t})\in\mathbf{H}^{-1}$.

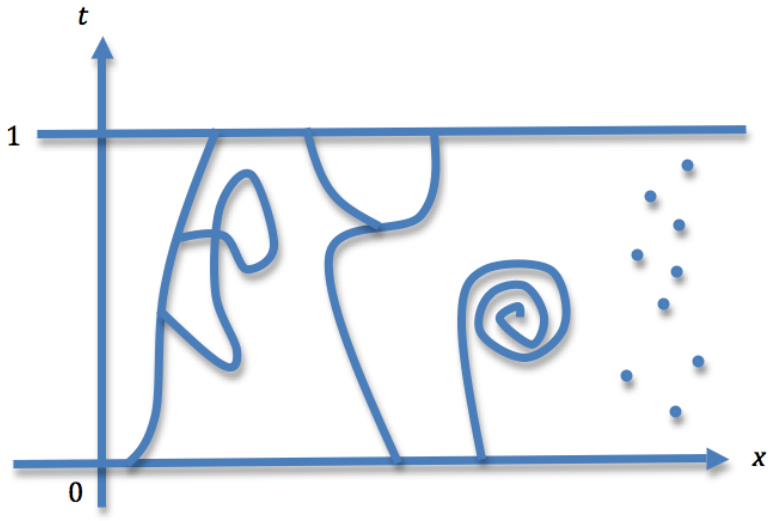
Theorem 7. *Implicit function theorem: Let* $\mathbf{H}(\mathbf{x},t):\mathbb{R}^{n+1}\longrightarrow\mathbb{R}^n$ *be continuously differentiable,* $(\bar{\mathbf{x}},\bar{t})\in\mathbf{H}^{-1}$ *and* $\mathbf{H}'_x(\bar{\mathbf{x}},\bar{t})$ *be invertible. Then in a neighborhood of* $(\bar{\mathbf{x}},\bar{t})$ *all points* $(\mathbf{x},t)$ *that satisfy* $\mathbf{H}(\mathbf{x},t)=\mathbf{0}$ *are on a single continuously differentiable path through* $(\bar{\mathbf{x}},\bar{t})$.

That is, if the local linearization of $\mathbf{H}$ is invertible everywhere, the points $(\mathbf{x},t)\in\mathbf{H}^{-1}$ are over curves that may be locally linearly approximated, that is, they lie on a unique, continuosly differentiable path. This means that no splittings, bifurcations, forks, crossings, infinite endless spirals as shown in Figure A.1(b) can occur. Next, define the partial Jacobian $\mathbf{H}'_{-i}$ to be the $n\times n$ matrix formed from $\mathbf{H}'$ by deleting the i-th column. The implicit function theorem requires that $\mathbf{H}'_{-n+1}$ is invertible. But actually the theorem holds as long as any $\mathbf{H}'_{-i}$ is invertible. If one or several of the different $\mathbf{H}'_{-i}$ are invertible the implicit function theorem guarantees that just one solution path exists passing through $(\bar{\mathbf{x}},\bar{t})$. Suppose that the Jacobian matrix $\mathbf{H}'(\bar{\mathbf{x}},\bar{t})$ is full rank. Then, we can always find some partial invertible Jacobian matrix $\mathbf{H}'_{-i}(\bar{\mathbf{x}},\bar{t})$. Then we can state a theorem setting the conditions to ensure that a path from starting points up to the final solutions exists.

Theorem 8. *Path theorem: Let* $\mathbf{H}:\mathbb{R}^{n+1}\longrightarrow\mathbb{R}^n$ *be continuously differentiable and suppose that for every* $(\bar{\mathbf{x}},\bar{t})\in\mathbf{H}^{-1}$ *the Jacobian* $\mathbf{H}'(\bar{\mathbf{x}},\bar{t})$ *is of full rank. Then* $\mathbf{H}^{-1}$ *consists only of continuously differentiable paths.*



(a)



(b)

Figure A.1: Some possible distributions of the solutions to the homotopy system of equations (a) $\mathbf{H}^{-1}$ consisting only of finite length paths, some of them not connecting the starting points and the final solutions. (b) $\mathbf{H}^{-1}$ includes isolated points, bifurcations, infinite spirals.

A.2.3 Motion along a path

A homotopy function $\mathbf{H}$ it is *regular* if $\mathbf{H}'(\mathbf{y})$ is of full rank for all $\mathbf{y} \in \mathbf{H}^{-1}$. Therefore, regularity ensures that $\mathbf{H}^{-1}$ consists of continuosly differentiable paths. We denote $\mathbf{y}(p) = (\mathbf{x}(p), t(p))$ for $\mathbf{y}(p) \in \mathbf{H}^{-1}$, where p is the distance moved along the path, the arclength of the solution curve from the initial solution $\mathbf{x}(0)$ up to $\mathbf{x}(p)$. To determine a path $\left\{\mathbf{y}(p) \in \mathbf{H}^{-1}\right\}$ it is possible to specify a set of *basic differential equations* (BDE) such that starting from an initial state $\mathbf{y}(0)$, the points in the path are obtained by integration of the BDE. Specify $\dot{y}_i \equiv \dot{y}_i(p)$ as $\dot{y}_i = \frac{dy_i}{dp}, i = 1, ..., n+1$. We have that $\mathbf{H}(\mathbf{y}(p)) = \mathbf{0}$, differentiating both sides yields, $\sum_{i=1}^{n+1} \frac{\partial \mathbf{H}}{\partial y_i} \dot{y}_i = \mathbf{0}$, where $\partial \mathbf{H} / \partial y_i$ is a column of the Jacobian $\mathbf{H}'$, so that

$$\mathbf{H}'(\mathbf{y})\dot{\mathbf{y}} = \mathbf{0}$$

is a system of n linear equations in $n+1$ unknowns $\dot{y}_i$. The $\dot{\mathbf{y}}$ that satifies this system yields the *basic differential equations* (BDE):

$$\dot{y}_i = (-1)^i \det \mathbf{H}'_{-i}(\mathbf{y}), \quad i = 1, ..., n+1 \tag{A.1}$$

Theorem 9. *BDE theorem: Let* $\mathbf{H} : \mathbb{R}^{n+1} \longrightarrow \mathbb{R}^n, \mathbf{H} \in C^2$, *be a regular homotopy. Given a starting point* $\mathbf{y}^0$ *in* $\mathbf{H}^{-1}$, *the solution of the basic differential equations* (A.1), *starting from* $\mathbf{y}(p^0) = \mathbf{y}^0$, *is unique and determines a path in* $\mathbf{H}^{-1}$.

The theorem states that solving the BDE for $\mathbf{y}(p)$ as p varies will permit us to follow a path in $\mathbf{H}^{-1}$ and thereby obtain a solution point $\mathbf{x}^1 \in \mathbf{H}^{-1}(1)$. There are a full variety of algorithms to follow a path and obtain a solution, however the BDE yield far more than merely a means to solve an equation system. They can provide immediate and important information on the direction of the path.

A.3 Path-Following Algorithms

There are two fundamental types of numerical algorithms to follow a path in $\mathbf{H}^{-1}$:

- Simplicial algorithms (Piecewise-linear methods)
- Differential equation approaches, including the Predictor-corrector methods.

A.3.1 Simplicial algorithms

The algorithm used will generate its own path that approximates an underlying path in $\mathbf{H}^{-1}$. Before utilizing any algorithm, there is an initial phase which consist on select a homotopy that is appropiate to solve the specific problem and ensure that following a path in $\mathbf{H}^{-1}$ does indeed lead to a solution to the problem.

Simplicial algorithms operate as follows. First they take a simplex s^0 and generate the line segment across s^0. Then they take an adjacent simplex s^1 and generate a line segment across s^1. By adjacent, we mean that s^0 and s^1 share a commmon facet and lie on opposite sides of the facet. The line segment across s^1 will connect with the line segment across s^0. Then they take a third simplex, s^2, where s^2 is adjacent to s^1, and generate the line segment across s^2. Also, the line segment across s^2 will connect with the line segment across s^1. In this manner, simplex and line segment by line segment, a path of line segments is generate. On each simplex a new approximation $\mathbf{G}$ is made of $\mathbf{H}$, and the line segment is generated by solving $\mathbf{G}(\mathbf{w}) = \mathbf{0}$ on that simplex. If the simplices are small, we get a good approximation to a path in $\mathbf{H}^{-1}$. The path is *piecewise linear* because it is linear on each simplex. The main difference among simplicial algorithms is in how the new vertex is selected on each step, although they share the basic step of linearly crossing the simplices generated

A.3.2 Differential equation approaches

Path following can be made solving the BDE. The simplest differential equation solution procedure is the Euler method. Unfortunately, it tends to drift off the path, so to overcome that, other methods are introduced. The Newton method does not follow the path but instead tries to leap ahead to a solution. Predictor-corrector methods apply corrections to get back into the path.

Euler's Method Euler's method immediately applies to the BDE

$$\dot{y}_i = (-1)^i \det \mathbf{H}'_{-i}(\mathbf{y}), \quad i = 1, ..., n+1,$$

where $\mathbf{y}(0) = \mathbf{y}^0$ is given. We can determined a differential equation method to following the path,

$$\dot{y_i^{k+1}} = y_i^k + \left(p^{k+1} - p^k\right)(-1)^i \det \mathbf{H}'_{-i}(\mathbf{y}^k), \quad i = 1, ..., n+1,$$

where $0 < p^1 < p^2 < ... < p^k < p^{k+1} < ...$

Homotopy diffential equations (HDE) The BDE were obtained by differentiating $\mathbf{H}(\mathbf{x}(p), t(p)) = \mathbf{0}$ with respect to p. A new system, called the *homotopy differential equations* (HDE), is obtained by differentiating $\mathbf{H}(\mathbf{x}(t), t) = \mathbf{0}$ with respect to t. This system

$$\frac{d\mathbf{x}}{dt} = -\left[\mathbf{H}'_x\right]^{-1} \mathbf{H}'_t,$$

holds whenever $\mathbf{H}'_x$ is of full rank. Solving de HDE will give a path in $\mathbf{H}^{-1}$ since de HDE were obtained directly form the homotopy equation itself.

If $\mathbf{H}$ is regular we may write $\mathbf{y} = (\mathbf{x}, t)$ as a function of p and obtain the BDE. Under the slightly stronger assumption that $\mathbf{H}'_x$ is of full rank we may write $\mathbf{x}$ as a function of t and obtain the HDE. The two alternative differential equation

Algorithm A.1 Predictor-Corrector method

Initialization

Let $\epsilon_1 > \epsilon_2 > 0$ be given, and set $\mathbf{y}^0 = \mathbf{y}(0)$, $k = 0$.

Predictor

Given $\mathbf{y}^k$, take a predictor step from $\mathbf{y}^k$ to obtain $\mathbf{y}^{k+1}$.
Replace k by $k+1$.
If $\left\|\mathbf{H}\left(\mathbf{y}^k\right)\right\| < \epsilon_1$, repeat the predictor step, otherwise, go to the corrector step.

Corrector

Let $\mathbf{b} \in \mathbb{R}^{n+1}$ describe a hyperplane $\mathbf{b}\left(\mathbf{y} - \mathbf{y}^k\right) = \mathbf{0}$, which intersects the path $\mathbf{y}(p)$ at a nearby point $\mathbf{y}\left(p^k\right)$ further along the path. Setting $\mathbf{y}^k = \mathbf{y}^{k,0}$ compute

$$\mathbf{y}^{k,l+1} = \mathbf{y}^{k,l} - \begin{pmatrix} \mathbf{H}'\left(\mathbf{y}^{k,l}\right) \\ \mathbf{b} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{H}\left(\mathbf{y}^{k,l}\right) \\ 0 \end{pmatrix}, l = 0, 1, ...$$

Stop when $\left\|\mathbf{H}\left(\mathbf{y}^{k,l+1}\right)\right\| < \epsilon_2$.

If $\mathbf{y}^{k,l+1}$ is near $t = 1$, terminate. Otherwise, let $\mathbf{y}^{k+1} = \mathbf{y}^{k,l+1}$ and go to the predictor step with $k+1$ replacing k.

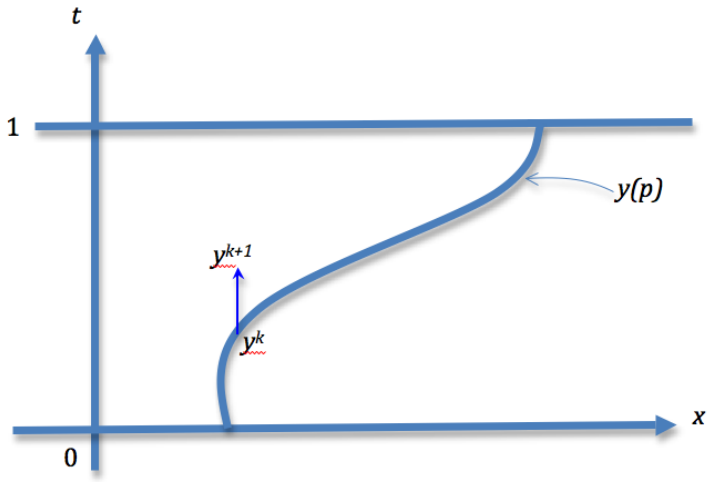
systems were derived from the same homotopy and they must generate the same path. The BDE, HDE and any other differential equation can be solved by the Euler approach.

A.3.3 Predictor-Corrector Methods

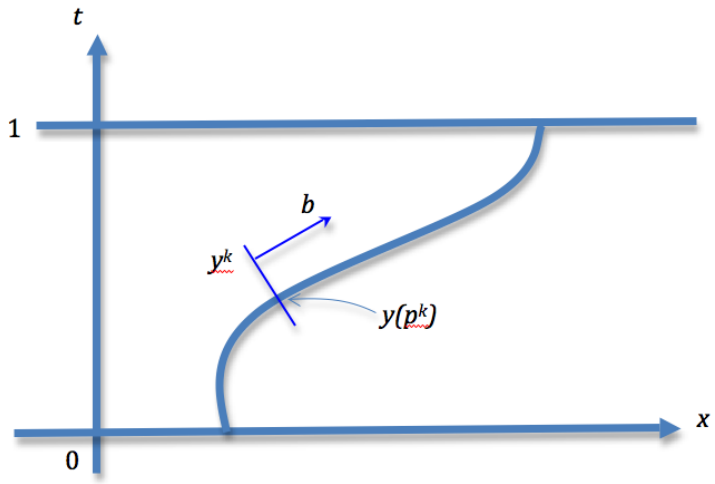
In Predictor-corrector methods, the Euler, or any related method, serves as the predictor. It predicts where the path is going, and we move in that direction. After several steps the Euler can drift off the path. The Newton method is then called upon as a corrector to get us back close to the path. The sequence is followed iteratively. The method is described in Algorithm A.1. Figure A.2 illustrates the behavior of the predictor corrector method.

A.4 Graduated Non-Convexity

Graduated Non-Convexity (GNC) algorithm was introduced by A. Blake & A. Zisserman [38, 40] as a deterministic annealing method for approximating the global solution for nonconvex minimization of unconstrained, continuous problems. GNC algorithm has been presented as a discrete continuation method. [223, 80, 37, 45, 50, 46, 233, 74, 206, 51, 259]. The book by Blake & Zisserman



(a)



(b)

Figure A.2: Predictor-corrector integration method for solution continuation: (a) Predictor step. (b) Corrector step

[40] address the problem of image reconstruction, presenting it as a piecewise continuous reconstruction of the data with weak continuity constraints, that is, a constraint that can be broken occasionally. The problem is formulated as an energy minimization problem. The energy function is decomposed into a reconstruction error and constraint terms penalizing the lack of continuity of the solution. The energy function is non-convex with many local minima. Any method used to minimize the energy function needs to avoid falling into these local minima. Stochastic methods avoid local minima performing random jumps. Under certain conditions their convergence to the global minimum is guaranteed, although the amount of computation required may be very large [94]. The GNC algorithm constructs a convex approximation to the energy function, which is deformed gradually from its convex form to its target form during the process of approximating the global solution using a gradient-based algorithm. GNC finds good solutions with much less cost than stochastic simulated annealing [39, 23, 181].

Formally, GNC is based on a convex approximation $F^{(1)}(\mathbf{x})$ to the original non-convex energy $F(\mathbf{x})$. A family of functions $F^{(p)}(\mathbf{x})$, $p \in [0,1]$ is defined such that

- $F^{(1)}(\mathbf{x})$ is convex,
- $F^{(0)}(\mathbf{x}) = F(\mathbf{x})$, and
- $F^{(p)}(\mathbf{x})$ varies continuously in a particular prescribed manner as p decreases from 1 to 0.
- There is a finite discrete sequence of parameter values $1 = p_0 > p_1 > ... > p_{n-1} > p_n = 0$.

The GNC algorithm performs the minimization of the whole sequence of functions $F^{(p)}$, one after the other, using the optimal vector $\mathbf{x}^*$ result of the minization of optimisation as the initial condition for the next. There are numerous ways to minimise each $F^{(p)}(\mathbf{x})$, including direct descent and gradient methods. Figure A.3 illustrates the process showing how the minimum found for the initial convex approximation is continued to the minimum of the non-convex function $F(\mathbf{x})$.

In general, there is no guarantee that this method obtains the global minimum, but, if the energy function does not have too many local minima, the result can be quite satisfactory, but sub-optimal [40, 80, 23, 205]. Extensive experiments have shown that for a finite number of iterations GNC leads to minimizers having a lower (hence better) energy than SA [39].

A.5 GNC for line and surface reconstruction

Surface reconstruction is a very ill-posed inverse problem so a regularization must be introduced in its formulation. It will serve in this appendix as an illustration of the GNC approach. The regularization is done adding a *smoothness*

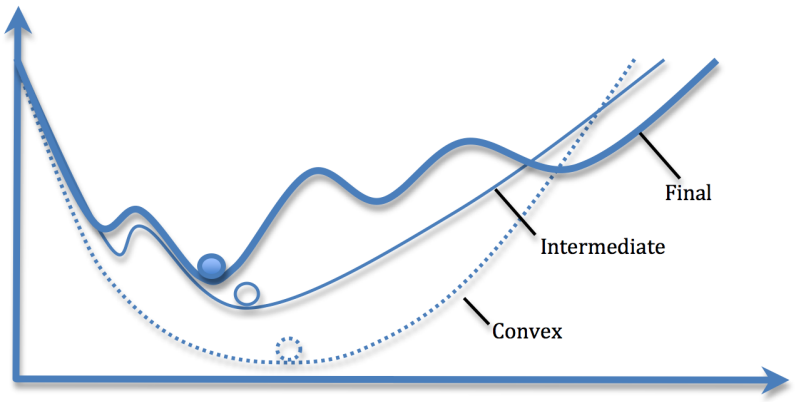


Figure A.3: The evolution of the energy functions minimized during GNC.

term which incorporates some *a priori* knowledge on the expected properties of the solution. The solution is found by minimizing a weighted combination of the original problem with the *smoothness term*. The principal drawback of this approach is that discontinuities between homogeneous regions of the image are not adequately recovered if the adopted smoothness term not satisfies properties of concavity and finite asymptotic behaviour.

Incorporation of discontinuities into the reconstructed surface by means of weak continuity constraints was originally suggested in [38]. In 1D, the *weak string* nicely models this class of signals as a locally Gaussian, non-stationnary first-order Markov chain with a Boolean non-interacting line process. The weak string preserves discontinuities without any prior information about their existence or location.

The illustrative problem is the detection of step discontinuities in 1D data. The aim is to construct a piecewise smooth 1D function $u(x)$ which is a good fit to some data $d(x)$. The problem of finding $u(x)$ is a minimization problem of the energy associated to the weak string. The behaviour of the string over an interval $x \in [0, N]$ is defined by its energy, which is a sum of three components:

- D: a measure of faithfulness to data: $D = \int_0^N (u-d)^2 \, dx$, that is of the form $D = \sum_{i=0}^N (u_i - d_i)^2$ for discrete functions.
- S: a measure of how severely the function $u(x)$ is deformed: $S = \lambda^2 \int_0^N u'^2 dx$, which has the form $S = \lambda^2 \sum_{i=1}^N (u_i - u_{i-1})^2 (1 - l_i)$ for discrete functions.
- P: the sum of penalties α levied for each break (discontinuity) in the string, which is of the form $P = \alpha \sum_{i=1}^N l_i$, where l_i is a so-called "line-process". It is defined such that each l_i is a boolean-valued variable, if $l_i = 1$ there is a discontinuity or if $l_i = 0$ indicates continuity in $x \in [i-1, i]$.

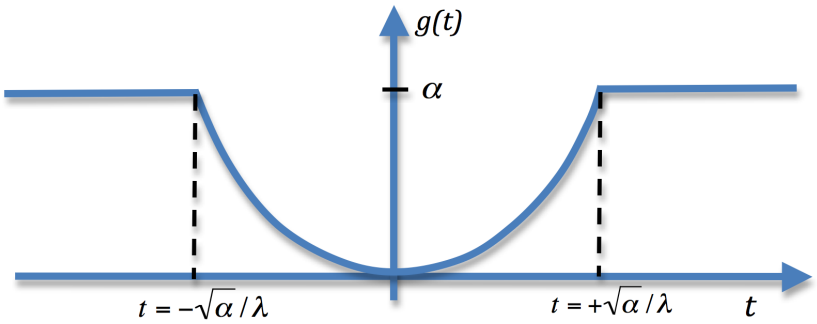


Figure A.4: Energy of interaction between neighbours after minimization over $l \in \{0, 1\}$.

The problem is to minimize the total energy:

$$E = D + S + P$$

finding, for a given $d(x)$, the function $u(x)$ for which the total energy E is smallest. There isn't any interaction between line variables. The problem is stated as:

$$(u_i^*, l_i^*) = \arg \min_{\{u_i, l_i\}} E.$$

Discontinuities are incorporated into the functions. However, the minimization over the $\{l_i\}$ can be done in advance and thus the problem reduces simply to a minimization over the $\{u_i\}$:

$$u_i^* = \arg \min_{\{u_i\}} F.$$

where

$$F = D + \sum_{i=1}^{N} g_{\alpha,\lambda} \left(u_i - u_{i-1} \right).$$

The neighbour interaction function g is defined as

$$g_{\alpha,\lambda}(t) = \begin{cases} \lambda^2 (t)^2 & if\ |t| > \sqrt{\alpha}/\lambda \\ 0 & otherwise \end{cases}.$$

Figure A.4 presents a plot of the neighbour interaction function. The absence of boolean variables allows the application of the GNC algorithm.

Function F is not convex, with many local minima. The goal of the weak string computation is to find the global minimum of F. In the GNC method, the cost function F is first approximated by a new function $F^\star$ which is convex and hence can only have one local minimum, which must also be its global minimum. Then, define the whole sequence of cost functions $F^{(p)}$, for $1 \geq p \geq 0$, where

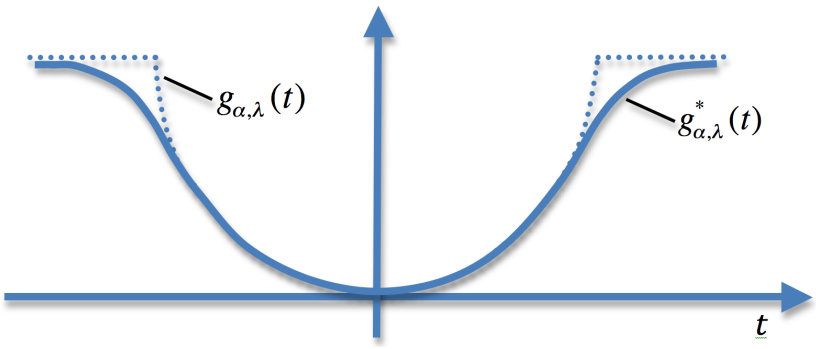


Figure A.5: Plot of function $g^{\star}$

p is the non-convexity parameter. Chose $F^{(0)} = F$, the true cost function, and $F^{(1)} = F^{\star}$, the convex approximation to F. In between, $F^{(p)}$ changes, in a continuous fashion, between $F^{(1)}$ and $F^{(0)}$. The GNC algorithm finds the minima of the whole sequence of $F^{(p)}$, one after the other, using the result of one optimisation as the starting point for the next. See Figure A.3. The initial convex function used is

$$F^{\star} = D + \sum_{i=1}^{N} g^{\star}\left(u_i - u_{i-1}\right)$$

by constructing an appropriate neighbour interaction function $g^{\star}$. Blake & Zisserman [40] proposes a way to construct this function $g^{\star}$:

$$g^{\star}_{\alpha,\lambda}(t) = \begin{cases} \lambda^2 (t)^2 & if\ |t| < q \\ \alpha - c^{\star}\left(|t| - r\right)^2/2 & if\ q \le |t| < r \\ \alpha & if\ |t| \ge r \end{cases}$$

where $r^2 = \alpha\left(\frac{2}{c^{\star}} + \frac{1}{\lambda^2}\right)$, and $q = \frac{\alpha}{\lambda^2 r}$. The parameter $c^{\star}$ is a scalar constant and is chosen to satisfy the conditions imposed during the construction of $g^{\star}$ to make it as close as possible to g (see [40] for more information). For example, for the weak string has calculated a value of $c^{\star} = 1/2$.

The one-parameter family of cost functions $F^{(p)}$ is defined as:

$$F^{(p)} = D + \sum_{i=1}^{N} g^{(p)}\left(u_i - u_{i-1}\right)$$

with

$$g^{(p)}_{\alpha,\lambda}(t) = \begin{cases} \lambda^2 (t)^2 & if\ |t| < q \\ \alpha - c\left(|t| - r\right)^2/2 & if\ q \le |t| < r \\ \alpha & if\ |t| \ge r \end{cases}$$

where $c = \frac{c^{\star}}{p}$, $r^2 = \alpha\left(\frac{2}{c} + \frac{1}{\lambda^2}\right)$, and $q = \frac{\alpha}{\lambda^2 r}$.

Appendix B

Vector Quantization Bayesian Filtering

The Vector Quantization Bayesian Filtering (VQBF) is applied in the main part of the thesis as a filtering method with good properties. Though the main contributions of the thesis are related to the algorithms for the computation of the optimal Vector Quantizer, we feel that a proper treatment of the VQBF approach is worthwile.

Section B.1 introduces the notation for this appendix. Section B.2 introduces the block model around the pixels. Section B.3 introduces the complete model for VQBF. Section B.4 provides some theoretical reasoning about the reasons for border preservation in VQBF. Finally, in Section B.5 presents the conclusions of the appendix.

B.1 Notation

We start recalling standard notation of Bayesian image processing [**?**], [282]. An image is described a tuple $x = \left(x^P, x^L, x^E, ..\right)$ whose components correspond to interesting features for the application at hand: image intensity x^P, pixel class labels x^L, edge positions x^E, etc. Let be S^P a finite square grid where each site represents a pixel position in the image domain. Vector $x^P = \left(x_s^P\right)_{s \in S^P}$ represents a pattern of configurations of the pixel grayscale values. If we try to classify pixels into some thematic map then $x^L = \left(x_s^L\right)_{s \in S^L}$ is a pattern of labels associated to the blocks of pixels in set S^L, so that $x_s^L = l \in L$ is the class label of image block s. Image blocks may overlap.

The observable data y are a function Y of the true image x. We denote $\mathbf{Y}$ and $\mathbf{X}$ the space of observable data and true images, respectively. Given $x \in X$, the probalistic model of Y is the likelihood $\mathsf{P}\left(y \left| x \right.\right)$, the probability of observing y when the true image is x. Therefore, for each $x \in \mathbf{X}$, $\mathsf{P}\left(y \left| x \right.\right)$ is a probability distribution over $\mathbf{Y}$. i.e. $\mathsf{P}\left(y \left| x \right.\right) \geq 0$ and $\sum_y \mathsf{P}\left(y \left| x \right.\right) = 1$.

The *a priori* expectations on the image can be formulated as constraints defined on the true image. The normalized and positive function $\Pi(x)$ defined on the image space $\mathbf{X}$ is the *a priori* distribution. The choice of *a priori* model depends on the problem, being a key step of the whole image Bayesian analysis. The *a priori* distribution Π and the likelihood distributions determine the joint image and data distribution in the product space $\mathbf{X} \times \mathbf{Y}$:

$$\mathsf{P}(x, y) = \Pi(x)\,\mathsf{P}(y\,|x), x \in \mathbf{X}, y \in \mathbf{Y}. \tag{B.1}$$

This is the probabilistic model of the pair of random variable (X, Y) defined in $\mathbf{X} \times \mathbf{Y}$ where X follows Π, and Y follows Γ, given by $\mathrm{d}\,\Gamma(Y = y) = \sum_x \mathsf{P}(x, y)$. The *a posteriori* probability of $x \in \mathbf{X}$ is given by

$$\mathsf{P}(x\,|y) = \frac{\Pi(x)\,\mathsf{P}(y\,|x)}{\sum_z \Pi(z)\,\mathsf{P}(y\,|z)}. \tag{B.2}$$

For continuous valued data the *a priori* distribution Π has the general form of a Gibbs distribution:

$$\Pi(x) = Z^{-1}\exp(-H(x)), Z = \sum_{z \in \mathbf{X}} \exp(-H(z)), \tag{B.3}$$

where H is an energy function. In many instances, the *a posteriori* probability model is another Gibbs distribution, i.e. there is an energy function $H(\cdot\,|y)$ defined in a subspace $\widehat{\mathbf{X}}$ of $\mathbf{X}$ such that

$$\mathsf{P}(x\,|y) = Z^{-1}(y)^{-1}\exp(-H(x\,|y)), x \in \widehat{\mathbf{X}}. \tag{B.4}$$

This energy function is also related to the likelihood distribution. The *a posteriori* energy can be written down as follows:

$$H(x\,|y) = \widetilde{c}(y) - \ln \mathsf{P}(y\,|x) + H(x) \tag{B.5}$$

The mode of the *a posteriori* distribution $\hat{x} = \max_x \{\mathsf{P}(x\,|y)\}$ is the *Maximum a Posteriori* (MAP) estimation of the true image x given the observation y. If the processing is performed independently over all image sites, the mode of the marginal posterior $\widehat{x}_s$ maximizes the posterior marginal distribution $\mathsf{P}(x_s\,|\widehat{y})$.

Maximization of a Gibbs distribution is equivalent to the minimization of its energy function. Therefore, the MAP estimation can be computed without needing to evaluate the partition function Z. Let it be an observation functional dependence on the true data $Y = \varphi(X, \eta)$, where the noise η probabilistic model is denoted Γ. If noise η and the true image X are independent, then the likelihood probabilities are as follows:

$$\mathsf{P}(Y = y\,|X = x) = \Gamma(\varphi(X, \eta) = y). \tag{B.6}$$

Specifically, if noise is Gaussian, the likelihood and *a posteriori* distributions can be written as Gibbs distributions [**?**], [282].

We are dealing with images that can be decomposed into blocks as follows: for each pixel in the image we extract a window of neighboring pixels and we treat it independently from the remaining of the image. We classify it according to a given codebook. The result of this process can be either the central value of the matching codevector or the its class. This approach is termed *VQ Bayesian Filter* (VQBF).

B.2 Block model

Given a collection of image block representatives, a codebook $\Omega^* = \{\omega_i^*; i = 1, .., c\}$ where each representative is an image block of size $d \times d$ pixels, $\omega_i = \left(x_s^P\right)_{s \in S^{d \times d}}$ where the grid $S^{d \times d}$ is defined as a neighborhood:

$$S^{N(d \times d)} = \{s : -d/2 \leq |s| \leq d/2\}. \tag{B.7}$$

Given a sample of image blocks

$$\mathbf{y}^b = \left\{y_i^b; i = 1, .., n; \right\}, \tag{B.8}$$

where

$$y_i^b = \left(y_s^P\right)_{s \in S^{d \times d}}, \tag{B.9}$$

the codebook results from the execution of a codebook design algorithm trying to minimize some objective function related to the data distortion. We do assume that the codebook is designed to minimize the mean square error over the sample::

$$\Omega^* = \min_{\Omega} \left\{ \sum_{i=1}^{n} \left\| y_i^b - \omega_{j(i)} \right\|^2 \right\}, \tag{B.10}$$

where

$$j(i) = y\left(y_i^b\right) = \arg\min \left\{ \left\| y_i^b - \omega_j \right\|^2 ; j = 1, .., c \right\}. \tag{B.11}$$

This minimization corresponds to the maximum likelihood estimation of the parameters of a mixture of Gaussians with identity covariance matrices, which is the model assumed for image block:

$$\mathsf{P}\left(Y^b = y^b\right) = \frac{1}{c} \sum_{j=1}^{c} \frac{1}{(2\pi)^{d/2}} e^{-\frac{1}{2}\left\| y^b - \omega_j \right\|^2} = \sum_{j=1}^{c} \Pi\left(x^b = \omega_j\right) \mathsf{P}\left(Y^b = y^b \left| x^b = \omega_j \right.\right). \tag{B.12}$$

Therefore, the search for the closest codevector

$$j\left(y^b\right) = \arg\min \left\{ \left\| y^b - \omega_j \right\|^2 ; j = 1, .., c \right\}, \tag{B.13}$$

corresponds to the MAP classification of the image block. We assume that the *a priori* probabilities of the image block classes are the same:

$$\Pi\left(\left(x^b = \omega_j^*\right)\right) = \frac{1}{c}, j = 1, .., c. \tag{B.14}$$

The vector quantization given by

$$\widehat{x} = \omega^*_{j(y^b)} \tag{B.15}$$

corresponds to the MAP decision, assuming that the *a posteriori* probabilities of the image blocks x^b are as follows:

$$P\left(x^b = \omega_i^* \left| y^b \right.\right) = \frac{\exp\left(-\frac{1}{2}\left\|y^b - \omega_i^*\right\|^2\right)}{\sum_{j=1}^{c}\exp\left(-\frac{1}{2}\left\|y^b - \omega_j^*\right\|^2\right)}, i = 1, .., c. \tag{B.16}$$

Both the likelihood and the *a posteriori* probabilities of the image blocks may have the shape of Gibbs distributions with energy functions:

$$H\left(Y^b = y^b \left| x^b = \omega_j^* \right.\right) = H\left(x^b = \omega_i^* \left| Y^b = y^b \right.\right) = \frac{1}{2}\left\|y^b - \omega_i^*\right\|^2. \tag{B.17}$$

If the objective function for the vector quantizer design algorihtm is different from the mean square error of the data relative to the representatives, then the probabilistic models above will be completely different.

B.3 VQBF model

In the VQBF the image is not decomposed into image blocks. For each pixel site $s \in S^P$ we select a window around it $S_s^{N(d\times d)} \subset S^P$. Each window $y_s^b = \left(y_{s'}^P\right)_{s' \in S_s^{N(d\times d)}}$ is processed independently. The pixel neighborhood plays the role of contour condition for the pixel process. The pattern matching process between image windows and codevectors is a kind of non-lineal dependence related to the Markovian models of the classical Bayesian image analysis [**?**], [282]. The image is processed in two possible ways, in filtering mode or in classification mode:

1. In filtering mode, the restored pixel value is estimated as the central pixel of the codevector matching the neighboring window:

$$\widehat{x}_s^P = \left(\omega^*_{j(y_s^b)}\right)_{(0,0)}, s \in S^P \tag{B.18}$$

2. In classification mode, the class of the matching codevector becomes the class of the pixel

$$\widehat{x}_s^L = j\left(y_s^b\right), s \in S^P \tag{B.19}$$

Let us consider the filtering mode. Asuming that the pixel neighborhoods are independent, then the posterior probabilities are as follows:

$$\mathsf{P}\left(x \left| y \right.\right) = \prod_{i=1}^{M}\prod_{j=1}^{N} P\left(x^b_{(i,j)} = \omega^*_{j\left(y^b_{(i,j)}\right)} \left| y^b_{(i,j)} \right.\right) \tag{B.20}$$

and the *a posteriori* has the form:

$$H\left(x\,|y\right) = C + \frac{1}{2}\sum_{s}\left\|y_s^b - \omega^*_{j(y_s^b)}\right\|^2. \tag{B.21}$$

The *a priori* distribution corresponds to the joint probability of the pixel windows:

$$\Pi\left(X = x\right) = P\left(x_s^b = \omega_s^*; s \in S^P\right) \tag{B.22}$$

and can not be put into product form. One way to obtain an *a priori* distribution with a Gibbs form is assuming that neighboring pixels have the same value or class if the differences between the windows centered around them are small. This is an *a priori* constraint derived from the codebook. Some expressions for the energy function which follow from this assumption are:

$$H\left(x\right) = \sum_{s,t}|s-t|\left(x_s - x_t\right)^2\left\|x_s^b - x_t^b\right\|^2, \tag{B.23}$$

or

$$H\left(x\right) = \sum_{s}\sum_{t\in N(s)}\left(x_s - x_t\right)^2\left\|x_s^b - x_t^b\right\|^2. \tag{B.24}$$

Taking into account the expression of the *a posteriori* energy function in equation (B.5) we obtain as the logarithm of the likelihood probability [282]:

$$\ln \mathsf{P}\left(y\,|x\right) = C - H\left(\widehat{x}\,|\widehat{y}\right) - H\left(x\right) \tag{B.25}$$

Assuming the energy funciton in equation (B.24) as the *a priori* energy function, we reach the following expression for the log-likelihood

$$\ln \mathsf{P}\left(y\,|x\right) \approx -\sum_{s}\left\|y_s^b - \omega^*_{j(y_s^b)}\right\|^2 - \sum_{s}\sum_{t\in N(s)}\left(x_s - x_t\right)^2\left\|x_s^b - x_t^b\right\|^2 \tag{B.26}$$

If we assume Gaussian additive noise, then the expressions for the potential deformations derived from the likelihood are:

$$Y = \varphi\left(X,\eta\right) = \widehat{\eta} - \left(\sum_{s}\left\|y_s^b - \omega^*_{j(y_s^b)}\right\|^2 - \widehat{\eta}\right) - \sum_{s}\sum_{t\in N(s)}\left(x_s - x_t\right)^2\left\|x_s^b - x_t^b\right\|^2 \tag{B.27}$$

This expression can be interpreted as the definition of the ability of VQBF to correct smooth deformations involving the pixel's neighborhood.

B.4 VQBF edge preservation

The *a posteriori* distribution maximized by VQBF implies that image edges will be preserved by VQBF if the neighborhoods at the two sides of the edge show

a significative variation. This condition is rather general and easy to comply with. It is the justification for the good edge preservations of VQBF. Formally:

$$\widehat{x}_s \neq \widehat{x}_t \text{ si } j\left(y_s^b\right) \neq j\left(y_t^b\right), \tag{B.28}$$

(assuming that different codevectors will have different central pixel values). This expression corresponds to the probability assigned by the likelihood of codevector $\omega^*_{j(y_s^b)}$ to the space outside its decision region $R_{j(y_s^b)}$

$$\begin{aligned} P\left(\widehat{x}_s \neq \widehat{x}_t \,|y\right) &= 1 - P\left(j\left(y_s^b\right) = j\left(y_t^b\right)|y\right) \\ &= 1 - \int_{R_{j(y_s^b)}} \mathsf{P}\left(y_s^b \,|x^b\right) dx^b. \end{aligned} \tag{B.29}$$

Notice that two pixels with identical values can be recovered as different if their neighboring windows change abruptly. However, all these possibilities depend on the codebook and the image statistics.

B.5 Conclusions

This appendix presents a formalization of the VQBF used in other parts of this Thesis, starting from the Bayesian formalization of the image processing. The *maximum a posteriori* (MAP) estimation corresponds to the decision of associating an image block to the pixel under process. Assuming that the *a posteriori* distribution is Gibbs, its logarithm corresponds to the energy function being decomposed into the *a priori* and likelihood energies. From the Bayesian formulation, assuming a conventional *a priori* energy function embodying the smoothness constraints, we can derive the expression of the likelihood distribution. The conventional role of the likelihood distribution is to account for additive noise and assumed image deformations. Following this reasoning, we derive a likelihood energy function that explains the VQBF robustness against some image deformations, and the edge preservation of the VQBF.

Appendix C

Face localization system

The purpose of the system is to provide fast and reliable face localization techniques in real time an in real-life scenes. Person localization is included in this problem. The end application sought is the ability of mobile robots to navigate in human populated environments, and to start visual interaction with them. The system uses motion segmentation, signature analysis and color processing. Signature analysis provides fast hints of the person and face localization. Color processing is used to confirm the face hypothesis. The technique can be implemented in real time and combined with other approaches to enhance the recognition results.

Specifically, the operation of the system presented in [127] is decomposed into two stages. The first tries to localize the head region based on the analysis of the signatures of temporal derivative images. The second provides confirmation of the face hypothesis through the color analysis of the extracted face subimage. The color analysis is a decision performed on the results of color quantization process. The color representatives are computed through an adaptive neural network technique, in fact a supervised version of the well known Kohonen's SOM. In this appendix we describe the first stage of the system in some detail. The second stage has been described in Chapter 13.

C.1 Head localization based on time difference signatures

Aiming to real-time implementations, we have taken a simple and fast computational approach. Segmentation is done on the basis of the analysis of the signatures of the binarized image differences. We compute the pixelwise difference image of two consecutive Gaussian smoothed images. This is the temporal gradient image. When the camera position is fixed, most of the background is removed by such a process. Most of the information that the difference image contains relates to the position of the persons and their faces in the scene. Further, we compute the spatial gradient image of this time difference image. On

this image we compute an optimal thresholding based on the minimum variance clustering to obtain the binary image upon which the signature analysis is performed. Signatures are computed as projections of the image on the image axes. Signature analysis is a classic and useful technique for binary image segmentation [55]. The signature is a very noisy function. Therefore, we have included several heuristics that improve the robustness of the signature analysis, such as the aspect ratio preservation of the localized face window. The goal of this process method is the localization of the rectangle that includes the face and other features that correspond to the head. The signature analysis proceeds in three steps:

1. Individual isolation of each human shape,
2. Rough localization of the head of each individual, and
3. A refinement step for a more precise localization of the head and face, which repeats the previous step restricted to the head subimage.

C.1.1 Individual isolation

For the isolation of human shapes we compute the vertical projections of the binarized difference image. Our hypothesis is that each human shape would correspond to a hill in this signature, so that individuals could be localized looking for local minima between the signature peaks. However, the 1D signature obtained is very noisy. For its heuristic analysis we perform a 1D closing morphological filter. The closing is implemented, as the erosion followed by the dilation of the curve with the same structural element, in our case a flat 1D structural element. In practice this morphological operator is implemented by a 1D min-max filter. The size of the structural element is related to the expected size of the basis of noisy peaks in the function. The morphological closing is expected to remove the noisy peaks leaving the remaining of the function unchanged. Faces and individuals whose projection is smaller than the size of the structural element will disappear. Therefore, the morphological filter bounds the sensitivity of the method to scale variations in faces. On the smoothed signature we search for local minima, that we associate with the boundaries of the human shapes. Peaks corresponding to different individuals must be separated more than the size of the expected local minima basis to be discriminated as separated individuals. We have found experimentally that the approach performs very well for a broad spectrum of scales with a fixed size of the filter. In Figure C.1 we show an instance of a couple images the detection of an individual. This person isolation is limited to the case when individuals are not vertically adjacent.

C.1.2 Localization of the head

Selecting the subimages defined by above procedure, we proceed to the rough localization of the heads. For this purpose we compute the horizontal projections of each binary subimage corresponding to an identified individual. These

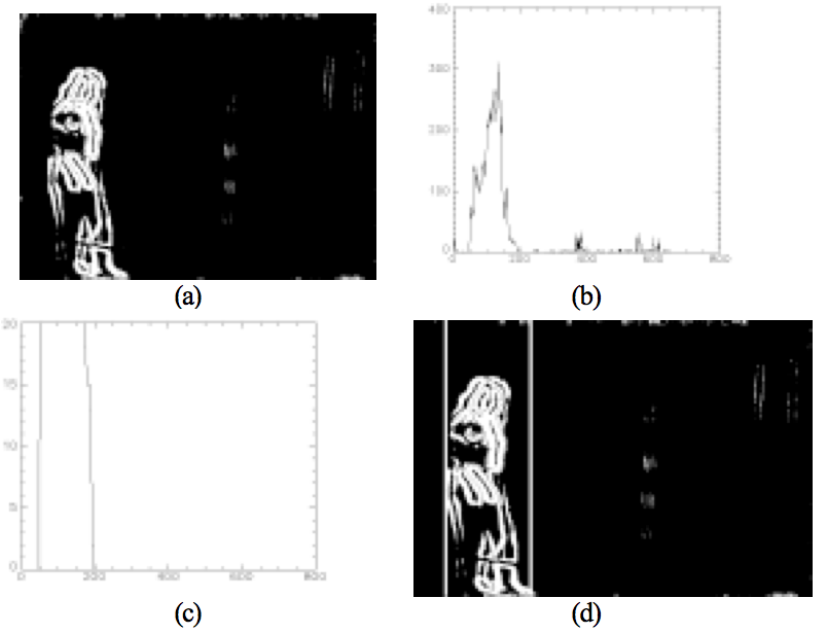
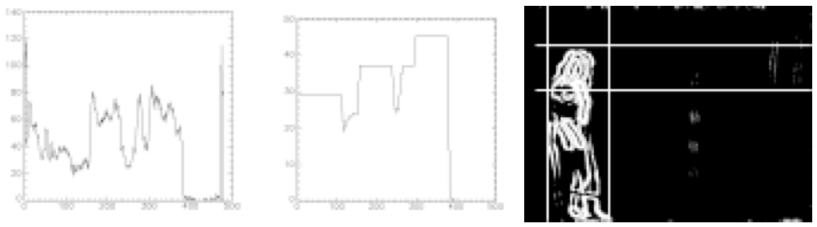


Figure C.1: An instance of the individual isolation (a) motion detected, (b) vertical signature, (c) vertical signature after morphological filtering and (d) individual detected.



(a) (b) (c)

Figure C.2: (a) Horizontal signature of the individual selected region, (b) horizontal signature after morphological filtering, (c) head detected in the binary image.

horizontal signatures are again smoothed with the same morphological closing filter described above. We assume a small threshold so that signature values below it are discarded as noise. To detect the band in which the head is located we look for the first peak that appears examining the smoothed horizontal signature from top to bottom. This peak has the shape of a plateau, and we detect the top and bottom edges of this plateau as the top and bottom limits of the head box. We assume that further peaks correspond to the texture of clothes. In the case of uniform textured clothes the task is simplified, because the only significative peak corresponds straightforwardly to the head. Figure C.2 illustrates the head extraction process.

We select this band of the binary subimage and perform on it a further analysis to detect the left and right sides of the head box. This corresponds to the third stage of the process enumerated above. Again we compute the vertical signature and smooth it with the morphological closing filter. On the smoothed vertical signature, the head appears again as a plateau whose left and right edges correspond to the left and right sides of the head box.

C.1.3 Some experimental results

We have performed experiments on image sequences that show one or two individuals moving before backgrounds of varying complexity. Image sequences were taken using the videocamera of a Silicon O2 workstation. Frame size were 640x480. For the experiments image sequences were captured a different frame rates. Good results were usually obtained at 10 frames per second. The prototype algorithm has been implemented in IDL, and the processing time required for each couple of frames is 5 seconds. Substantial speedup can be expected of more optimized implementations. In Figure C.3 we show some frames of the face localization results obtained over one of these experimental sequences. We have selected this sequence because it shows a very strong change in scale of the face as the subject approaches and goes away from the camera. The size of the face increases steadily from frame #1 up to frame #12, and then decreases

to the end of the sequence. It can be appreciated that the window enclosing the hypothetical face follows the scale increase, although it starts enclosing the whole head and ends, at maximum face size, enclosing the face features (eyes, nose and mouth). Besides this scaling effect the sequence shows also several arbitrary head movements and poses. The face has an almost frontal view in frames #3 to #12. Profile views appear in the succeeding frames. The algorithm performs robustly against these variations in pose quite naturally. We have tested this robustness in several image sequences. The main effect of the changes in illumination and the variation of the amount of motion between frames is the irregular sizing and centering of the box.

C.2 Face validation based on color quantization

There have been some recent works that use color for the task of face localization [110, 52, 255, 262] based on an *a priori* characterization of the skin color in the color space. They assume their color models to be invariant under capture conditions, illumination, camera distortions and so on. On the contrary, our proposition in this paper is to apply an adaptive approach, so that face skin color representatives will be estimated from small samples of images. Besides that, the color process is used to confirm the selection of the hypothetical head described in the previous section.

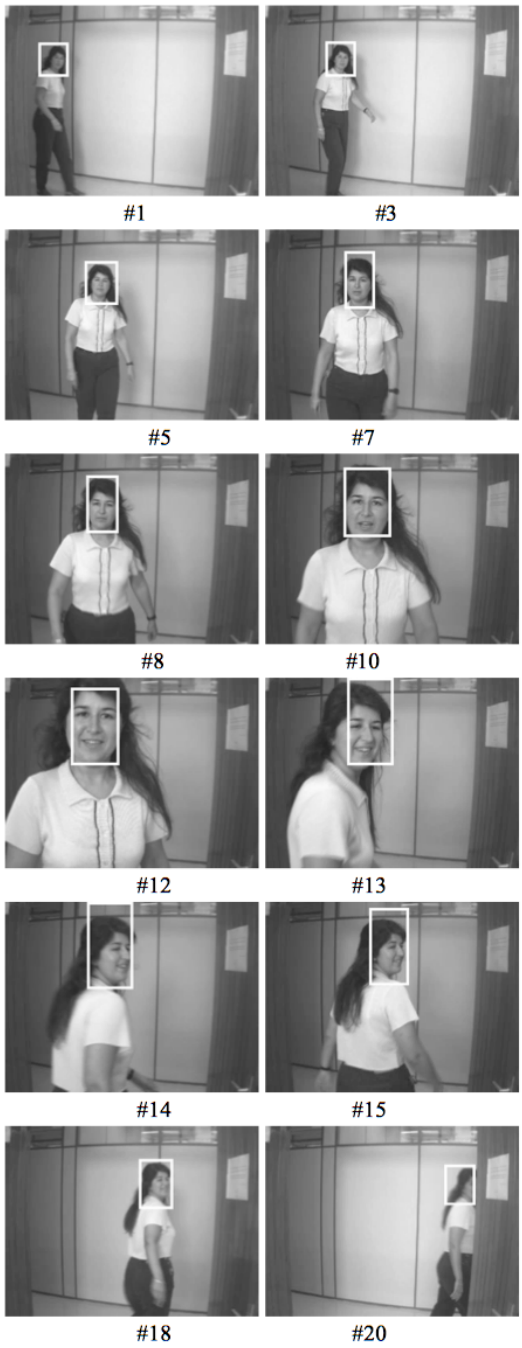


Figure C.3: Localization of the face in several frames extracted from an experimental sequence. The frames show a big change in scale of the face and various poses.

Appendix D

MRI dataset for VQ filtering

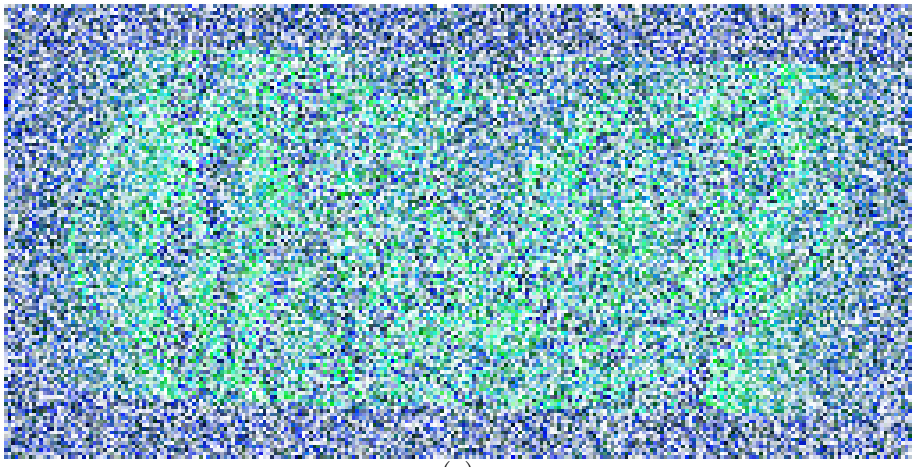
We have used two images provided by the Unit of Magnetic Resonance of the Universidad Complutense. The images have been obtained with an experimental magnet of 4.7 Teslas.

D.1 Human embryo.

It is a 3D image that corresponds to a sequence of 128 cuts of a human embryo. Each image is 128x256 pixels of 32 bits/pixel. A reduction to 8 bits/pixel has been done by ad-hoc manipulations of the intensities based on the statistics over the whole sequence. In Figure (D.1) we show the slice #80 of the sequence as it appears before and after the manipulation. We have worked with this sequence in [115].

D.2 Rat.

A 2D micro-Magnetic Resonance image that corresponds to a model of Miositis produced by Aspergillus in rats. The original image is of 718x717 pixels and is shown in Figure (D.2). We have presented experiment results on this image in [126, 104, 107].



(a)

(b)

Figure D.1: Original frame #80 (a) before and (b) after manual intensity range reduction to 8 bits/pixel

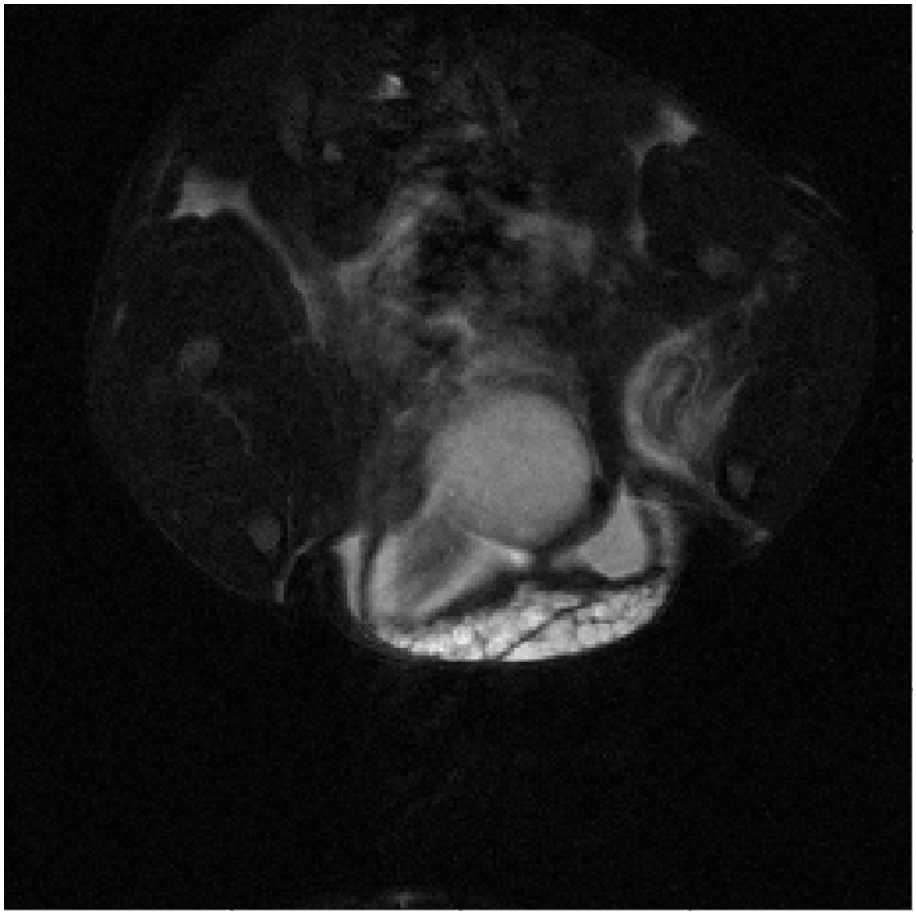


Figure D.2: Original image

Appendix E

Dataset for non-stationary clustering

The sequences of images used for experiments are horizontal rotation pannings of the laboratory taken with an electronic Apple Quicktake camera. Apple CCD colour video- camera designed for video-conferencing

E.1 First Sequence

E.1.1 Original sequence

We have created a sequence of 24 images where each consecutive images overlap 50% of the scene with the next image. Every image has dimension 480x640 pixels. The sequence is shown in Figure (E.1). This sequence has been used in the experiments gathered in our papers [125, 110, 109, 102, 116, 114, 111, 103, 106, 105].

It has also been working on a strip center of size 160x640 pixels extracted from each image of this sequence [112, 113]. We preserve the original horizontal resolution because this is the spatial axis in which camera is moving. This shrunken sequence is shown in figure (E.3).

E.1.2 Distribution in the RGB cube

The main feature of this image sequence is that the distribution of the pixel colors in the RGB space is non-stationary and unpredictable. This is illustrated in Figures (E.4)and E.5, where we present the visualization of the color pixels in the RGB unit cube for the images in the original sequence.

Figure E.1: Original sequence (#1-12)

Figure E.2: Original sequence (#13-24)

Figure E.3: Shrunken sequence

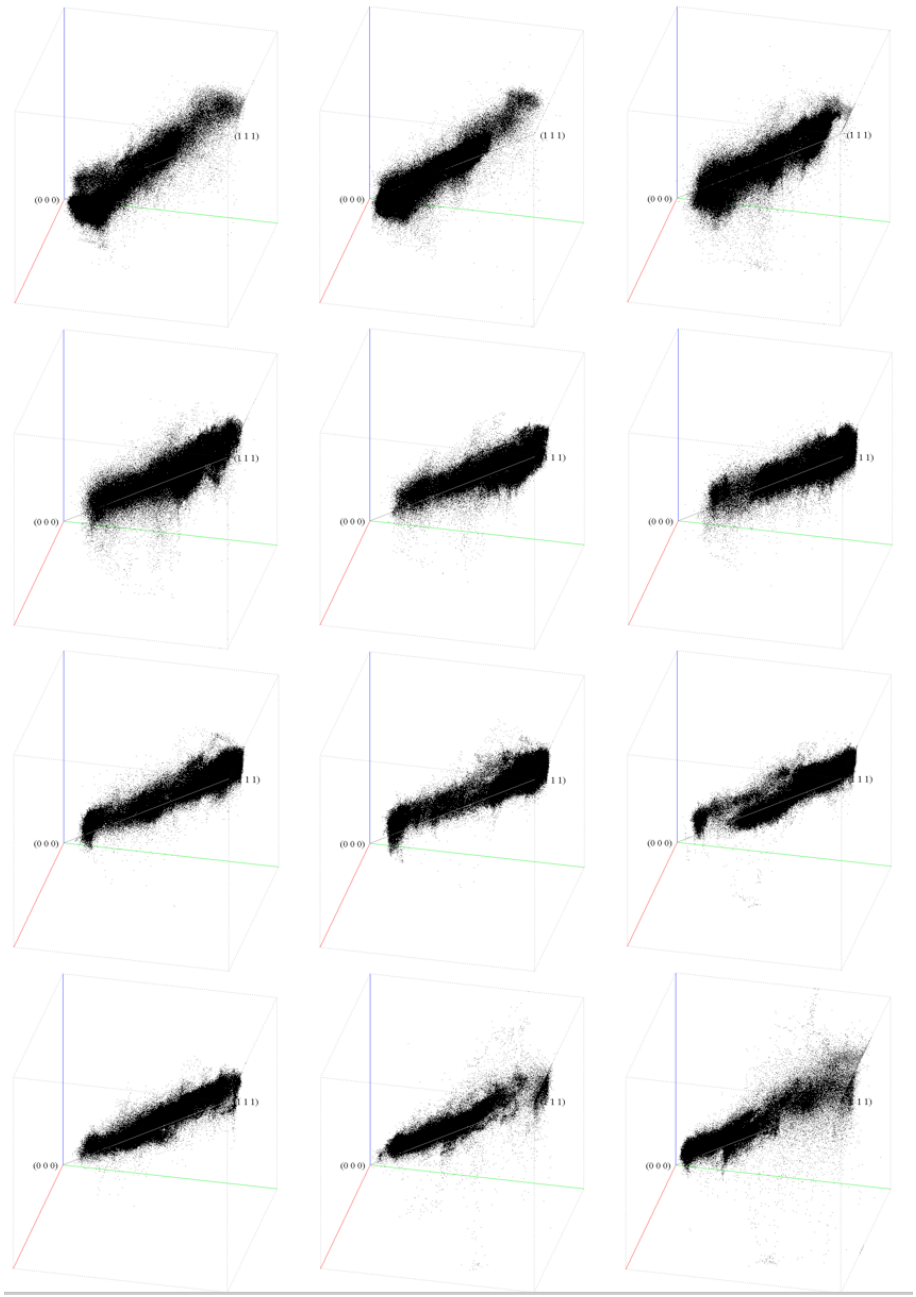


Figure E.4: Distribution of pixels in the RGB cube (#1-12)

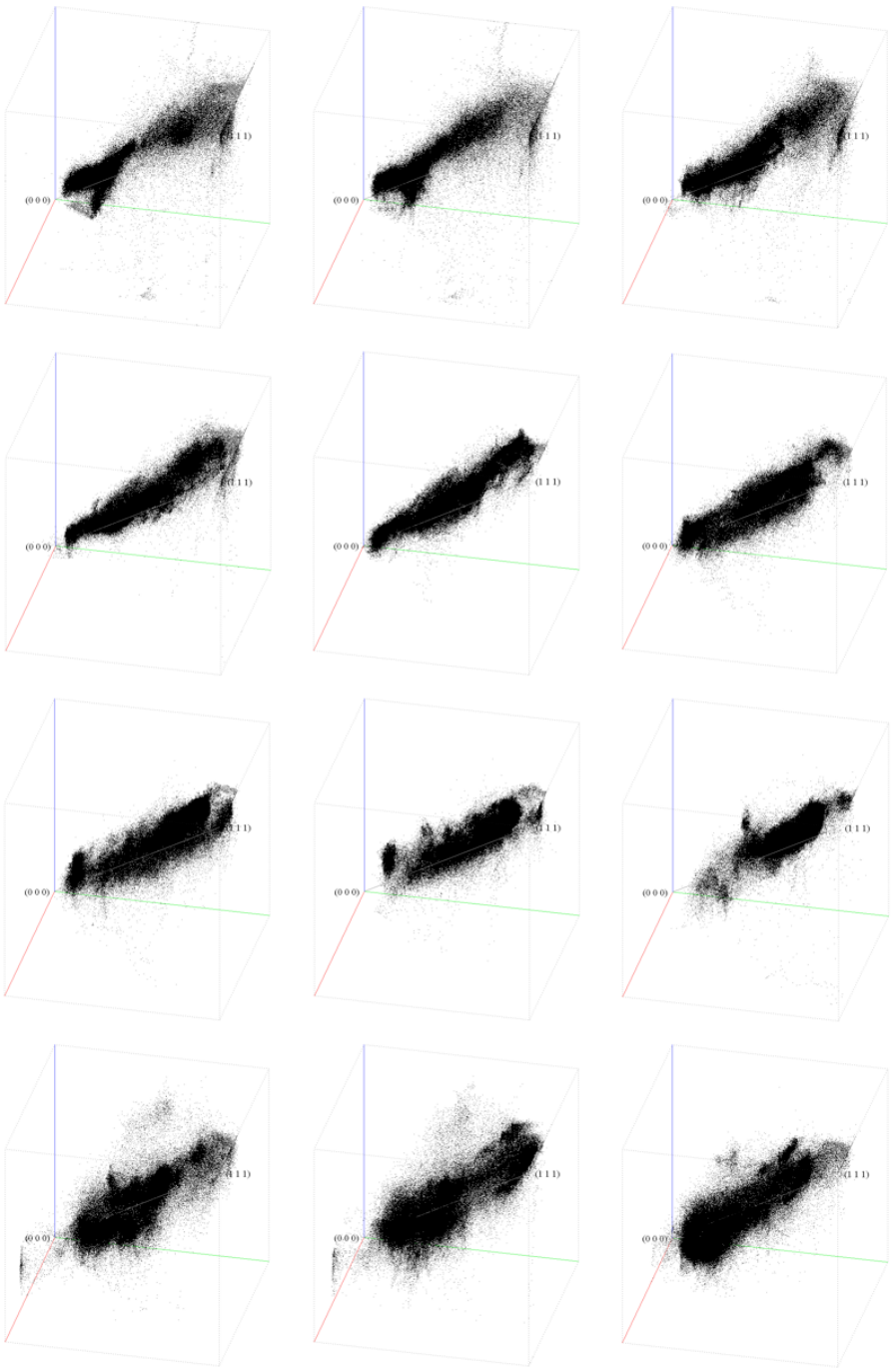


Figure E.5: Distribution of pixels in the RGB cube (#13-24)

E.2 Second Sequences

E.2.1 Original sequences

Original images have an spatial resolution of 240x320 pixels. We have created two sequences. In one of them each of two consecutive images overlap roughly 50% of the scene, a total of 18 images in Figure (E.6). In the other the overlap is of 33%, a total of 27 images in Figures (E.7) y (E.8). This sequences are utilized in [108].

E.2.2 Distributions in the RGB cube

The distributions of pixels in the RGB cube are shown only for the sequence with an overlap of 50% in the following Figures (E.9) and (E.10).

Figure E.6: Sequence with an overlap of 50%.

Figure E.7: Sequence with an overlap of 33% (first part).

Figure E.8: Sequence with an overlap of 33% (last part).

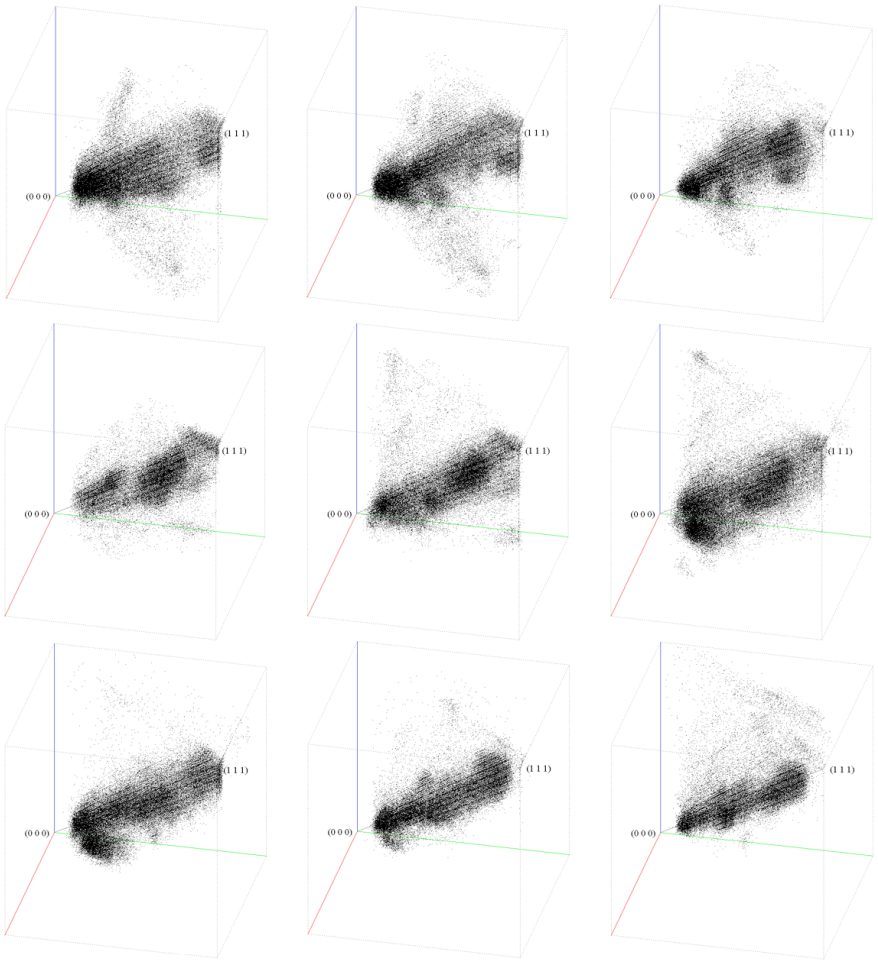


Figure E.9: Distribution of pixels in the RGB cube (#1-9).

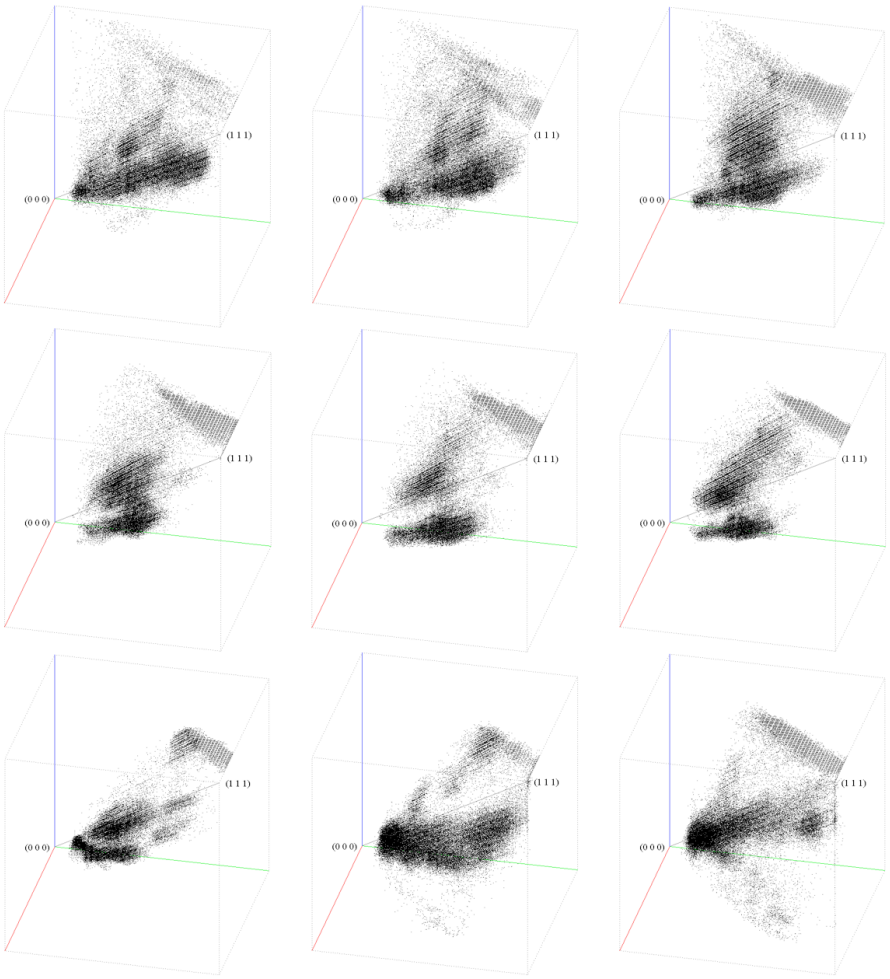


Figure E.10: Distribution of pixels in the RGB cube (#10-18).

Appendix F

The test problems for the HOBM

The learning problems, that we have used in Chapter 15 to test learning power of High Order Boltzmann Machines, have the following common characteristics:

1. The data are in the public domain, and can be accessed by anonymous FTP.

2. Other techniques have been applied to the data, and the results are public. These results play the role of objective references to assert the quality of our own results.

3. The experimental method is clearly defined by the existence of separate train and test data sets.

4. They are classification problems. The output of the network is a binary vector. The class assignment is a vector of zeros with only one 1 component. This characteristic reduces the complexity of the search for the output to a given input.

F.0.3 The Monk's problems

The Monk's problems were defined in [263] over an artificial robot domain, where each robot is described by six discrete variables:

$\mathbf{x}_1$:	head_shape	{round, square, octagon}
$\mathbf{x}_2$:	body_shape	{round, square, octagon}
$\mathbf{x}_3$:	is_smiling	{yes, no}
$\mathbf{x}_4$:	holding	{sword, balloon, flag}
$\mathbf{x}_5$:	jacket_color	{red, yellow, green, blue}
$\mathbf{x}_6$:	has_tie	{yes, no}

The problem statements, previous results and data were taken from "archive.cis.ohio-state.edu", under "pub/neuroprose" via *anonymous ftp* (now avalilable in `http://archive.ics.uci.edu/ml/datasets/MONK%27s+Problems`). Each learning problem is defined by a logical expression involving those variables, that defines the class of robots that must be discovered by the learning algorithms. (Monk's problems are two class problems). Training and test data are produced following the logical definitions. The test data for each problem are the class assignment to the whole space (432 feature vectors). The train data are random subsets of the test data. The methodology used in [263] consists of the elaboration of the model using the train data and testing it against the test data. The result reported for each learning algorithm is the percentage of correct answers to the test set. The logical definition of each problem follows:

$\mathbf{M}_1$ is defined by the relation:

(head_ shape = body_ shape) or (jacket_ color = red)

$\mathbf{M}_2$ is defined by the relation:

Exactly two of the six attributes have their first value

$\mathbf{M}_3$ is defined by the relation:

(jacket_ color is green and holding a sword)
or (jacket_ color is not blue and body_ shape is not octagon).

$\mathbf{M}_1$ is a simple Disjunctive Normal Form expression, and it is supposed to be easily learned by any symbolic algorithm. $\mathbf{M}_2$ is close to a parity problem, difficult to state either as a Disjunctive Normal Form or Conjuctive Normal Form. Finally, the training set for $\mathbf{M}_3$ contains a 5% of erroneous (noisy) patterns, and is intended to evaluate the robustness in the presence of noise.

In [128] we reported some early results on the Monk's problems, still using simulated annealing to estimate the connection statistics. These results vary from the ones reported here due to the absence of the simulated annealing (the results in this paper improve in many cases those in [128]). In the case of the Monk's problems we have used the knowledge of the logical statement of the problems and the logical interpretation of high order connections as generalised AND operators [67, 68] to obtain "a priori" topologies that serve to verify the applicability of Boltzmann Machines to these problems. The "a priori" topologies also show the ideal performance that the learning algorithms may achieve if they are able to discover these "a priori" topologies.

F.0.4 Classification of Sonar Targets

We have used the data used by Gorman and Sejnowski [117] in their study of sonar signal recognition using networks trained with backpropagation. The data has been obtained from the public database at the CMU (node ftp.cs.cmu.edu, directory /afs/cs/project/connect/bench) (now available in `http://archive.ics.uci.edu/ml/datasets/Connectionist+Bench+%28Sonar,+Mines+vs.+Rocks%29`). The goal of the experiment is the discrimination between the sonar signals reflected by rock and metal cylinders. Both the train and test data consist of 104 patterns. The partition between train and test data has been done taking care that the same distribution of incidence angles appears in both sets.

Each pattern has 60 input features and a binary output. Input features are real numbers falling in the $[0, 1]$ interval.

In [117] a set of experiments was performed with a varying number of hidden units, to explore the power of the learning algorithm depending on the topology. Results were averaged over 10 replications of the learning process with varying random initial weights. The best result reported was an average 90.4 per cent of success on the test data, with a standard deviation of 1.8, for a topology with 12 hidden units. Besides the topological exploration, in this paper the problem serves to introduce continuous units with $[0, 1]$ state space. Obviously, there is no known "a priori" topology for this problem.

F.0.5 Vowel recognition

The data for this problem has also been obtained from the CMU public database (now available in `http://archive.ics.uci.edu/ml/datasets/Connectionist+Bench+%28Vowel+Recognition+-+Deterding+Data%29`. They have been used, at least, to realise two PhD Thesis [235, 72, 234]. The Thesis of Robinson included the application of several neural architectures to the problem. The best results reported by Robinson were obtained with a Euclidean nearest neighbour classifier. It attains a 56% success on the test data. Other source of results for this database is [61] where a best result of 58% success on the test was reported. Each pattern is composed of 10 input features. Input features are real numbers. The class (vowel) assignment is given by a discrete variable.

The details of the data gathering and pre-processing can be found in [235, 234]. The training data contains 528 patterns, and the test data contains 462 patterns. From our point of view, there are three specific characteristics that make this problem worthy of study. First, it is a multicategorical classification problem, whereas the Monk's and Sonar problems involve only two categories. Second, the input features are not normalised in the $[0, 1]$ interval. Roughly, they take values in the $[-5, 5]$ interval. Finally, the convexity of the Kullback-Leibler distance in this case is doubtful. We wish to test the robustness of the approach taken (especially the initialisation of the weights to zero) in this clearly unfavourable case.

Bibliography

[1] E.H.L. Aarts and J.H.M. Korst. *Simulated Annealing and Boltzmann Machines: a stochastic approach to combinatorial optimization and neural computing*. John Wiley & Sons, 1989.

[2] D.H. Ackley, G.E. Hinton, and T.J. Sejnowski. A learning algorithm for boltzmann machines. *Cogn. Sci.*, 9:147–169, 1985.

[3] S.C. Ahalt, A.K. Krishnamurthy, P. Chen, and D.E. Melton. Competitive learning algorithms for vector quantization. *Neural Networks*, 3:277–290, 1990.

[4] A.N. Aizawa and B.W. Wah. Scheduling of genetic algorithms in a noisy environment. *Evolutionary Computation*, 2(2):97–122, 1994.

[5] F. X. Albizuri, A. D'Anjou, M. Graña, and J. A. Lozano. Convergence properties of high-order boltzmann machines. *Neural Netw.*, 9:1561–1567, December 1996.

[6] F.X. Albizuri. *Maquina de Boltzmann de alto orden: una red neuronal con tecnicas de Monte Carlo para modelado de distribuciones de probabilidad. Caracterizacion y estructura*. PhD thesis, Dept. CCIA Univ. Pais Vasco, 1995.

[7] F.X. Albizuri, A. D'Anjou, M. Graña, F.J. Torrealdea, and M.C. Hernandez. The high order boltzmann machine: learned distribution and topology. *IEEE Trans. Neural Networks*, 6(3):767 – 770, 1995.

[8] C. Alippi and R. Cucchiara. Cluster partitioning in image analysis classification: a genetic algorithm approach. In *Proc IEEE COMPEURO, Computer Systems and Software Engineering*, pages 139–144, 1992.

[9] E. L. Allgower and K. Georg. *Numerical Continuation Methods. An Introduction*, volume 13 of *Computational Mathematics*. Springer-Verlag, Berlin/Heidelberg, Germany, 1990. (2ed 2003).

[10] E.L. Allgower and K. Georg. *Introduction to Numerical Continuation Methods*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2003.

[11] J. Alspector, T. Zeppenfeld, and S. Luna. A volatility measure for annealing in feedback neural networks. *Neural Computation*, 4:191–195, 1992.

[12] S. Amari. Dualistic geometry of the manifolds of high-order neurons. *Neural Networks*, 4:443–451, 1991.

[13] S. Amari, K. Kurata, and H. Nagaoka. Information geometry of boltzmann machines. *IEEE trans. Neural Networks*, 3(2):260–271, 1992.

[14] E.B. Andersen. *The statistical analysis of categorical data 1991.* Springer Verlag, 1991.

[15] P. Andrey and P. Tarroux. Unsupervised image segmentation using a distributed genetic algorithm. *Pattern Recognition*, 27(5):659–673, 1994.

[16] D. Androutsos, K.N. Plataionitisand, and A.N. Venetsanopoulos. A perceptually motivated technique to query by example using color cardinality. In *Proc. Multimedia Storage and Archiving Systems IV, Proc. SPIE*, volume 3846, pages 137–145, 1999.

[17] M. Atkins and B. Mackicwich. Fully automated segmentation of the brain in magnetic resonance imaging. *IEEE Trans. Med. Imaging*, 17:98–107, 1998.

[18] G. P. Babu and N.M. Murty. Clustering with evolution strategies. *Pattern Recognition*, 27(2):321–329, 1994.

[19] T. Back and H.P. Schwefel. An overview of evolution algorithms for parameter optimization. *Evolutionary Computation*, 1:1–24, 1993.

[20] T. Back and H.P. Schwefel. Evolutionary computation: an overview. In *IEEE ICEC'96*, pages 20–29, 1996.

[21] A. Barmpoutis and G.X. Ritter. Orthonormal basis lattice neural networks. In *Fuzzy Systems, 2006 IEEE International Conference on*, pages 331 –336, 2006.

[22] R. Baumgartner, L. Ryner, W. Richter, R. Summers, M. Jarmasz, and R. Somorjai. Comparison of two exploratory data analysis methods for fmri: fuzzy clustering vs. principal component analysis. *Magnetic Resonance Imaging*, 18:89–94, 2002.

[23] L. Bedini, I. Gerace, and A. Tonazzini. A gnc algorithm for constrained image reconstruction with continuous-valued line processes. *Pattern Recogn. Lett.*, 15(9):907–918, 1994.

[24] C. Berge. Optimisation and hypergraph theory. *European Journal of Operational Research*, 46:297–303, 1990.

[25] T. Berger. *Rate Distortion Theory.* Prentice Hall, Englewood Cliffs, NJ, 1971.

[26] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 1981.

[27] J.C. Bezdek, S. Boggavaparu, L.O. Hall, and A. Bensaid. Genetic algorithm guided clustering. In *Proc 1st IEEE Conf. Evol. Comp*, pages 34–39. IEEE press, 1994.

[28] J.C. Bezdek, L.O. Hall, and L.P. Clarke. Review of mr image segmentation using statistical pattern recognition. *Med. Phys.*, 20:1033–48, 1993.

[29] J.C. Bezdek and R.J. Hathaway. Optimization of fuzzy clustering criteria using genetic algorithms. In *Proc 1st IEEE Conf. Evol. Comp.*, pages 589–594, 1994.

[30] J.C. Bezdek and N.R. Pal. Two soft relatives of learning vector quantization. *Neural Networks*, 8:729–743, 1995.

[31] S. M. Bhandarkar and H. Zhang. Image segmentation using evolutionary computation. *IEEE Transactions on Evolutionary Computation*, 3(1):1–21, 1999.

[32] J. N. Bhuyan, V.V. Raghavan, and V.K. Elayavalli. Genetic algorithm for clustering with an ordered representation. In *Proc. 4th Int. Cong. Gen. Alg.*, pages 408–415, 1991.

[33] G.L. Bilbro, R. Mann, T.K. Miller, W.E. Snyder, D.E. van den Bout, and M. White. Optimization by mean field annealing. In Touretzky, editor, *Advances in Neural Information Processing Systems I*, pages 91–98. Morgan Kauffamn, 1989.

[34] G.L. Bilbro, W.E. Snyder, S.J. Garnier, and J.W. Gault. Mean field annealing: A formalism for constructing gnc-like algorithms. *IEEE trans. Neural Networks*, 3(1):131–138, 1992.

[35] C. Bishop. Improving the generalization properties of radial basis function neural networks. *Neural Computation*, 3:579–588, 1991.

[36] Y.M. Bishop, S.E. Fienberg, and P.W. Holland. *Discrete Multivariate Analysis. Theory and Practice*. MIT Press, 1975. (10th edition 1989).

[37] M.J. Black and A. Rangarajan. On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *Int. J. Comput. Vision*, 19(1):57–91, 1996.

[38] A. Blake. The least disturbance principle and weak constraints. *Pattern Recognition Letters*, pages 393–399, 1983.

[39] A. Blake. Comparison of the efficiency of deterministic and stochastic algorithms for visual reconstruction. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(1):2–12, 1989.

[40] A. Blake and A. Zisserman. *Visual Reconstruction*. The MIT Press, Cambridge, Massachusetts London, England, 1987.

[41] K. Blekas and A. Stafylopatis. Real coded genetic optimization of fuzzy clustering. In *Proc. EUFIT'96*, pages 461–465, 1996.

[42] E. Bodt, M. Cottrell, P. Letremy, and M. Verleysen. On the use of self-organizing maps to accelerate vector quantization. *Neurocomputing*, 56:187–203, 2004.

[43] E. Bodt, M. Verleysen, and M. Cottrell. Kohonen maps versus vector quan- tization for data analysis. In M. Verleysen, editor, *Proc. ESANN'97*, pages 211–218. dFacto press, Brussels, 1997.

[44] J.M. Boone. Neural networks at the crossroads. *Radiology*, 189:357–359, 1993.

[45] Y. Boykov, O. Veksler, and R. Zabih. A new algorithm for energy minimization with discontinuities. In *EMMCVPR '99: Proceedings of the Second International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 205–220, London, UK, 1999. Springer-Verlag.

[46] T. Brox, M. Rousson, R. Deriche, and J. Weickert. Colour, texture, and motion in level set based segmentation and tracking. Technical report 147, Dept. of Mathematics, Saarland University, Saarbrücken, Germany, August 2005.

[47] B.P. Buckles, F.E. Petry, D. Prabhu, R. George, and R. Srikanth. Fuzzy clustering with genetic search. In *Proc. 1st IEEE Conf. Evol. Comp.*, pages 46–50, 1994.

[48] G. Bueno, O. Musse, F. Heitz, and J.P. Armspach. Three-dimensional segmentation of anatomical structures in mr images on large data base. *Magnetic Resonance Imaging*, 19:73–88, 2001.

[49] J. Buhmann and H. Kuhnel. Vector quantization with complexity costs. *Information Theory, IEEE Transactions on*, 39(4):1133–1145, jul 1993.

[50] J.M. Buhmann and J. Puzicha. Basic principles of annealing for large scale non-linear optimization. In Springer-Verlag Berlín, editor, *Online Optimization of Large Scale Systems*, pages 749–777. Martin Grötschel,Sven O. Krumke,Jörg Rambau, 2001.

[51] C.S. Calude and J.P. Lewis. Is there a universal image generator? Reseach Report Series CDMTCS-344, Centre for Discrete Mathematics and Theoretical Computer Science, January 2009.

[52] D. Chai and K.N. Ngan. Locating facial region of a head-and-shoulders color image. In *Proc Int. Conf on Autom. Face and Gesture recognition*, pages 124–129. IEEE press, 1998.

[53] C. Chan and M. Vetterli. Lossy compression of individual signals based on string matching and one pass codebook design. In *Proc. ICASSP'95*, pages 2491–2494, 1995.

[54] C. Chan and M. Vetterli. Lossy compression of individual signals based on string matching and one pass codebook design. In *IEEE Conference on Acoustics, Speech and Signal Processing*, volume 4, pages 2491–2494, 1995.

[55] R. Chelappa, C.L. Wilson, and S. Sirohey. Human and machine recognition of faces: A survey. *Pro*, 83(5):705–740, 1995.

[56] H.H. Chen, Y.S. Chen, and W.H. Hsu. Low rate sequence image coding via vector quantization. *Signal Processing*, 26:265–283, 1995.

[57] J.Y. Chen, C. Taskiran, A. Albiol, C.A. Bouman, and E.J. Delp. Vibe: A video indexing and browsing environment. In *Proc Multimedia Storage and Archiving Systems IV, Proc SPIE*, volume 3846, pages 148–164, 1999.

[58] O.T. Chen, B.J. Chen, and Z. Zhang. An adaptive vector quantization based on the gold-washing method for image compression. *IEEE Trans circuits & systems for video techn.*, 4(2):143–156, 1994.

[59] R.J. Chen and B.C. Chien. Three dimensional morphological pyramid and its application to color image sequence coding. *Signal Processing*, 44:163–180, 1995.

[60] C. Chinrungrueng and C. Sequin. Optimal adaptive k-means algorithm with dynamic adjustment of learning rate. *IEEE Trans. on Neural Networks*, 6(1):157–169, 1995.

[61] F.L. Chung and T.Lee. Fuzzy competitive learning. *Neural Networks*, 7(3):539–551, 1994.

[62] D. Chyzhyk, M. Graña, A. Savio, and J. Maiora. Hybrid dendritic computing with Kernel-LICA applied to Alzheimer's Disease detection in MRI. *Neurocomputing*, in press, 2011.

[63] L.P. Clarke, R.P. Velthuizen, M.A. Camacho, J.J. Heine, M. Vaidyanathan, L.O. Hall, R.W. Thatcher, and M.L. Silbiger. Mri segmentation: methods and applications. *Magnetic Resonance Imaging*, 13(3):343–68, 1995.

[64] J.H. Conway and N.J.A. Sloane. Fast quantizing and decoding algorithms for lattice quantizers and codes. *IEEE trans. Inf. Theory*, 28:227–232, 1982.

[65] P.C Cosman, K.L. Oehler, E.A. Riskin, and R.M. Gray. Using vector quantization for image processing. *IEEE Proceedings*, 81(4):1326–1341, 1993.

[66] M. Cotrell, J.C. Fort, and G. Pages. Two or three things that we know about the kohonen algorithm. In M. Verleysen, editor, *ESANN'94*, pages 235–244. Brussels: dFacto press, 1994.

[67] A. D'Anjou, M. Graña, F.J. Torrealdea, and M.C. Hernandez. Maquinas de boltzmann para la resolucion del problema de la satisfiabilidad en el calculo proposicional. *Revista Española de Informatica y Automatica*, 24:40–49, 1992.

[68] A. D'Anjou, M. Graña, F.J. Torrealdea, and M.C. Hernandez. Solving satisfiability via boltzmann machines. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(5):514–521, 1993.

[69] B.M. Dawant, S.L. Hanmann, J.P. Thirion, F. Maes, D. Vandermeulen, and P. Demaerel. Automatic 3d segmentation of internal structures of the head in mr images using a combination of similarity and free-form transformations: Part 1. methodology and validation on normal subjects. *IEEE Trans. Med. Imaging*, 18:909–26, 1999.

[70] R. Perez de Alejo, J. Ruiz-Cabello, I. Echave M. Cortijo, J. Regadera, J. Arrazola, P. Avilés, P. Barreiro, D. Gargallo, and M. Graña. Computer-assisted enhanced volumetric segmentation magnetic resonance imaging data using a mixture of artificial neural networks. *Magnetic Resonance Imaging*, 21(8):901–912, 2003.

[71] A.H. Dekker. Kohonen neural networks for optimal colour quantization. *Network: Comp. Neural Sys.*, 5:351–367, 1994.

[72] D. H. Deterding. *Speaker Normalisation for Automatic Speech Recognition*. PhD thesis, University of Cambridge,, 1989.

[73] E. Diday and J.C. Simon. Clustering analysis. In K.S. Fu, editor, *Digital Pattern Recognition*, pages 47–94. Springer Verlag, 1980.

[74] G. Dong and K. Palaniappan. A robust method for edge-preserving image smoothing. *Lecture Notes Computer Science*, 5259:390–399, January 2008.

[75] N. Duda and M. Sonka. Segmentation and interpretation of mr brain images: an improved active shape model. *IEEE Trans. Med. Imaging*, 17:1049–62, 1998.

[76] R.D. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.

[77] R. Durbin and D.E. Rumelhart. Product units: A computationally powerful and biologically plausible extension to backpropagation networks. *Neural Computation*, 1(1):133–142, 1989.

[78] Mehdi Fallahnezhad, Mohammad Hassan Moradi, and Salman Zaferanlouei. A hybrid higher order neural classifier for handling classification problems. *Expert Syst. Appl.*, 38:386–393, January 2011.

[79] E. Farguell Matesanz. *A new approach to Decimation in High Order Boltzmann Machines*. PhD thesis, Universitat Ramon Llull. EALS - Electronica, 2011.

[80] M.A.T. Figueiredo and J.M.N. Leitao. Simulated tearing: an algorithm for discontinuity-preserving visual surface reconstruction. *Computer Vision and Pattern Recognition, 1993. Proceedings CVPR '93., 1993 IEEE Computer Society Conference on*, -:28–33, 1993.

[81] J.M. Fitzpatrick and J.J. Grefenstette. Genetic algorithms in noisy environments. *Machine Learning*, 3:101–120, 1988.

[82] J.C. Fort, P. Letremy, P.my, and M. Cottrell. Advantages and drawbacks of the batch kohonen algorithm. In M. Verleysen, editor, *Proc. of ESANN'2002*, pages 223–230. D Facto, Bruxelles, 2002.

[83] J.C. Fort and G. Pages. On the a.s. convergence of the kohonen algorithm with a general neighborhood function. *The Annals of Applied Probability*, 5(4):1177–1216, 1995.

[84] J.C. Fort and G. Pages. About the kohonen algorithm: strong or weak self-organization? *Neural Networks*, 9(5):773–785, 1996.

[85] J.C. Fort and G. Pages. Convergence of stochastic algorithms: from the kushner-clark theorem to the lyapunov functional method. *Adv. Appl. Prob.*, 28:1072–1094, 1996.

[86] J.E. Fowler. *Adaptive Vector Quantisation for the coding of nonstationary sources*. PhD thesis, Ohio University, 1996.

[87] B. Fritzke. Growing cell structures: a self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7:1441–1460, 1994.

[88] B. Fritzke. The lbg-u method for vector quantization - an improvement over lbg inspired from neural networks. *Neural Processing Letters*, 5(1):35–45, 1997.

[89] K. Fukunaga. *Statistical Pattern Recognition*. Academic Press, 1990.

[90] J.L. Furlani, L. McMillan, and L. Westover. Adaptive colormap selection algorithm for motion sequences. In *Proc. Multimedia'94,*, pages 341–347, San Francisco, CA, 1994.

[91] M. García-Sebastián, A. Savio, M. Graña, and J. Villanúa. On the use of morphometry based features for alzheimer's disease detection on mri. *Bio-Inspired Systems: Computational and Ambient Intelligence. / IWANN 2009 (Part I) Joan Cabestany, Francisco Sandoval, Alberto Prieto, Juan M. Corchado (Editors), LNCS 5517*, pages 957–964, 2009.

[92] H. Garudadri, P. Labute, G. Boulianne, and P. Kenny. Fast match acoustic models in large vocabulary continuous speech recognition. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on, ICASSP-94*, volume 3, 1994.

[93] H. Gefner and J. Pearl. On the probabilistic semantics of connectionist networks. In *Proc. 1st IEEE Int. Conf. Neural Networks*, pages 187–195, 1987.

[94] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions and the bayesian restoration of imagesieee trans. pami. *IEEE trans. PAMI*, 6:721–741, 1984.

[95] G. Gerig, O. Kübler, R. Rikinis, and F.A. Jolesz. Nonlinear anisotropic filtering of mri data. *IEEE Trans. Med. Imaging*, 11:221–32, 1992.

[96] A. Gersho. On the structure of vector quantizers. *IEEE Trans. Inf. Th.*, 28(2):157–166, 1982.

[97] A. Gersho and R.M. Gray. *Vector Quantization and signal compression*. Kluwer, 1992.

[98] J.O. Glass, W.E. Reddick, O. Goloubeva, V. Yo, and R.G. Steen. Hybrid artificial neural network segmentation of precise and accurate inversion recovery (pair) images from normal human brain. *Magnetic Resonance Imaging*, 18:1245–53, 2000.

[99] M. Goldberg and H. Sun. Image sequence coding using vector quantization. *IEEE trans. Communications*, 34:703–710, 1986.

[100] E. Goles and S. Martinez. *Neural and Automata Networks: Dynamical Behavior and Applications*. Kluwer Acad. Pub., 1991.

[101] Y. Gong, H. Zen, Y. Ohsawa, and M. Sakauchi. A color video image quantization method with stable and efficient color selection capability. In *Int. Conf. Pattern Recognition*, volume 3, pages 33–36, 1992.

[102] A. I. González and M. Graña. Competitive neural networks as adaptive algorithms for non-stationary clustering: experimental results on the color quantization of image sequences. In *ICNN97*, volume 3, pages 1602–1607. IEEE press, 1997. ISBN: 0-7803-4122-8.

[103] A. I. González and M. Graña. Una estrategia evolutiva para clustering no estacionario. *Inteligencia Artificial*, 98(5):68–73, 1998.

[104] A. I. González and M. Graña. Diseño mediante una estrategia evolutiva de filtros para imágenes basados en agrupamiento. In J. Mira and J. V. Sánchez-Andrés, editors, *CAEPIA99*, pages 106–113. Comité Organizador CAEPIA-TTIA'99, 1999. ISBN: 931170-2-1.

[105] A. I. González and M. Graña. Colour quantization of image sequences using competitive neural networks. In M.J. Turner J.M. Blackledge, editor, *Image Processing II: Mathematical Methods, Algorithms & Applications*, pages 313–338. Horwood publishing, 2000. ISBN: 1-898563-61-6.

[106] A. I. González, M. Graña, F. Albizuri, A. D'Anjou, and F. J. Torrealdea. A near real-time evolution-based adaptation strategy for dynamic color quantization of image sequences. *Information Sciences*, 122:161–183, February 2000.

[107] A. I. González, M. Graña, J. Ruiz Cabello, A. D'Anjou, and F. X. Albizuri. Experimental resuls of an evolution-based adaptation strategy for vq image filtering. *Inf. Sci. Inf. Comput. Sci.*, 133:249–266, April 2001.

[108] A. I. González, M. Graña, and M. Cottrell. Basic competitive neural networks as adaptive mechanisms for nonstationary color quantization. *Neural Computing and Applications*, 8:347–367, 1999.

[109] A. I. González, M. Graña, A. D'Anjou, F. Albizuri, and M. Cottrell. Self organizing map for adaptive non-stationary clustering: some experimental results on color quantization of image sequences. In M. Verleysen, editor, *ESANN97*, pages 199–204. dFacto press, Bruselas, 1997. ISBN: 2-9600049-7-3.

[110] A. I. González, M. Graña, A. D'Anjou, F. Albizuri, and M. Cottrell. A sensitivity analysis of the self organizing map as an adaptive onepass non-stationary clustering algorithm: The case of color quantization of image sequences. *Neural Processing Letters*, 6:77–89, December 1997.

[111] A. I. González, M. Graña, A. D'Anjou, F.X. Albizuri, and F. J. Torrealdea. A comparison of experimental results with a evolution strategy and competitive neural networks for near real-time color quantization of image sequences. *Applied Intelligence special issue Evolutionary Learning*, 8(1):43–51, January 1998.

[112] A. I. González, M. Graña, A. D'Anjou, and M. Cottrell. On the application of competitive neural networks to time-varying clustering problems. In L.B. Almeida F.L. Silva, J.C. Principe, editor, *Spatiotemporal models in biological and artificial systems*, pages 49–55. IOS Press, 1996. ISBN: 90-5199-304-8.

[113] A. I. González, M. Graña, A. D'Anjou, and M. Cottrell. On the application of competitive neural networks to time-varying clustering problems. In *WSC'96 –on line–*, 1996.

[114] A. I. González, M. Graña, A. D'Anjou, and F. J. Torrealdea. Anticipation versus adaptation in evolutionary algorithms: The case of non-stationary clustering. In D.M. Dubois, editor, *Computing Anticipatory Systems: CASYS - First International Conference*, volume 437, pages

517–527. Springer Verlag, AIP Conference Proceedings., 1997. ISBN: 1-56396-827-4.

[115] A. I. González, M. Graña, I. Echave, and J. Ruiz-Cabello. Bayesian vq image filtering design with fast adaptation competitive neural networks. In J. V. Sanchez-Andres J. Mira, editor, *Engineering Applications of Bio-inspired Artificial Neural Networks*, volume 1607, pages 341–350. Springer-Verlag, LNCS, 1999. ISBN: 3-540-66068-2.

[116] A. I. González, M. Graña, J. Lozano, and P. Larrañaga. Experimental results of a michigan-like evolutionary strategy for nonstationary clustering. In N C Steele G D Smith and R F Albrecht, editors, *Artificial Neural Nets and Genetic Algorithms*, pages 555–559. Springer Verlag, 1997. ISBN: 3-211-83087-1.

[117] R.P. Gorman, , and T.J. Sejnowski. Analysis of hidden units in a layered network trained to classify sonar targets. *Neural Networks*, 1:75–89, 1988.

[118] M. Graña. A brief review of lattice computing. In *Proc. WCCI 2008*, pages 1777–1781, 2008.

[119] M. Graña, D. Chyzhyk, M. García-Sebastián, and C. Hernández. Lattice independent component analysis for functional magnetic resonance imaging. *Information Sciences*, 181:1910–1928, 2011.

[120] M. Graña, A. D'Anjou, F. Albizuri, J. Lozano, P. Larrañaga, Y. Yurramendi, M. Hernandez, J.L. Jimenez, F. J. Torrealdea, M. Poza, and A. I. González. Experimentos de aprendizaje con maquinas de boltzmann de alto orden. *Informatica y Automatica*, 29(4):42–57, 1996.

[121] M. Graña, A. D'Anjou, A. I. González, F.X. Albizuri, and M. Cottrell. Local stochastic learning rule for image vector quantization. In A.R. Figueiras D. Docampo, editor, *Adaptive Systems, Intelligent approaches, massively parallel computing and emergent techniques in signal processing and communications*, pages 219–222. Universidad de Vigo, 1994. ISBN: 84-605-1547-8.

[122] M. Graña, A. D'Anjou, A.I. González, F.X. Albizuri, and M. Cottrell. Local stochastic competition for vector quantization of images. In B. Widrow P. Werbos, H. Szu, editor, *Proc. of INNS WCNN94*, volume 4, pages 205–210, San Diego, June 1994. Lawrence Elbaum. ISBN: 0-8058-1745-X.

[123] M. Graña and I. Echave. Real-time optical flow computation based on adaptive color quantization by competitive neural networks. In *Intelligent Robots and Computer Vision XVIII, Proc SPIE*, volume 3837, pages 165–174, 1999.

[124] M. Graña, J. Echave, I. Ruiz-Cabello, and M. Cortijo. Segmentation of infected tissues in iri using vq-bf filtering. In *Proc. ICSP 2002*, Beijing, Chine, 2002. IEEE Press.

[125] M. Graña, A. I. González, A. D'Anjou, J.A. Lozano, P. Larrañaga, and F. Albizuri. Evolutive strategies for adaptive color quantization. In F. Perez D. Docampo, A.R. Figueiras, editor, *Intelligent Methods for Signal Processing and Communications*, pages 174–178. Universidad de Vigo, 1996. ISBN: 84-8158-043-0.

[126] M. Graña, A. I. González, I. Echave, and J. Ruiz-Cabello. Vq based image filtering. In A. Sanfeliu M.I. Torres, editor, *Pattern Recognition and Image Analysis*, pages 471–478. Ediciones GNB., 1999. ISBN: 84-95120-80-1.

[127] M. Graña, A. I. González, B. Raducanu, and I. Echave. Fast face localization for mobile robots: signature analysis and color processing. In D. P. Casasent, editor, *Intelligent robots and computer vision XVII: Algorithms, Techniques and Active Vision*, pages 387–398. SPIE Conference Proceedings., 1998. ISBN: 0-8194-2983-X.

[128] M. Graña, V. Lavin, A. D'Anjou, F.X. Albizuri, and J.A. Lozano. High-order boltzmann machines applied to the monk's problems. In *Proc. ESSAN'94*, pages 117–122, 1994.

[129] M. Graña, M. Termenon, A. Savio, A. Gonzalez-Pinto, J. Echeveste, J.M. Pérez, and A. Besga. Computer aided diagnosis system for Alzheimer Disease using brain Diffusion Tensor Imaging features selected by Pearson's correlation. *Neuroscience letters*, 502(3):225–229, 2011.

[130] R.M. Gray. Vector quantization. *IEEE ASSP*, 1:4–29, 1984.

[131] R.M. Gray and D.L. Neuhoff. Quantization. *IEEE Transactions on Information Theory*, 44:2325–2384, oct 1998.

[132] E.I. Grigolyuk and V.I. Shalashilin. *Problems of nonlinear deformation: the continuation method applied to nonlinear problems in solid mechanics.* Kluwe, 1991.

[133] S. Grossberg. Adaptive pattern classification and universal recording, i: Parallel development and coding of neural feature detectors. *Biological Cybernetics*, 23:121–134, 1976.

[134] J.A. Gualtieri and S. Chettri. Support vector machines for classification of hyperspectral data. In *Proc. Geosci. Rem. Sens. Symp., IGARSS*, volume 2, pages 813–815, 2000.

[135] K.M. Gutzmann. Combinatorial optimization using a continuous state boltzmann machine. In *Proc. IEEE Int. Conf. Neural Networks*, volume III, pages 721–734, San Diego, CA, 1987.

[136] L. O. Hall, I. B. Ozyurt, and J. C. Bezdek. Clustering with a genetically optimized approach. *IEEE Transactions on Evolutionary Computation*, 3(2):103–112, 1999.

[137] R.M. Haralick and L.G Shapiro. *Computer and robot vision*. Addison-Wesley, 1992.

[138] J. Hartigan. *Clustering Algorithms*. Wiley, 1975.

[139] E.J. Hartman, J.D. Keeler, and J.M. Kowalski. Layered neural networks with gaussian hidden units as universal approximations. *Neural Computation*, 2:210–215, 1990.

[140] S. Haykin. *Neural Networks: A comprehensive foundation*. IEEE press, New York: Macmillan Coll. Pub. Co, 1994.

[141] P. Heckbert. Color image quantization for frame-buffer display. *Computer Graphics*, 16(3):297–307, 1980.

[142] J.A. Hertz, A.S. Krogh, and R.G. Palmer. *Introduction to the theory of neural computation*. Addison Wesley, 1991.

[143] M. Heywood and P. Noakes. A framework for improved training of sigma-pi networks. *IEEE trans. Neural Networks*, 6(4):893–903, 1995.

[144] G.E. Hinton. Connectionist learning procedures. *Artificial Intelligence*, 40:185–234, 1989.

[145] G.E. Hinton. Deterministic boltmann learning performs steepest descent in weight-space. *Neural Computation*, 1:143–150, 1989.

[146] J.H. Holland. *Adaptation in Natural and Artificial Systems*. The Univ. of Michigan Press, 1975.

[147] M.A Jacobs, R.A. Knight, H. Soltanian-Zadeh, Z.G. Zheng, A.V. Goussev, D.J. Peck, J.P. Windham, and M. Chopp. Unsupervised segmentation of multiparameter mri in experimental cerebral ischemia with comparison to t2, diffusion, and adc mri parameters and histopatological validation. *Magnetic Resonance Imaging*, 11:425–37, 2000.

[148] A. K. Jain and R.C. Dubes. *Algorithms for clustering data*. Prentice Hall, 1988.

[149] A.K. Jain. *Fundamentals of digital image processing*. Englewood-Clliffs: Prentice-Hall,, 1989.

[150] A.K. Jain and A. Vailaya. Image retrieval using color and shape. *Patt. Recog.*, 29(8):1233–1244, 1996.

[151] D. R. Jones and M.A. Beltrano. Clustering with genetic algorithms. Technical report, Operating Sciences Dept., General Motors Res. Lab., Warren, Mich., 1990.

[152] G. Joy and Z. Xiang. Center cut for color image quantization. *The Visual Computer*, 10:62–66, 1993.

[153] J. Jun, K. Kim, and E. Cha. Vector quantization using an improved competitive learning neural network for color images. In E.A. Sharon and E.J. Olmos, editors, *11th Annual simulators conference, Simulators XI*, pages 466–474. Society for Computer Simulation, 1994.

[154] M. Junghans, A. Leich, and H.-J. Jentschel. Lucas-kanade algorithm with gnc. In *7th International Conference on Signal Processing*. Aug. 31-Sept 4, 2004, Beijing, China, 2004.

[155] C.E. Kahn. Decision aids in radiology. *Radiol Clin North Am*, 34:607–628, 1996.

[156] M.S. Kankanhalli, B.M. Mehtre, and J.K. Wu. Cluster based color matching for image retrieval. *Pattern Recognition*, 29(4):701–708, 1996.

[157] N. B. Karayiannis and A.N. Venetsanopoulos. *Artificial Neural Networks: Learning algorithms, performance evaluation and applications*. Kluwer Acad. Pub., Norwell, 1993.

[158] N.B. Karayiannis. A methodology for constructing fuzzy algorithms for learning vector quantization. *IEEE trans. Neural Networks*, 8(3):505–518, 1997.

[159] N.B. Karayiannis, J.C. Bezdek, N.R. Pal, R.J. Hathaway, and Pin-I Pai. Repairs to glvq: a new family of competitive learning schemes. *Neural Networks, IEEE Transactions on*, 7(5):1062 –1071, sep 1996.

[160] J. M. Karlholm. Associative memories wih short-range higher order coupling. *Neural Networks*, 6:409–421, 1993.

[161] S. Kaski. *Data exploration using Self-Organizing Maps*. PhD thesis, Helsinki University of Technology, Neural Networks Research Centre, Espoo, Finland, 1997.

[162] A. Kelemen, G Székely, and G. Gerig. Elastic model-based segmentation of 3d neuroradiological data sets. *IEEE Trans. Med. Imaging*, 18:828–39, 1999.

[163] F.Z. Ketaff and J.P. Asselin de Beauville. Genetic and fuzzy based clustering. In *Proc. 5th Conf. Int. Fed. Class. Soc.*, pages 100–103, 1996.

[164] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[165] S. Kirpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 20:671–680, 1983.

[166] M.I. Kohn, N.K. Tanna, G.T. Herman, S.M. Resnick, P.D. Mozley, A. Zimmerman R.A. Gur, R.E. Alavi, and R.C. Gur. Analysis of brain and cerebrospinal fluid volumes with mr imaging. *Radiology*, 178(1):115–122, 1991.

[167] T. Kohonen. Clustering, taxonomy, and topological maps of patterns. In *Proc. 6th Int. Conf. Pattern Recognition*, pages 114–128, Munich, 1982.

[168] T. Kohonen. *Self Organization and Associative memory*. Springer Verlag, 1989.

[169] T. Kohonen. Learning vector quantisation and the self-organising map. In M. Taylor, editor, *Theory and Applications of Neural Network (3ed)*. Springer-Verlag, 1992.

[170] T. Kohonen. The self-organising map. *Neurocomputing*, 21:1–6, 1998.

[171] T. Kohonen. *Self-Organizing Maps*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001.

[172] B. Kosko. Stochastic competitive learning. *IEEE trans Neural Networks*, 2:522–529, 1991.

[173] B. Kosko. *Neural Networks and Fuzzy Systems*. Prentice Hall, 1992.

[174] H.K. Kushner and D.S. Clark. *Stochastic approximation for constrained and unconstrained systems*. Springer Verlag, Berlin, 1978.

[175] K. Kusuzaki, H. Shinjo, H. Murata, H. Takeshita, S. Hashiguchi, T. Nozaki, K. Emoto, T. Ashihara, and Y. Hirasawa. Hashiguchi s, nozaki t, emoto k, ashihara t, hirasawa y. (2000) relationship between doxorubicin binding ability and tumor volume decrease after chemotherapy in adult malignant soft tissue tumors in the extremities. *Anticancer Res.*, 20(5C):3813–6, 2000.

[176] R. Lancini and S. Tubaro. Adaptive vector quantization for picture coding using neural networks. *IEEE trans Comm.*, 43(2/3/4):534–544, 1995.

[177] S.L. Lauritzen. Lectures on contingency tables. Technical report, Inst. Elect. Sys., Dept. Math. Comp. Science, Univ. Aalborg (Denmark), 1989.

[178] C.H. Lee, J.S. Kim, and K.H. Park. Automatic human face location in a complex background using motion and color information. *Patt. Recog.*, 29(11):1877–1889, 1996.

[179] B. Lenze. How to make sigma-pi neural networks perform perfectly on regular training sets. *Neural Networks*, 7(8):1285–1293, 1994.

[180] Q. Li and P. F. Swaszek. One-pass vector quantizer design by sequential pruning of the training data. In *Proceedings of the 1995 International Conference on Image Processing*, volume 3, pages 3105–, Washington, DC, USA, 1995. IEEE Computer Society.

[181] S.Z. Li. *Markov Random Field Modeling in Image Analysis*. Advances in Pattern Recognition. Computer Science Workbench, 3 edition, 2009.

[182] S.J. Liao. *Beyond Perturbation: Introduction to the Homotopy Analysis Method*. Boca Raton: Chapman & Hall/ CRC Press, 2003.

[183] C.T. Lin and C.S.G. Lee. A multi-valued boltzmann machine. *IEEE trans. Systems, Man and Cybernetics*, 25(4):660–669, 1995.

[184] S.H Lin, S.Y. Kung, and L.J. Lin. Face recognition/detection by probabilistic decision-based neural network. *IEEE Trans Neural Networks*, 8(1):114–132, 1997.

[185] T.S. Lin and L.W. Chang. Fast color image quantization with error diffusion and morphological operations. *Signal Processing*, 43:293–303, 1995.

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

[187] E. Litmann and H. Ritter. Adaptive color segmentation - a comparison of neural and statistical methods. *IEEE Trans Neural Networks*, 8(1):175–185, 1997.

[188] C.B. Lucasius, A.D. Dane, and G. Kateman. On k-medoid clustering of large data sets with the aid of a genetic algorithm: background, feasibility and comparison. *Analytica Chimica Acta*, 282:647–669, 1993.

[189] S. Luchian, H. Luchian, and M. Petriuc. Evolutionary automated classification. In *Proc 1st IEEE Conf. Evol. Comp.*, pages 585–588, 1994.

[190] V.A. Magnotta, D. Heckel, N.C. Andreasen, T. Cizadlo, P.W. Corson, J.C. Ehrhardt, and W.T.C. Yuh. Measurement of brain structures with artificial neural networks: two- and three-dimensional applications. *Radiology*, 211:781–90, 1999.

[191] C. Malsburg. Self organization of orientation sensitive cells in the striate cortex. *Kybernetic*, 14:85–100, 1973.

[192] J. Mao and A.K. Jain. A self-organizing network for hyperellipsoidal clustering. *IEEE Trans. on Neural Networks*, 7(1):16–29, 1996.

[193] T. Martinetz, S. Berkovich, and K. Schulten. Neural-gas network for vector quantization and his application to time series prediction. *IEEE trans. Neural Networks*, 4(4):558 – 569, 1993.

[194] T. Mcinerney and D. Terzzopoulos. Deformable models in medical images analysis: a survey. *Medical Image Analysis*, 1:91–108, 1996.

[195] J. Mendel and L.X. Wang. Identification of moving -average systems using higher-order statistics and learning. In B. Kosko, editor, *Neural Networks for Signal Processing*, pages 91–120. Prentice-Hall, 1993.

[196] C.E. Metz. Some practical issues of experimental design and data analysis in radiological roc studies. *Invest Radiology*, 24:234–45, 1989.

[197] Z. Michalewicz. Evolutionary computation: practical issues. In *IEEE ICEC'96*, pages 30–39, 1996.

[198] I.R. Moraczewski, W. Borkowski, and A. Kierzek. Clustering geobotanical data with the use of a genetic algorithm. *COENOSES*, 10(1):17–28, 1995.

[199] B.K. Natarajan. Filtering random noise via data compression. In *Proc. IEEE Data Compression Conference*, pages 60–69, Snowbird, Utah, 1993.

[200] B.K. Natarajan. Filtering random noise from deterministic signals via data compression. *IEEE Trans. on Signal Processing*, 43(11):2595–2605, 1995.

[201] B.K. Natarajan, K. Konstantinides, and C. Herley. Occam filters for stochastic sources with application to digital images. *IEEE Trans. on Signal Processing*, 46(5):1434–1438, 1998.

[202] M. Nielsen. Graduated nonconvexity by functional focusing. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(5):521–525, 1997.

[203] M. Nikolova. Markovian reconstruction using a gnc approach. *Image Processing, IEEE Transactions on*, 8(9):1204–1220, Sep 1999.

[204] M. Nikolova, J. Idier, and A. Mohammad-Djafari. Inversion of large-support ill-posed linear operators using a piecewise gaussian mrf. *IEEE Transactions on Image Processing*, 7(4):571–585, 1998.

[205] M. Nikolova and A. Mohammad-Djafari. Discontinuity reconstruction from linear attenuating operators using the weak-string model. In *European Signal Proc. Conf. (EUSIPCO)*, pages 1062–1065, Sep 1994.

[206] M. Nikolova, M.K. Ng, S.Q. Zhang, and W.K. Ching. Efficient reconstruction of piecewise constant images using nonsmooth nonconvex minimization. *SIAM J. Imaging Sciences*, 1(1):2–25, 2008.

[207] M.T. Orchard and C.A. Bouman. Color quantization of images. *IEEE Trans. on Signal Processing*, 39(12):2677–2690, 1991.

[208] J.M. Ortega and W.C. Rheinboldt. *Iterative solution of nonlinear equations in several variables*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[209] N.R. Pal, J.C. Bezdek, and E.C.K. Tsao. Generalized clustering networks and kohonen's self-organizing scheme. *IEEE Trans. Neural Networks*, 4:549–557, 1993.

[210] T. Pappas. An adaptive clustering algorithm for image segmentation. *IEEE trans. Signal Proc.*, 40(4):901–913, 1992.

[211] D.C. Park and I. Dagger. Gradient based fuzzy c-means (gbfcm) algorithm. In *Proc ICCNN'94*, volume 3, pages 1626–1631, 1994.

[212] J. Park and J.W. Sandberg. Universal approximation using radial basis function networks. *Neural Computation*, 3:246–257, 1991.

[213] L. Parra and G. Deco. Continuous boltzmann machine with rotor neurons. *Neural Networks*, 8(3):375–385, 1995.

[214] D.W. Patterson. *Artificial neural networks. Theory and applications*. Prentice Hall PTR, 1998.

[215] S.J. Perantonis and P.J.G. Lisboa. Translation, rotation and scale invariant pattern recognition by high-order neural networks and moment classifiers. *IEEE Trans. Neural Netw.*, 3(2):241–251, 1992.

[216] B.S. Peterson, P.A. Feineigle, L.H. Staib, and J.C. Gore. Automated measurement of latent morphological features in the human corpus callosum. *Human Brain Mapp*, 12:232–45, 2001.

[217] C. Peterson and J.R. Anderson. A mean field algorithm for neural networks. *Complex Systems*, 1:995–1019, 1987.

[218] C. Peterson and E. Hartman. Explorations of the mean field theory learning algorithm. *Neural Networks*, 2(6):475–494, 1989.

[219] C. Peterson and B. Soderberg. A new method for mapping optimization problems onto neural networks. *Int. Journal Neural Systems*, 1(1):3–22, 1989.

[220] G. Pinkas. Energy minimization and satisfiability of propositional logic. In D. S. Touretzky, J. L. Elman, T. J. Sejnowski, and G. E. Hinton, editors, *Proc. Connectionist Models Summer School*. 1990.

[221] W.K. Pratt. *Digital Image Processing*. Wiley, 1991.

[222] B. Raducanu and M. Graña. An approach to face localization based on signature analysis. In *Proc. AVSP Workshop on Audiovisual Speech Processing*, Rhodes (Greece), 1997.

[223] A. Rangarajan and R. Chellappa. Generalized graduated non-convexity algorithm for maximum a posteriori image estimation. In *Proc. 10th ICPR*, pages 127–133, Atlantic Ciyt, N.J., USA, June 1990.

[224] S.L. Richter and R.A. DeCarlo. Continuation methods: Theory and applications. *IEEE Trans. Automatic control*, 28:660–665, 1983. Circuits and Systems, IEEE Transactions on, 30(6), pp.347 - 352.

[225] B.D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[226] Gerhard Ritter and Paul Gader. Fixed points of lattice transforms and lattice associative memories. volume 144 of *Advances in Imaging and Electron Physics*, pages 165 – 242. Elsevier, 2006.

[227] G.X. Ritter and L. Iancu. Single layer feedforward neural network based on lattice algebra. In *Neural Networks, 2003. Proceedings of the International Joint Conference on*, volume 4, pages 2887 – 2892 vol.4, jul. 2003.

[228] G.X. Ritter and L. Iancu. A morphological auto-associative memory based on dendritic computing. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 2, pages 915 – 920 vol.2, jul. 2004.

[229] G.X. Ritter, L. Iancu, and G. Urcid. Morphological perceptrons with dendritic structure. In *Fuzzy Systems, 2003. FUZZ '03. The 12th IEEE International Conference on*, volume 2, pages 1296 – 1301 vol.2, may. 2003.

[230] G.X. Ritter and G. Urcid. Lattice algebra approach to single-neuron computation. *Neural Networks, IEEE Transactions on*, 14(2):282 – 295, mar. 2003.

[231] H. Ritter, T. Martinetz, and K. Schulten. *Neural computation and self-organizing maps: an introduction*. MA: Addison-Wesley, Reading, 1992.

[232] G. Rizzo, P. Scifo, M. Gilardi, V. Bettinardi, F. Grassi, S. Cerutti, and F. Fazio. Matching a computerized brain atlas to multimodal medical images. *Neuroimage*, 6:59–69, 1997.

[233] M.C. Robini, A. Lachal, and I.E. Magnin. A stochastic continuation approach to piecewise constant reconstruction. *IEEE Transactions On Image Processing*, 16(10):2576–2589, 2007.

[234] A.J. Robinson. *Dynamic Error Propagation Networks*. PhD thesis, Cambridge University, Engineering Department, 1989.

[235] A.J. Robinson and F. Fallside. A dynamic connectionist model for phoneme recognition. In *Proceedings of nEuro'88*, 1988.

[236] K. Rose, E. Gurewitz, and G.C. Fox. Vector quantization by deterministic annealing. *IEEE trans. Inf. Theory*, 38(4):1249–1257, 1992.

[237] H.A. Rowley, S. Baluja, and T. Kanade. Human face detection in visual scenes. Technical Report CMU-CS-95-158R, Comp. Sci., Carnegie Mellon Univ., 1995.

[238] H.A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE PAMI*, 20(1):23–38, 1998.

[239] M. Rudin, N. Beckmann, R. Porszasz, T. Reese, D. Bochelen, and A. Sauter. In vivo magnetic resonance methods in pharmaceutical research: current status and perspectives. *NMR Biomed*, 12:69–97, 1999.

[240] J. Ruiz-Cabello, J Regadera, C. Santisteban, M. Graña, R. Pérez de Alejo, I. Echave, P. Avilés, I. Rodríguez, I. Santos, D. Gargallo, and M. Cortijo. Monitoring acute inflammatory processes in the mouse muscle by mr imaging and spectroscopy: a comparison with pathological results. *NMR Biomed*, 15:204–14, 2002.

[241] J. Rynkiewicz. Relation between the som algorithm and the distortion measure. In *Proc. WSOM'2005*, Paris, F, 2005.

[242] N. Saeed. Magnetic resonance image segmentation using pattern recognition, and applied to image registration and quantitation. *NMR Biomed*, 11:157–67, 1998.

[243] F. Samaria and F. Fallside. Face identification and feature extraction using hmm. In *Proceedings of the 2nd IEEE Workshop on Applications of Computer Vision*, Sarasota (Florida), 1994.

[244] L. Saul and M.I. Jordan. Learning in boltzmann trees. *Neural Computation*, 6:1174–1184, 1994.

[245] A. Savio, M. García-Sebastián, M. Graña, and J. Villanúa. Results of an adaboost approach on alzheimer's disease detection on mri. *Bioinspired applications in Artificial and Natural Computation. J. Mira, J. M. Ferrández, J.R. Alvarez, F. dela Paz, F.J. Tolede (Eds.) LNCS 5602*, pages 114–123, 2009.

[246] A. Savio, M. García-Sebastián, C. Hernández, M. Graña, and J. Villanúa. Classification results of artificial neural networks for alzheimer's disease detection. *Intelligent Data Engineering and Automated Learning- IDEAL 2009, Emilio Corchado, Hujun Yin (eds) LNCS 5788*, pages 641–648, 2009.

[247] A. Savio, M.T. Garcia-Sebastian, D. Chyzhyk, C. Hernandez, M. Graña, A. Sistiaga, A. Lopez de Munain, and J. Villanua. Neurocognitive disorder detection based on feature vectors extracted from VBM analysis of structural MRI. *Computers in Biology and Medicine*, 41:600–610, 2011.

[248] T.J. Sejnowski. Higher order boltzmann machines. In Denker, editor, *Proc. conf. Neural Networks for computing AIP 151*, pages 398–403, 1986.

[249] I.K. Sethi. A fast algorithm for recognizing nearest neighbors. *IEEE trans. Syst. Man Cyb.*, 11:245–248, 1981.

[250] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[251] D. Shen, S.M. Moffat, S.and Renick, and C. Davatzikos. Measuring size and shape of the hippocampus in mr images using a deformable shape model. *Neuroimages*, 15:422–34, 2002.

[252] H. Soltanian-Zadeh, D.J. Peck, J.P. Windham, and T. Mikkelsen. Brain tumor segmentation and characterization by pattern analysis of multispectral nmr images. *NM*, 11:201–8, 1998.

[253] R.K. Srihari. Automatic indexing and content based retrieval of captioned images. *Computer*, 28(9):49–56, 1995.

[254] D.G. Stork and M.E. Hennecke. *Speechreading by Humans and Machines*. Springer Verlag, 1996.

[255] Q.B. Sum, W.M. Huang, and J.K. Wu. Face detection based on color and local symmetry information. In *Proc Int. Conf on Autom. Face and Gesture recognition,*, pages 130–135, Napa(Japan), 1998. IEEE press.

[256] K.K. Sung and T. Poggio. Example-based learning for view-based human face detection. Technical Report 1521, A.I.Memo MIT, 1994.

[257] S. Sunthakar and V.A. Jaravine. Invariant pattern recognition using high-order neural networks. In *Proc. SPIE*, volume 1826, pages 160–167, 1992.

[258] M.J. Swain and D.H. Ballard. Color indexing. *Int J Computer Vision*, 7(1):11–32, 1991.

[259] R. Szeliski and D. Terzopoulos. Visual reconstruction for researchers. Image and Vision Computing, 7(4), 308-309, November 1989. A review of Blake and Zisserman's Visual Reconstruction, MIT Press, 1987.

[260] Antonio J. Tallón-Ballesteros and César Hervás-Martínez. A two-stage algorithm in evolutionary product unit neural networks for classification. *Expert Syst. Appl.*, 38:743–754, January 2011.

[261] J.G. Taylor and S. Coombes. Learning higher order correlations. *Neural Networks*, 6:423–427, 1993.

[262] J.C. Terrillon, M. David, and S. Akamatsu. Automatic detection of human face in natural scene images by use of a skin color model and of invariant moments. In *Proc. Int. Conf. on Autom. Face and Gesture recognition*, pages 112–117, Napa(Japan), 1998.

[263] S. B. Thrun, J. Bala, E. Bloedorn, I. Bratko, B. Cestnik, J. Cheng, K. De Jong, S. Dzeroski, S. E. Fahlman, D. Fisher, R. Hamann, K. Kaufman, S. Keller, I. Kononenko, J. Kreuziger, R.S. Michalski, T. Mitchell, P. Pachowicz, Y. Reich, H. Vafaie, W. Van De Welde, W. Wenzel, J. Wnek, and J. Zhang. The monk's problems a performance comparison of different learning algorithms. Technical Report CMU-CS-91-197, Carnegie Melon Univ., 1991.

[264] M. E. Tipping. The Relevance Vector Machine. In S. A. Solla, T. K. Leen, and K.-R. Muller, editors, *Advances in Neural Information Processing Systems 12*, pages 652–658. MIT Press, 2000.

[265] M. E. Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, 2001.

[266] M. E. Tipping. Bayesian inference: An introduction to principles and practice in machine learning. In U. von Luxburg O. Bousquet and G. Rätsch, editors, *Advanced Lectures on Machine Learning*, pages 41–62. Springer-Verlag New York, Inc., 2004.

[267] L. Tomassini. *Apprentissage d'une representation statistique et topologique d'un environment.* PhD thesis, Ecole Nationale Superieure de l'Aeronautique et de l'Espace., 1992.

[268] J.T. Tou and R.C. Gonzalez. *Pattern recognition principles.* Addison-Wesley, Reading,, 1974.

[269] Hsing-Chih Tsai. Hybrid high order neural networks. *Appl. Soft Comput.*, 9:874–881, June 2009.

[270] E.C.K. Tsao, J.C. Bezdek, and N.R. Pal. Fuzzy kohonen clustering networks. *Pattern Recognition*, 27(5):757–764, 1994.

[271] M. Turk and A. Pentland. Eigenfaces for recognition. *J. Cognitive Neuroscience*, 3(1):71–86, 1991.

[272] T. Uchiyama and M.A. Arbib. Color image segmentation using competitive learning. *IEEE Ttrans.on Pattern Analysis and Machine Intelligence*, 16(12):1197–1206, 1994.

[273] N.B. Venkateswarlu and P.S.V.S.K. Raju. Fast isodata clustering algorithms. *Pattern Recognition*, 25(3):335–342, 1993.

[274] M. Verleysen and K. Hlavackova. An optimized rbf network for approximation of functions. In *Proceedings of the European Symposium on Artificial Neural Networks, ESANN'94*, 1994.

[275] J. M. Vincent, D.J. Myers, and R.A. Hutchinson. Image feature location in multi-resolution images using a hierarchy of multilayer perceptrons. In R. Lingaard, D.J. Myers, and C. Nightingale, editors, *Neural Networks for Vision, Speech and Natural language*, pages 13–29. Chapman & Hall, London, 1992.

[276] V.V. Vinod, S. Chaudhury, J. Mukherjee, and S. Ghose. A connectionist approach for clustering with applications in image analysis. *IEEE Trans. on. Sys. Man & Cyb.*, 24(3):365–383, 1994.

[277] A.G. Voloboj. The method of dynamic palette construction in realistic visualization systems. *Comp. Graph. Forum,*, 12:289–296, 1993.

[278] H. Wacker, editor. *Continuation Methods.* Academic Press Inc, Nv 1978.

[279] J. B. Waite. Training multilayer perceptrons for facial feature location : a case study. In *Neural Networks for Vision, Speech and Natural language*, pages 30–49. Chapman & Hall, London, 1992.

[280] E. Wasserstrom. Numerical solutions by the continuation method. *SIAM Review*, 15:89–119, 1973.

[281] L.T. Watson. Globally convergent homotopy methods. In Christodoulos A. Floudas and Panos M. Pardalos, editors, *Encyclopedia of Optimization 2nd ed*, pages 1272–1277. Springer, 2009.

[282] G. Winkler. *Image analysis, random fields and dynamic Monte Carlo methods*. Springer-Verlag, 1995.

[283] X. Wu. Efficient statistical computations for optimal color quantization. In J. Arvo, editor, *Graphics Gems II*, pages 126–133. Academic Press Professional, 1991.

[284] Y Wu, M.L. Giger, K. Doi, C.J. Vyborni, R.A. Schmidt, and C.E. Metz. Artificial neural networks in mammography: applications to decision making in the diagnosis of breast cancer. *Radiology*, 187:81–87, 1993.

[285] Z. Xiang. Color image quantisation by minimizing the maximum intercluster distance. *ACM Trans Graphics*, 16(3):260–276, 1997.

[286] M. Xie. Automatic feature matching in uncalibrated stereo vision through the use of color. *Rob Aut Sys*, 21:355–365, 1997.

[287] L. Xu, A. Krzyzak, and A. Yuille. On radial basis function nets and kernel regression: statistical consistency, convergence rates and receptive field size. *Neural Networks*, 7:609–628, 1994.

[288] E. Yair, K. Zeger, and A. Gersho. Competitive learning and soft competition for vector quantization. *IEEE Trans. Sign. Proc.*, 40(2):294–308, 1992.

[289] Y. Yokoo and M. Hagiwara. Human faces detection method using genetic algorithms. In *Proc. ICEC96*. IEEE press, 1996.

[290] K.C. Yow and R. Cipolla. Finding initial estimates of human face location. Technical Report TR- 239, University of Cambridge, 1995.

[291] W. I. Zangwill and C. B. Garcia. *Pathways to Solutions, Fixed Points, and Equilibria*. Prentice Hall, Englewood Cliffs, N.J. 07632, 1981.

[292] K. Zeger, J. Vaisey, and A. Gersho. Globally optimal vector quantizer design by stochastic relaxation. *IEEE Trans. Sign. Proc.*, 40:310–322, 1992.

[293] R.S. Zemel, C.K.I. Willliams, and M.C. Mozer. Directional-unit boltzmann machines. In S. J. Hanson, J.D. Cowan, and C.L. Giles, editors, *Adv. Neural Information Processing Systems*, volume 5, pages 172–179. Morgan Kauffamn, 1993.

[294] X. Zeng, L. Staib, R. Schultz, and J. Duncan. Segmentation and measurement of the cortex from 3d mr images using coupled-surfaces propagation. *IEEE Trans. Med. Imaging*, 18:927–37, 1999.

[295] Z. Zhang and V.K. Wei. An on-line universal lossy data compression algorithm via continuous codebook refinement - part i: Basic results. *IEEE Trans Inf Theory*, 42(3):822–836, 1996.

[296] Sun Zhanquan, Xi Guangcheng, and Yi Jianqiang. Unsupervised high order boltzmann machine and its application on medicine. *International Conference on Natural Computation*, 1:343–347, 2007.

[297] D. Zhong, S. Chang, and J.R. Smith. Differential compression and optimal caching methods for content-based image search systems. In *Proc Multimedia Storage and Archiving Systems IV, Proc SPIE 1999*, volume 3846, pages 413– 421, 1999.

[298] S. Zhong and J. Ghosh. A unified framework for model-based clustering. *Journal of Machine Learning Research*, 4:1001–1037, 2003.

[299] A. Zijdenbos, B. Dawant, R. Margolin, and A. Palmer. Morphometric analysis of white matter lesions in mr images: method and validation. *IEEE Trans. Med. Imaging*, 13:716–24, 1994.