Universidade do Porto
Faculdade de Engenharia
FEUP

Data Classification with Neural Networks and Entropic Criteria

Jorge Manuel Fernandes dos Santos

Thesis submitted to the Engineering University of Porto, Portugal, for the partial fulfillment of the requirements for the degree of Doctor of Philosophy made under the supervision of Doctor Joaquim Marques de Sá, Full Professor at the Faculty of Engineering of the University of Porto, Portugal and Doctor Luís A. Alexandre, Assistant Professor at the Beira Interior University, Covilhã, Portugal.

January 2007

... for my daughter Sofia

and my son Lucas

Abstract

The concept of entropy and related measures has been applied in learning systems since the 1980s. Several researchers have applied entropic concepts to independent component analysis and blind source separation. Several previous works that use entropy and mutual information in neural networks are basically related to prediction and regression problems.

In this thesis we use entropy in two different perspectives: first as a cost function in neural networks (multi-layer perceptrons - MLP) for supervised classification problems; and second, as the basis for a new entropic measure for unsupervised classification (clustering).

Related with the new entropic cost function we show in the present work how to use Rényi's quadratic entropy of the errors between the output of an MLP and the desired targets as a cost function in classification problems: the Error Entropy Minimization Algorithm (EEM). We present several optimization procedures for this algorithm, namely: how to use an appropriate adaptive learning rate; how to tune the smoothing parameter when performing entropy gradient computation; and, the conditions to apply a batch-sequential algorithm for training with the EEM algorithm.

Regarding unsupervised classification, we present a clustering algorithm based on a new entropic dissimilarity matrix. This matrix is the basis for a new clustering process, based on layered entropic subgraphs, which we call LEGClust. We also used this algorithm to perform task decomposition in modular neural networks for classification problems.

Resumo

Desde os anos 80 que o conceito de entropia e medidas relacionadas se aplicam a sistemas de aprendizagem tendo-se assistido à aplicação destes a problemas de *independent component analysis* e *blind source separation*. Trabalhos posteriores que usam a entropia e a informação mútua em redes neuronais estão fundamentalmente relacionados com problemas de regressão e predição.

No trabalho que aqui apresentamos, usamos a entropia de duas formas distintas: primeiro, como função de custo em redes neuronais (perceptrão multi-camada) para problemas de classificação; segundo, como base para um nova medida entrópica para classificação não supervisionada (*clustering*).

Relativamente à nova função de custo, mostramos como usar a entropia quadrática de Rényi dos erros entre a saída do perceptrão multi-camada e as respectivas etiquetas, como função de custo em problemas de classificação: o algoritmo de minimização da entropia do erro (EEM). Apresentamos também um conjunto de optimizações para este algoritmo, nomeadamente: a adopção de uma taxa de aprendizagem variável; a afinação do parâmetro de suavização para estimação do gradiente da entropia; as condições para o uso de um algoritmo "batch-sequential" para treino das redes neuronais com o algoritmo EEM.

No que respeita à classificação não supervisionada, apresentamos um algoritmo de *clustering* baseado numa nova matriz entrópica de dissemelhança que serve como base para um novo processo de *clustering*, ao qual chamamos LEGCLust. Também usamos este novo algoritmo para efectuar a decomposição de tarefas para redes neuronais modulares em problemas de classificação.

Résumé

Le concept d'entropie et les mesures relationnés s'appliquent à des sistémes d'apprentissage depuis les années 80, principalement à des problèmes d'analyse de components independents et de séparation aveugle de sources. Des travaux postérieurs qui usent l'entropie et l'information mutuelle en réseaux de neurones artificiels sont relationnés avec la régression et la prédiction.

Dans cette thèse, on use l'entropie avec deux perpectives distinctes: comme une fonction de coût des réseaux de neurones (perceptron multicouche) pour des problèmes de classification supervisée, et comme base pour une nouvelle mesure entropique pour des problèmes de classification non supervisée.

Relationné avec la nouvelle fonction de coût, on présente ici comme utiliser l'entropie quadratique de Rényi des erreurs entre la sortie du réseaux de neurones et les étiquettes comme fonction de coût: l'algorithme de la minimization de l'entropie de l'erreur (EEM). Nous présentons aussi quelques méthodes d'optimization de l'EEM, à savoir: comme utiliser un coefficient d'apprentissage variable approprié; comme ajuster le facteur d'aplanissage en calculant le gradient de l'entropie; et les conditions pour appliquer un algorithme "batch-sequential" pour l'entrainement du perceptron multicouche avec l'algorithme EEM.

Ce qui concerne la classification non supervisée, nous présentons un algorithme pour "clustering" basé dans une nouvelle matrice de dissimilitude. Cette matrice est la base pour la nouvelle méthode de "clustering" qu'on a nomée LEGClust. On use aussi LEGCLust pour éffectuer la décomposition de tâches dans les réseaux de neurones modulaires pour des problèmes de classification.

Acknowledgements

My first words of recognition go to my supervisors, Professor Marques de Sá and Professor Luís Alexandre for their guidance and advice throughout my PhD studies. I thank Prof. Marques de Sá for having introduced me to the neural networks field and for his pragmatism and valuable comments about my work. It is not everyday that we find someone that we really appreciate to work with. To Prof. Luís Alexandre I would like to thank him for his friendship and for all the support that he has provided me during this three years of study. I thank both of them for giving me the freedom of exploring on my own some of the research subjects.

Some words of acknowledgement go to Professor Mark Embrechts that received me in Troy, New York and provided me some useful insights in classification problems and data mining. The two month visit at Troy allowed me a full time dedication to my work.

I also would like to thank some of my colleagues from the mathematics department of my school, that have incited me to follow my studies, and also to my colleagues at INEB for their everyday friendship and support.

I would like to thank ISEP (Engineering Institute of Porto) and PRODEP for the period of free leave that made possible the accomplishment of this dissertation.

Finally I would like to thank my wife Isabel, my daughter Sofia and my son Lucas, for their constant love and support, specially in the periods of my absence. They are the main reason for my decision to do this "late" PhD.

Contents

List of Figures

List of Tables

Symbols and Abbreviations

Global Symbols

C	number of clusters
n, N	number of elements of a data set
m	vector dimension
$\mathbf{w}$	weight vector
w_i	weight i
w_0	bias
h, h_n	bandwidth or smoothing parameter
E	approximation error
X	data set
i.i.d.	independent and identically distributed
r.v.	random variable

Mathematical and Statistical Symbols

x	variable
x_i	i-th component of vector $\mathbf{x}$
$\mathbf{x}$	vector
$\mathbf{x}^T$	transpose vector
$\mathbf{x}^T\mathbf{y}$	inner product of $\mathbf{x}$ and $\mathbf{y}$
$\mathbf{A}$	matrix

a_{ij}	i-th row, j-th column element of matrix $\mathbf{A}$
$\lvert\mathbf{A}\rvert$	determinant of matrix $\mathbf{A}$
$\mathbf{I}$	identity matrix
$\binom{m}{k}$	combinations of m elements taken k at a time
$\mathbb{R}$	real numbers set
η	learning rate
Σ	covariance matrix
σ	standard deviation
$E[X]$	expected value of a random variable X
$P(.)$	discrete probability
$H(X)$	Entropy of a random variable X

Abbreviations

CTG	Cardiotocography
MLP	Multi-layer Perceptron
MNN	Modular Neural Network
MSE	Mean Squared Error
NN	Neural Network
pdf	Probability Density Function
ROC	Receiver Operating Characteristics
SVM	Support Vector Machine
XOR	Exclusive OR

Chapter 1

Introduction

One can probably state that the modern era of classification theory has its roots on the work of Thomas Bayes and his famous theorem [16]. However, as we know, any kind of progress is based on previous knowledge and new developments wouldn't be possible without the accomplishments of our antecessors. The foundations of classification can be viewed both from a philosophical and mathematical point of view. Plato and Aristotle, who distinguish between an "essential property" (which would be shared by all members in a class) from an "accidental property" (which could differ among members in the class) [42] were perhaps the firsts to attempt a coherent view of questions concerning human understanding that led to classification theory. Later, other philosophers like William of Ockham with the concept that is now known as "Ockham's razor" and René Descartes who added the rigor of mathematics by his strong dependency upon deductive reasoning, provided the basis for the emergence of classification theory to be developed by mathematicians. Given that the Bayes theorem is a ratio of probability density functions, there is a connection between classification and probability theory, whose initial development roughly span the time of Pascal to Laplace. In Fig. 1.1 we can see a time window of the basic foundations of modern classification theory.

Classification of objects is probably one of the most common and ancient

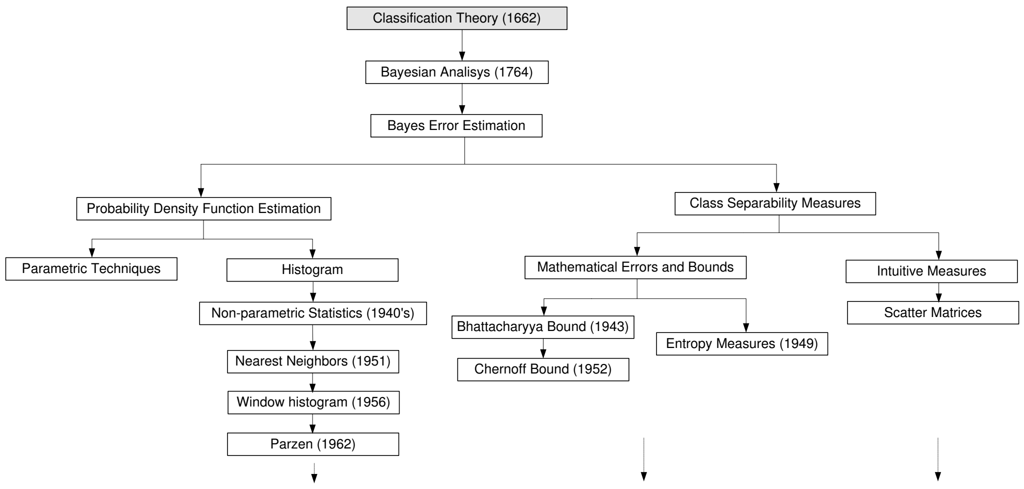


Figure 1.1: Time window of classification theory foundations.

decision tasks performed by humans. It can be seen as the ability of assigning a specific object to a predefined group or class based on a number of observed attributes of that object. The classification process was primarily related to our natural senses: humans recognize or classify objects based on the data acquired by their natural sensors. The technological evolution allowed us to develop sophisticated sensors and the consequent acquisition of more complex signals. On the other hand by using mathematical tools one can transform the original data characteristics and obtain other derived features. In Fig. 1.2 we present the different steps involved in a classification problem. The data collected by the sensors is converted to specific features that are the input for the chosen classification method.

New algorithms and new strategies for classification were necessary due to the emerging of more challenging and computationally demanding applications in several fields such as: bioinformatics; data mining; biometric identification; speech recognition; document, image and video analysis and classification; industrial automation; credit scoring.

Nowadays there are a huge variety of data classification methods. The most

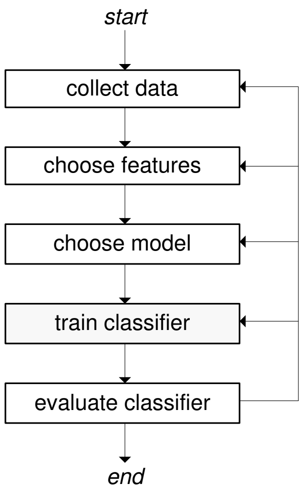


Figure 1.2: Schematic view of the classification process.

Table 1.1: Some of the most used classification methods.

Based on probabilistic rules	Not based on probabilistic rules
Fisher's linear discriminant	K-nearest neighbor
Naive Bayes classifier	Fuzzy logic classifiers
Decision trees	Support vector machines
Bayesian networks	**Neural networks**
Markov models	

common ones are presented in Table 1.1.

Neural networks (NN) have emerged as important tools for data classification. The extensive use of these models has proved that they represent a valid alternative to various conventional classification methods. Neural networks are used in fields such as function approximation, regression analysis, time series prediction, data processing, filtering, compression, blind signal separation or clustering. The advantages of neural networks lie in the following aspects:

1. Neural networks are universal approximators [37].

2. Neural networks are (usually) highly nonlinear machines.

3. Neural networks are adaptive, model-free machines.

4. Neural networks can estimate Bayesian *a posteriori* probabilities.

Neural networks have been successfully applied to real world classification tasks in fields such as medical diagnosis, business, pattern recognition or bio-informatics. They have also been applied to prediction and control problems.

Early works on neural networks were mainly concerned with investigating the mean-square-error (MSE) and other second-order statistics as optimality criteria. This was due to the fact that initial research on linear systems used the second-order optimality criteria because its quadratic performance surface permits to obtain analytical expressions for the optimal solution, allowing theoretical analysis of the learning process. The results obtained with MSE were also very satisfactory when the scientific community started to apply it on non-linear systems. These good results and the belief that the second-order criterion was sufficient (supported by the central limit theorem), made this criterion the main focus of interest for some decades. However, by the arising of more complex problems like those involving blind source separation (BSS) or independent component analysis (ICA), researchers understood that higher order statistics should be used to describe properly these processes.

In 1948, Shannon [174] introduced what is considered one of the most important achievements in communication systems: the concept of information entropy. His work was the foundation of a new research area currently known as *information theory*. This appealing new branch of mathematics attracted several researchers that produced contributions both in theoretical and practical aspects. Alfred Rényi [154, 155] was probably the one producing the most important contribution by showing that Shannon's information theoretic quantities were special cases of a more general family of definitions: Rényi's entropy and Rényi's mutual information.

Although the information measures were originally adopted in communication systems it is such a fundamental concept that it has been widely applied in areas such as physics, chemistry, computer science, neuroscience, economics, biology, psychology and linguistics. The application of Shannon's information theory to learning systems started in the late 1980s when Linsker presented the principle of maximum information preservation (InfoMax) [122] that consists on the maximization of the mutual information between the output and the input of the network so that the information about the input is best preserved in the output. In the 1990's several researchers draw their attention to the application of Shannon's information-theoretic measures to ICA and BSS namely by introducing the principle of maximum entropy and the principle of minimum mutual information [9, 18, 118, 201, 202].

In the late 1990's early 2000's, Príncipe and his co-workers have applied Rényi's entropy and other related optimality criteria to problems of BSS, blind convolution and equalization, feature reduction, ICA and time series prediction [45, 46, 55, 150–152, 197].

1.1 Motivation and Objectives

It was the influence of the above mentioned works of Príncipe, Xu and Erdogmus, that led us to the attempt of applying entropy to data classification problems. Since we worked with neural networks we first tried to use the entropy as cost function in multi-layer perceptrons (MLP). We knew that the usual MSE was not the most appropriate for neural network classification problems since this cost function assumes that the errors (difference between the output of the neural network and the desired targets) are Gaussian distributed [1], and that is definitely not the case in classification problems. By using the new entropic cost function we expected to achieve better results in data classification since we were not limited by the second-order statistics of the MSE.

In a later stage of our work we tried to apply entropy to unsupervised clas-

[1]This Gaussian distributed assumption is related with the maximum likelihood principle and the central limit theorem.

sification (clustering), in the form of a new clustering algorithm. We aimed to develop this new clustering algorithm supported in a new dissimilarity matrix not based on typical distance measures. As our final goal we intended to use the developed algorithms in data classification with Modular Neural Networks (MNN), namely by performing an entropic task decomposition and by using the entropic cost function and the related optimization procedures in classification problems.

1.2 Contributions

The main contributions of this research are:

- The use of Rényi's quadratic entropy of the errors as cost function in classification problems with MLP's, and the Error Entropy Minimization Algorithm (EEM). A theoretical result is presented to this respect [164].
- Several optimization procedures for the EEM algorithm, namely:
  - An appropriate adaptive learning rate [169].
  - Tuning the smoothing parameter when performing entropy gradient computation [167].
  - A batch-sequential algorithm for neural network training with the EEM algorithm [166].
- A new entropic dissimilarity matrix for clustering [168].
- A new clustering process based on the previous matrix, which we called LEGClust [168].
- The application of the LEGClust algorithm to perform task decomposition in modular neural networks for classification problems (illustrated with several experiments) [165].

1.3 Thesis Outline

In Chapter 2 we give an overview of the basis of neural networks and an introduction to the concept of entropy and its estimation. We use this introduction also to present some notation.

In Chapter 3, –Error Entropy Minimization Algorithm–, we present an algorithm for neural network training (namely multi-layer perceptrons) based on an entropic cost function for data classification. We also present some optimization procedures to achieve a more accurate and faster convergence.

Unsupervised classification, or clustering, with entropic criteria is presented in Chapter 4 where, after presenting an introduction to clustering and to the most popular clustering algorithms, we introduce a new clustering algorithm based on subgraphs with an entropic dissimilarity measure.

In Chapter 5 we apply, to classification problems, our new entropic clustering algorithm for the task decomposition phase of modular neural networks.

The conclusions are drawn in Chapter 6, where we also present some future research directions.

In Appendix A the most important characteristics of all the real data sets used in this work are listed (artificial data sets are included in the running text).

A work on human clustering of bi-dimensional artificial data sets is presented in Appendix B. This work served as motivation to develop our new clustering algorithm.

Chapter 2

Definitions and Background

Since one of the purposes of this work is the application of neural networks with entropic criteria to supervised classification, we start by presenting a review on neural networks and on the most important topics related to them, followed by an introduction to entropy, information theory and entropy estimation. In Section 2.1 we present the basics of neural networks and in Section 2.2 the basics of entropy and related concepts.

2.1 Basics of Neural Networks

Artificial Neural Networks or simply Neural Networks, also known as connectionist models, are based on the attempt to mimic our nervous system, namely by using structures with a large number of identical and interconnected computational elements (neurons), trying to achieve similar problem solving strategies as the human brain. Neural networks are model-free machines possessing universal approximation capabilities. A possible definition of a neural network presented by Haykin in [80] is:

> A Neural Network is a massively parallel distributed processor made up of simple processing units, which has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:

1. Knowledge is acquired by the networks from the environment through a learning process.
2. Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.

Neural networks have advantages over classical statistical approaches especially when the training set size is small compared with the dimensionality of the problem to be solved and the underlying data distribution is unknown.

The first mathematical model of a neural network was presented by McCulloch and Pitts [132], a neurophysiologist and a logician, based on their understanding of the nervous system. This was a very simple model of the neuron function. In the following years many researchers, following this model, tried to use computer simulations to apply it. The close contact between engineers, neurophysiologists and even psychologists, made possible some notable progresses in the field. In 1958, Rosenblatt [158] introduced one of the fundamental foundations of neural networks, the Perceptron. The perceptron was able to learn, to connect or associate a given input to a random output unit. It had three layers, with the middle one known as association layer. This innovation became clear when Rosenblatt demonstrated the Perceptron Convergence Theorem [159]. This theorem says that if there is a set of weighted connections of a perceptron, such that the perceptron gives the desired responses for a set of stimulus patterns, then after a finite number of presentations of the stimulus-response pairs and applications of the training procedure, the perceptron will converge to that set of weights which would enable it to respond correctly to each stimulus in the set. Meanwhile in 1960, another system was developed by Widrow and Hoff [194] who employed the Least Mean Square (LMS) learning rule: the ADALINE (ADAptive LInear Element).

In 1969 Minsky and Papert [135], allegedly following a connectionists campaign against the neural network followers, published a book in which they argued that there were a number of fundamental problems with perceptrons. They said that the perceptron was unable to perform certain tasks, such as the calculation of topological function of connectedness (problem of telling connected patterns from disconnected ones). These limitations proved to be particularly significant, as they showed that a perceptron could not learn to evaluate the logical function of exclusive-or (XOR). They also asserted that perceptrons and their possible extensions were a "sterile" direction of research. This caused the stagnation of neural network research and a consequent period of disrepute.

Despite the lack of funding for neural network research in the following years some authors have presented a few new ideas. Examples are the works by Amari [8], Little and Shaw [124], Paul Werbos [193], Hopfield [82, 83], Kohonen [112], Ackley, Hinton and Sejnowski [1], Grossberg [66, 67] and Rumelhart [161]. In fact, Rumelhart's work was one of the most important events in the renaissance of network models. He worked in the back-propagation algorithm for multi-layer perceptrons first introduced by Paul Werbos in his 1974 PhD Thesis. Since then, a lot of research has been done in this field. Neural networks are, nowadays, extensively used in application fields such as Engineering, Economics, Biology, Chemistry, Medicine, Social Sciences, etc.

2.1.1 The Neuron Model

As we said earlier, a neural network is a parallel distributed processor made up of simple processing units. These basic information-processing units are known as *neurons*. A diagram of a neuron is shown in Fig.2.1. This neuron is made of three basic components:

1. A set of connections from the input signals to the summing junction, each one with an associated *weight*. The connection between input signal x_j and neuron k is weighted by w_{kj}. The first subscript of w_{kj} will always refer to the neuron and the second to the input signal origin of the connection. These weights can have any positive or negative value.

2. The summing of the input signals weighted by the respective connections.

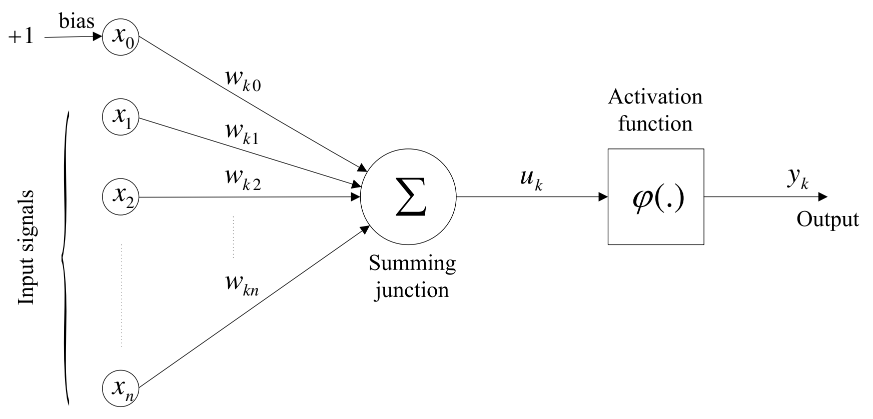


Figure 2.1: A nonlinear model of a neuron.

3. An activation function, also known as *squashing function*, responsible for limiting the amplitude of the output y_k of the neuron. The usual amplitude of the output neuron is a closed interval $[-1, 1]$ or $[0, 1]$.

The extra input signal x_0 is permanently set to unity.

The neuron is described by the following equation:

$$y_k(x) = \varphi(u_k) = \varphi\left[\left(\sum_{j=1}^{n} w_{kj}x_j\right) + w_{k0}\right], \tag{2.1}$$

where $x_1, x_2, ..., x_n$ are the input signals, $w_{k1}, w_{k2}, ..., w_{kn}$ are the weights of neuron k, w_{k0} is the bias, $\varphi(.)$ is the activation function and y_k is the output signal of the neuron. If we define $\mathbf{w}_*$ as vector $(w_{k1}, w_{k2}, ..., w_{kn})^T$ and $\mathbf{x}_*$ as vector $(x_1, x_2, ..., x_n)^T$ we can represent equation 2.1 as:

$$y_k(x) = \varphi(u_k) = \varphi\left(\mathbf{w}_*{}^T\mathbf{x}_* + w_{k0})\right). \tag{2.2}$$

Equation 2.1 can be simplified if one includes w_{k0} in vector $\mathbf{w}_*$. Let $\mathbf{w} = (w_{k0}, w_{k1}, ..., w_{kn})^T$ and $\mathbf{x} = (x_0, x_1, ..., x_n)^T$. The output of the neuron is now defined by:

$$y_k(x) = \varphi(u_k) = \varphi\left(\mathbf{w}^T\mathbf{x}\right). \tag{2.3}$$

The mentioned bias term, w_{k0}, also known as *threshold*, (this bias has nothing to do with the statistical bias) is used to allow the transformation of the linear combination of the input signals and the weights. This transformation determines the position of the hyperplane defined by the linear combination. In Fig. 2.2 we can see an example of a linear decision boundary (hyperplane) in a 2-dimensional input space. The bias defines the position of the plane in terms of its perpendicular distance to the origin.

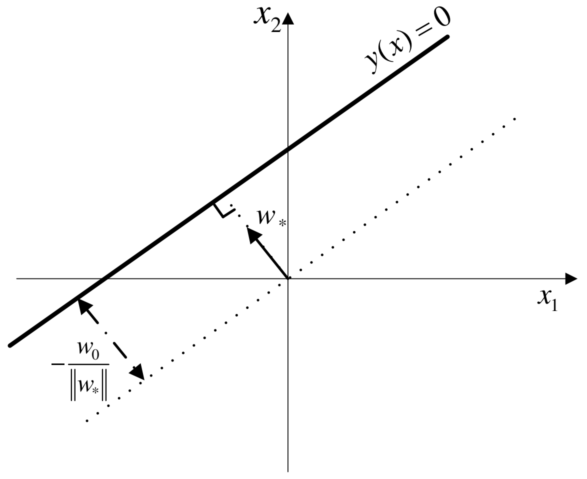


Figure 2.2: The weight vector w_* defines the orientation of the decision plane $y(x) = 0$, while the bias w_0 defines the position of the plane in terms of its perpendicular distance to the origin.

The simple neuron can be viewed as a simple discriminant function. One can combine multiple neurons like in Fig. 2.3, to obtain a multi-class discriminant function for a problem of c classes. In this case we get a combination of decision boundaries that is always simply connected and convex.

2.1.2 Activation Functions

The activation function, $\varphi(u)$, usually a monotonic and bounded function, is just a function that is used to introduce nonlinearity in the network. Examples of activation functions are presented in Fig. 2.4.

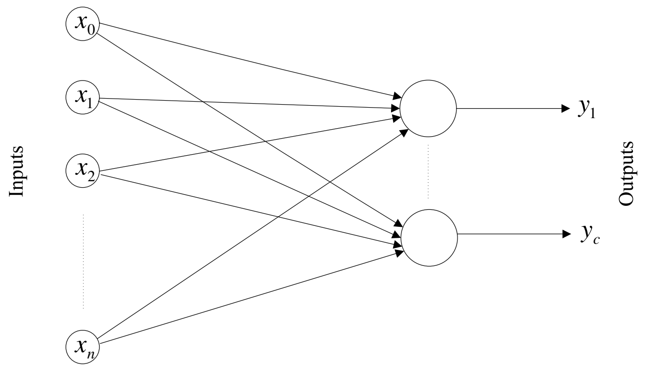
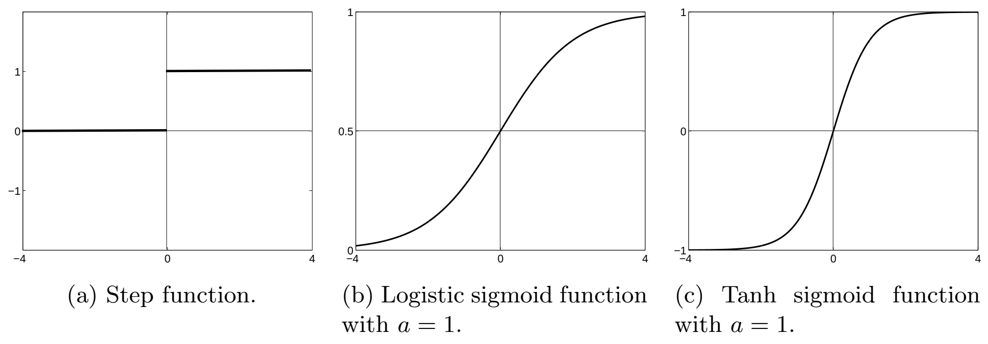


Figure 2.3: A multiple output neural model.



(a) Step function. (b) Logistic sigmoid function with $a = 1$. (c) Tanh sigmoid function with $a = 1$.

Figure 2.4: The three most popular activation functions.

The step function, also known as *Heaviside or threshold function* is defined by:

$$\varphi(u) = \begin{cases} 1 & \text{if} \quad u \geq 0 \\ 0 & \text{if} \quad u < 0 \end{cases} . \tag{2.4}$$

With this activation function the output of the neural network is 1 if u_k is nonnegative, and 0 otherwise. The neuron model with the step function, first introduced by McCulloch-Pits, and posteriorly developed by Rosenblatt, is known as Perceptron.

The logistic sigmoid function, defined as:

$$\varphi(u) = sig(u) = \frac{1}{1+e^{-au}} \tag{2.5}$$

and the *tanh* sigmoid function, defined as:

$$\varphi(u) = tanh(u) = \frac{e^{au} - e^{-au}}{e^{au} + e^{-au}} \ , \tag{2.6}$$

are both S-shaped curves and the two most used activation functions in neural networks. Parameter a controls the slope of the curves. The outputs of the neural networks having logistic and *tanh* activation functions are in the intervals $[0, 1]$ and $[-1, 1]$, respectively. These two activation functions are differentiable in all the domain with derivatives (for $a = 1$):

$$sig\,'(u) = \frac{e^{-u}}{(1+e^{-u})^2} = sig(u)(1 - sig(u)) \tag{2.7}$$

and

$$tanh\,'(u) = \frac{4}{(e^{u}+e^{-u})^2} = 1 - tanh^2(u). \tag{2.8}$$

Both sigmoidal functions possess a linear behavior near the zero crossing and a similar behavior with the step function for higher values of u.

Functions such as *tanh* that produce both positive and negative values tend to yield faster training than functions that produce only positive values such as logistic sigmoid, because of better numerical conditioning. On the other hand, logistic sigmoid activation function allows the output of the neural network to be interpreted as posterior probabilities [21], providing more than a simple classification.

In all our experiments with MLP's we use the *tanh* activation function.

2.1.3 Neural Networks Architectures

The structure of a neural network is related to the way neurons are connected to each other and with the learning algorithm. Depending on these different characteristics we will have two main groups of neural networks: networks with only feedforward connections and networks with both feedforward and feedback connections. In the first group we can include the single layer neural network, the multi-layer neural network and the Kohonen network [112] and in the second group we have the Hopfield networks and other types of so-called recurrent neural networks.

Single-layer and multi-layer neural networks are organized in layers of neurons. The most simple layered neural network is constituted by the input nodes and a layer of output neurons. There are only connections between the inputs and the output layer (not the opposite), hence information is processed from the input to the output, reason for calling them feedforward neural networks. In Fig. 2.3 we represented a feedforward neural network with multiple outputs. This neural network has a single layer of neurons (single-layer feedforward neural network or single-layer perceptron). There are only connections from input nodes to output neurons. The term "single-layer" refers to the layer of the processing neurons. Input nodes are not considered to be a layer because they aren't involved in any processing.

One can have more complex layered neural networks by adding more layers of neurons building a multi-layer feedforward neural network also known as multi-layer perceptron (MLP). An example of such a neural network is depicted in Fig. 2.5. This is a fully connected neural network since a neuron in any layer of the network is connected with all the neurons/nodes of the previous layer. The output signals from the first layer are the inputs for the output layer. The layers between the input and output layers are known as hidden layers (neurons in these layers are called hidden neurons). In this work we use the notation $[a : b : c]$ to denote the structure of a neural network with one hidden layer. In this notation the first parameter, a, designates the number of inputs, the last parameter, c, the number of output neurons and the intermediate parameter, b, the number of hidden neurons in the hidden layer (for more than one hidden layer we have more than one intermediate parameter).

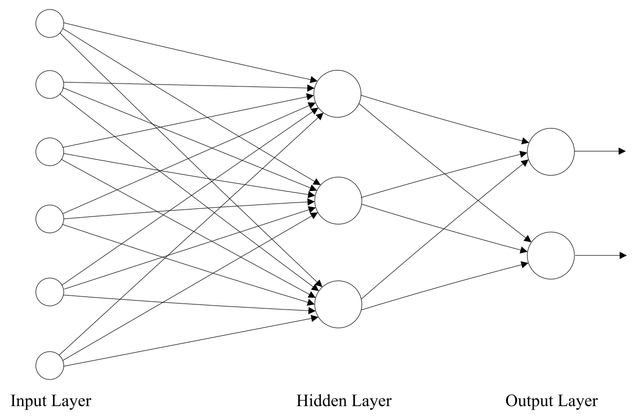


Figure 2.5: A [6:3:2] multi-layer perceptron with one hidden layer.

Multi-layer perceptrons are capable of more complex mappings than single-layer perceptrons. In Fig. 2.6 we show the types of regions that one can get from the use of different kinds of single- and multi-layer perceptrons having threshold activation functions. A single-layer perceptron can only implement a linear discriminant producing an hyperplane as decision boundary. A multi-layer perceptron with one hidden layer is able to generate an open or closed convex region in the input space, whose boundary are segments of hyperplanes. This convex region is obtained with the ability of the hidden layer to perform an AND operation. To get non-convex and/or disjoint regions we must add a layer to perform the OR operation. Multi-layer perceptron with two hidden layers and threshold activation functions are capable of defining arbitrary regions.

The explanation of the mapping capability of multi-layer perceptrons based on the properties of the operation AND and OR is just a simple proof of their ability to map any region of the input space. However, by relaxing some of the conditions, any given arbitrary decision boundary can be approximated arbitrarily close by a two-layer (one hidden layer) network having sigmoidal activation functions [21]. For this reason we will use in all our experiments (if not stated otherwise) this kind of multi-layer perceptrons with one hidden layer and sigmoid

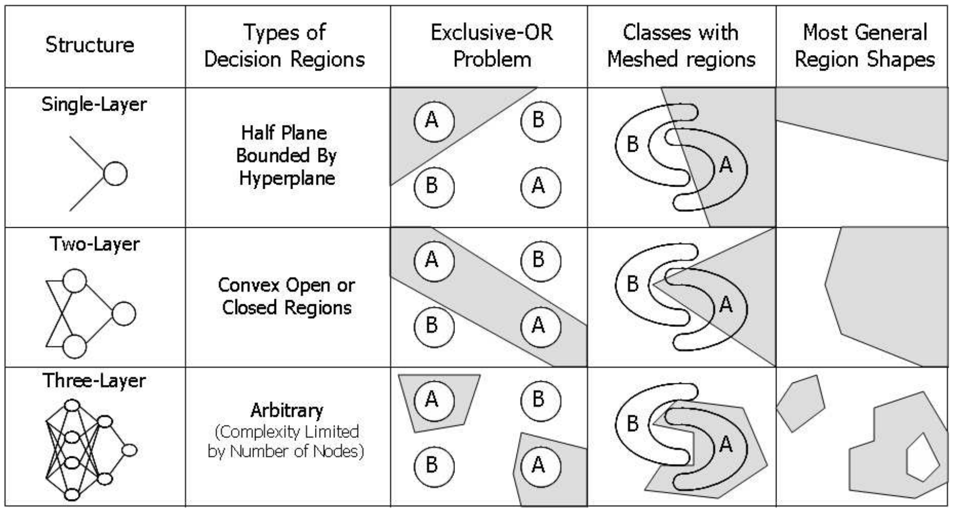

Structure
Types of Decision Regions
Exclusive-OR Problem
Classes with Meshed regions
Most General Region Shapes
Single-Layer
Half Plane Bounded By Hyperplane
Two-Layer
Convex Open or Closed Regions
Three-Layer
Arbitrary (Complexity Limited by Number of Nodes)
A
B

Figure 2.6: Types of decision regions that can be obtained with single- and multi-layer perceptrons with one or two layers of hidden units. (from [123])

activation functions.

In the other group of neural networks the information processing can be operated not only in a feedforward way but also with feedback connections. Information processing can be made from the output to the input, as in the Hopfield networks [82, 83] without self-feedback, or even with a self-feedback, as in recurrent neural networks [7, 101, 195], with time-delay units. These time delayed feedback connections allow the neural networks to exhibit a non-linear dynamical behavior with memory properties.

2.1.4 Neural Network Learning

Learning is what defines a neural network, it's the "reason" of its existence. Neural networks are capable of learning from the environment and to improve their performance through learning. The neural network learns from the input data and, over time, and according with some measures, it adjusts its weights and bias in order to achieve a better performance. The learning process starts with the acquisition by the neural network of the data reflecting the environment. Afterwards and according to some rules the neural network adjusts its weights in accordance to the inputs and a performance criteria. By continuously checking the inputs and the learning state, the neural network is capable of reaching the best possible performance for the family of functions it is able to implement. The set of rules on which the neural network bases its learning is called the *learning algorithm*. There is a variety of learning algorithms, each one offering its own advantages. According to [80], there are five basic learning rules: error-correction learning, memory based learning, Hebbian learning, competitive learning and Boltzmann learning (error-correction learning is the one that we are going to discuss with more detail). Learning can also be done in a supervised (learning with a teacher) or unsupervised (learning without a teacher) way. Supervised learning assumes that the desired output data (target data) is supplied together with the input data.

2.1.4.1 Error-correction learning

Let us suppose we have a neural network like the one depicted in Fig. 2.7, having one or more layers, and a single output neuron and that the desired response or target for that output neuron is $d_k(n)$. Let us denote $y_k(n)$ the output signal of neuron k at iteration n. Suppose that

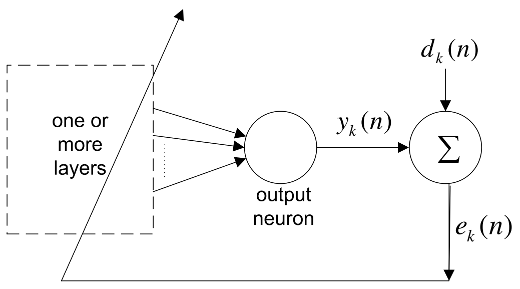


Figure 2.7: The error-correction learning.

The error between the desired target $d_k(n)$ and the output signal $y_k(n)$ is:

$$e_k(n) = d_k(n) - y_k(n). \tag{2.9}$$

The learning process acts by adjusting the weights of the neural network in such a way that the output of the neuron becomes more and more close to the desired target. This adjustment can be made, for example, by doing the minimization of an error function $E(n)$ defined in terms of the error signal $e_k(n)$ as:

$$E(n) = \frac{1}{2}\, e_k^2(n). \tag{2.10}$$

Learning is performed by adjusting the weights of the neural network until we reach a steady state where the minimum $E(n)$ is obtained. The minimization of this error function can be achieved by applying the well known delta rule or Widrow-Hoff rule [194]. Let us consider a single-layer neural network with linear activation function where $w_{kj}(n)$ is the value of weight w_{kj} of the connection between neuron k and the element $x_j(n)$ of the input vector $\mathbf{x}(n)$ at step n. The delta rule states that the adjustment $\Delta w_{kj}(n)$ applied to weight $w_{kj}(n)$ at time step n is defined by

$$\Delta w_{kj}(n) = \eta\, e_k(n)\, x_j(n), \tag{2.11}$$

where η is a positive constant determining the rate of learning when processing from one learning step to another. Parameter η is known as the *learning rate parameter*.

The update of the weight w_{kj}, after computing $\Delta w_{kj}(n)$, is obtained by

$$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n). \tag{2.12}$$

The learning rate is one of the most important parameters in the stability of a closed-loop feedback system like the neural network depicted in Fig. 2.7. The learning rate must be carefully chosen so that the convergence of the iterative learning process is achieved.

In the error-correction learning the weight adjustment can be made by propagating backwards, layer by layer, the error signals originated at the output of the neural network. This is done using the *Back-propagation algorithm*, probably the most popular method for NN training, that we will discuss later. Other popular methods used to perform NN learning (weight adjustment) are the conjugate-gradient, the Levenberg-Marquardt and genetic algorithms.

2.1.4.2 Cost Functions

We saw in the previous section that the learning process can be made by adjusting the weights of the neural network in order to minimize an error function E defined in terms of the error signal $e = d - y$. Error functions are also known as *cost functions* [1] or *objective functions.* There are several cost functions used in NN, both for regression and classification problems. In the following list we present the most used cost functions [21] (we are still considering a NN with a single output neuron.):

Sum-of-squares error $$\frac{1}{2}\sum_{i=1}^{N} e_i^2 \qquad (2.13a)$$

Minkowski error $$\sum_{i=1}^{N} |e_i|^R \qquad (2.13b)$$

Mean Squared error (MSE) $$\frac{1}{N}\sum_{i=1}^{N} e_i^2 \qquad (2.13c)$$

Cross-entropy error $$-\sum_{i=1}^{N} [t_i \log y_i + (1 - t_i)\log(y_i)] \qquad (2.13d)$$

[1]The term cost function is used in optimization and is related to the problem of finding an optimal solution for a particular problem. This optimal solution is obtained by minimizing or maximizing a real function (cost or objective function) by systematically choosing the values of their real or integer variables.

where $e_i = d_i - y_i$ is the error for each element i. The sum-of-squares error is a particular case of the Minkowski error (R=2).

In this work we will show how to use an entropic cost function in classification problems. This entropic cost function is build by computing the entropy of the errors e_i as described in the following chapter.

The MSE is one of the most popular cost functions used in classification problems with MLP's. The MSE, along with other indexes based on the squared error, is also used to evaluate the performance of a neural network. These indexes are used to compare the performance of different neural network solutions. The following list presents the most used performance indexes (besides the MSE) [127]:

Error mean
$$m_e = \frac{1}{N}\sum_{i=1}^{N} e_i \tag{2.14a}$$

Absolute error mean
$$m_{|e|} = \frac{1}{N}\sum_{i=1}^{N} |e_i| \tag{2.14b}$$

Relative error
$$E_{rel} = \frac{1}{N}\sum_{i=1}^{N} \frac{e_i}{x_i} \tag{2.14c}$$

Root mean square (RMS) error
$$E_{RMS} = \sqrt{MSE} \tag{2.14d}$$

The error standard deviation

$$\sigma_e = \frac{1}{N-1}\sqrt{\sum_{i=1}^{N}(e_i - m_e)^2} \tag{2.15}$$

is also used as a measure to evaluate the performance in regression problems.

2.1.4.3 The Back-propagation Algorithm

In neural networks with differentiable activation functions the error back-propagation method [161] is the most used for updating the neural network weights. It is a computationally efficient method [21] that uses the derivatives of the error function with respect to the network weights and biases. This method plays a central role in the majority of the training algorithms for multi-layer networks. Since we use this method in our experiments we now present a brief explanation.

We depict in Fig. 2.8, a simplified signal flow in a single-layer neural network with sigmoidal activation function with focus on neuron k.

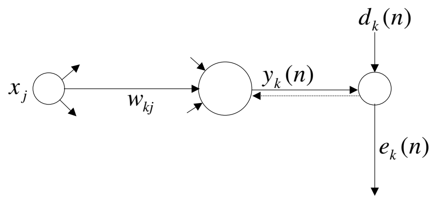


Figure 2.8: The signal flow and the back propagated error (dot line) in a single-layer perceptron.

Let $e_k(n) = d_k(n) - y_k(n)$ be the error between the output of neuron k and the desired target at step n. As we saw earlier, we need a cost function to perform the learning task. Let us assume the squared error cost function defined in 2.10. To update the neural network weights we use the delta rule, a gradient descent learning rule, to move through the "weight space" of the neuron in steps proportional to the gradient of the cost function with respect to each weight. The delta rule for the w_{kj} weight will be:

$$\Delta w_{kj}(n) = -\eta \, \frac{\partial E}{\partial w_{kj}} \; . \tag{2.16}$$

Noting that E is a function of $y_k(n)$ which, on the other hand, is a function of u_k and therefore they depend on the weights (Equation 2.3), in order to compute 2.16, we apply the chain rule of derivation and obtain:

$$\begin{aligned} \Delta w_{kj}(n) &= -\eta \, \frac{\partial E(n)}{\partial w_{kj}} \\ &= \eta \, \delta_k(n) \, x_j(n), \end{aligned} \tag{2.17}$$

where the local gradient $\delta_k(n)$ is defined by

$$\delta_k(n) = e_k(n) \, \varphi' \left(u_k(n) \right). \tag{2.18}$$

Let us now consider a two-layer perceptron with a simplified representation in Fig. 2.9.

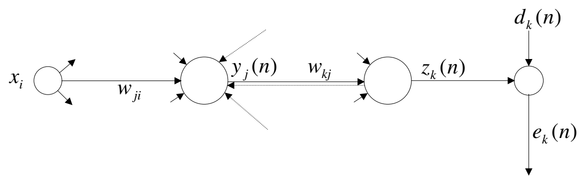


Figure 2.9: The signal flow and the back propagated errors (doted lines) in a two-layer perceptron.

To compute the update for weight w_{kj}, one must apply the same rule as for the previous single-layer perceptron. For the update of weight w_{ji}, one needs not only the error back propagated from neuron k, but also from all the neurons of the output layer. The error signal for a hidden neuron is determined recursively in terms of the error signals of all the output neurons to which the hidden neuron is directly connected.

Let us start by defining the local gradient $\delta_j(n)$ for the hidden neuron j as:

$$\delta_j(n) = -\frac{\partial E(n)}{\partial y_j(n)} \varphi_j' \left(u_j(n)\right). \tag{2.19}$$

To compute the partial derivative $\frac{\partial E(n)}{\partial y_j(n)}$ using all the information from the posterior neurons we must do

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial y_j(n)} \ , \tag{2.20}$$

which, by applying the chain rule, is:

$$\begin{aligned} \frac{\partial E(n)}{\partial y_j(n)} &= -\sum_k e_k(n)\varphi_k'(u_k(n))\, w_{kj}(n) \\ &= -\sum_k \delta_k(n)\, w_{kj}(n). \end{aligned} \tag{2.21}$$

Using 2.21 equation 2.19 can now be written as:

$$\delta_j(n) = \varphi_j'(v_j(n)) \sum_k e_k(n)\varphi_k'(u_k(n))\, w_{kj}(n) \tag{2.22}$$

and the update of weight $w_{ji}(n)$ can be finally obtained as follows:

$$\Delta w_{ji}(n) = \eta\, \delta_j(n)\, x_i(n). \tag{2.23}$$

2.1.4.4 The Learning rate

We have already mentioned that the learning rate is one of the most important factors when training a neural network. We also mentioned, in the previous section, that the back propagation algorithm, by using a gradient descent approach, performs, in each iteration, a local descent in the error surface. If the learning rate is small, we will obtain small changes in the weights and, consequently, an extremely smooth trajectory in the error surface. This can lead to a very slow learning and, if the curvature of the error surface is very smooth and changing with direction, the local gradient may not point towards the minimum, thus producing an even slower learning. On the other hand, if we use a large learning rate, we will have large changes in the weights causing big "jumps" in the trajectory in the error surface producing a highly erratic learning. A suitable learning rate parameter must be chosen for each experiment in order to avoid the mentioned problems. Strategies to optimize this learning procedure were made by several authors, mainly with the assumption that the learning rate must change along the learning process; in other words, one should use an adaptive learning rate. The learning rate should be high in the beginning of the training process and in the end one should use small learning rates in order to avoid getting out of the region near the global minimum. The adaptive learning rate is usually based on the following approaches:

1. Start with a small learning rate and increase it exponentially if successive iterations reduce the error, or rapidly decrease it if a significant error increase occurs [13, 192].

2. Start with a small learning rate and increase it if successive iterations keep gradient direction fairly constant, or rapidly decrease it if the direction of

the gradient varies greatly at each iteration [27].

3. An individual learning rate is given to each weight, which increases if the successive changes in the weights are in the same direction and decreases otherwise [92,156,177].

4. Use a closed formula to calculate a common learning rate for all the weights at each iteration or a different learning rate for each weight [51,84,125,126].

In Chapter 3 we shall present adaptive learning rates based on the approaches 2 and 3, in the scope of neural networks trained with entropic cost functions.

2.1.4.5 Training in Sequential and Batch Mode

The training of a neural network with the back-propagation algorithm is performed by presenting a set of training examples from the data set of a particular classification problem to the network. During the training phase, one complete presentation of the training set to the network is called an *epoch*. An usual practice to train a neural network is to maintain an epoch-by-epoch learning until there is a stabilization of the neural network weights and the convergence of the error function over the entire training set to a minimum value. However, and to avoid over-fitting [2], this procedure must be done with some caution. To avoid the loss of generalization capabilities we must not overtrain the network. Some techniques to avoid this behavior are mainly based on early stopping [3]; this means that one should stop training before the neural network has over fitted the training data.

[2] Over-fitting is related with the bias/variance dilemma. When training a neural network one can get a model with too little flexibility having high bias or a model with too much flexibility having a high variance. A tradeoff between these two proprieties as to be accomplish to obtain a model with good generalization capabilities. An extensive discussion about the bias/variance dilemma can be found in [60].

[3] We include a brief discussion about early stopping in the section "Practical Aspects of Neural Network Implementation"

When applying the back-propagation learning we can have two different learning modes:

1. Sequential Mode. In this learning mode, also referred as *Online* or *Stochastic* the weights update is performed each time a new element from the training set is presented to the neural network. For each new element $\mathbf{x}_1$, we compute the error between the output $\mathbf{y_1}$ of the neural network and the desired target $\mathbf{d}_1$, and we apply the back-propagation algorithm to update the weights of the neural network. By doing this, we will have, for each epoch, as much weights updates as the number of elements of the training set. The back-propagation algorithm presented earlier was denoted as a sequential mode algorithm.

2. Batch Mode. In this learning mode, the back-propagation weights update is performed after presenting all elements of the training set to the neural network and computing the respective error function. If using the mean squared error E (2.14d) the delta rule, similar to 2.17, will be defined, for the weight of the connection between node/neuron j and neuron k as:

$$\begin{aligned}\Delta w_{kj}(n) &= -\eta\,\frac{\partial E}{\partial w_{kj}}\\ &= \frac{\eta}{N}\sum_{n=1}^{N}\delta_k(n)\,x_j(n).\end{aligned}\tag{2.24}$$

   In batch mode learning we will have, for each epoch, a single update of the weights.

Each of the presented learning modes have advantages and disadvantages; with batch mode, by estimating accurately the gradient vector, the convergence to, at least, a local minima is guarantied. Also the algorithm is more easily parallelized. On the other hand, for the sequential mode, it is more difficult to establish theoretical conditions for convergence due to its stochastic nature. The sequential mode, however, requires less computational memory, being preferred for large data sets, and it is also preferred for data sets with some redundance.

2.1.5 Practical Aspects of Neural Network Implementation

We will present in this subsection some of the practical aspects related to the real implementation of neural networks. We will discuss the normalization of the data set, the splitting of the data set in training and test sets, the architecture of the neural network, the stopping criteria and several other practical conditions.

Let us assume that the original data set was already pre-processed and that problems with missing values and noise or outliers were already treated. In other words, we assume that our data set is a "clean" data set. The work on data preparation for neural network data analysis in [203] presents a study on several aspects regarding data pre-processing.

The first thing that usually one must do to this clean data set is to perform a data normalization, or standardization, to avoid that higher inputs assume a more important role in the learning process than small inputs. The usual normalization processes transform the data so that:

- every feature of the data is scaled in the interval $[0, 1]$ or $[-1, 1]$, or
- every feature is standardized to have zero mean and unitary standard deviation.

The next step is to choose the architecture of the neural network. There is no rule specifying the number of hidden layers and the number of neurons in each hidden layer. We mentioned earlier that a neural network with one hidden layer can approximate arbitrarily close any decision boundary, so, for most problems, a two-layer neural network will be sufficient. A three-layer network can be considered if a data set is particularly hard to train. Although there are some works suggesting formulas for determining the number of neurons in the hidden layer (examples are [26, 59, 157, 170, 205]), we still think that there is nothing like experimentation. Experiments should be performed with a range of values for the number of hidden neurons that must be chosen taking into account the complexity of the problem. There are also some techniques consisting on starting with a high complex neural network and then, during training, performing a pruning by eliminating those weights with small influence (very low value) [61, 105, 173, 182].

As for the initial values for the weights and bias, we should use random small values [4]. This is done to prevent the possibility that some of the initial output values could be in the saturation region of the activation function. As activation functions we use, in our experiments, the hyperbolic tangent in all neurons.

After choosing the architecture of the neural network and the initial weights, and before starting the training phase we will split the data set in two different subsets: the training set and the test set (some authors consider the partition of the data set in training set, validation set and test set, however, we use the test set for validation and test). Training will be performed over the training set. The test set will be used, in our experiments, for validation and testing. The 10-fold cross validation and the leave-one-out are the most used methods for splitting the data set. In the first one, the data set is randomly split in 10 subsets being, in each experiment, nine of them used for training and one used for testing. The experiment is repeated 10 times, each time one of the 10 subsets being used as test set. In the second method, the leave-one-out, all the data except one element is used for training and the remain element is used for testing (the training is repeated N times). In all our experiments we use a different splitting method: a 2-fold cross validation (this was made for comparison purpose with the results obtained by the other elements of our research group). In this method, each time, half of the data set is randomly chosen for training and the other half for testing. Then the data sets are used with inverted roles (the original training set became the test set and the original test set became the training set). We must point out that, different splitting will, probably, originate different final results. Classification errors using the 2-fold cross validation will, probably, be higher than using the 10-fold cross validation.

[4]In our experiments we use random values from a normal distribution with zero mean and unitary standard deviation, multiplied by 0.1.

There are several ways of stopping the training of a neural network. We may stop training when the error reaches a predefined small value, however, we will probably have a neural network model too much fitted to the training data and that does not possess the generalization capability. To stop the training phase before this happens, we validate the model with the test set. In Fig. 2.10 we represent the learning curves for training and testing/validation. Usually the neural network model does not do as well in the test set as it does in the training set used to build it. The training error curve decreases monotonically, as usual, as we increase the number of epochs (theoretically we can get zero training error if the network has enough complexity). The model is validated periodically in the test set. The test error curve will monotonically decrease until a certain point and then it will start increasing as training continues. Training should be stopped in a epoch around the minimum test error. Early stopping is related to

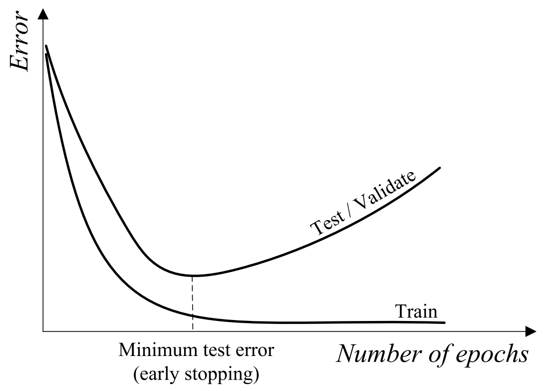


Figure 2.10: Train error versus test error and the early stopping rule.

the bias/variance trade-off referred earlier.

After training the neural network one has to test the obtained model on the test data (if using test data for validation, the test is performed simultaneously with the training). In order to avoid the possibility of having reached a local minima in the training phase we will make several runs with different initial weights. In each experiment we usually perform 20 runs for each combination of the involved parameters. This number of runs, N_{runs}, is proposed in [89] if one wants to be 99% confident that the best-of-all runs (random starts) will result in one of the best (lowest) 20% possible error values. This value is given by the formula:

$$N_{runs} = \frac{\ln(1 - F_w(a))}{\ln(1 - F_X(a))} , \tag{2.25}$$

where $F_w(a)$ is the level of confidence, and $F_X(a)$, the percentage vicinity of the lower tail of the distribution, which the best-of-N_{runs} is expected to provide.

The final result of each experiment will be the average of the errors of the N_{runs} and also the respective standard deviation. For each run we compute the classification error dividing the number of bad classified elements by the total number of elements of the test set. The standard deviation is important when comparing results obtained with different models.

One way to evaluate the errors in classification problems is to build a confusion matrix, a visualization tool used to see how the neural network is confused in the classification of a certain class as another. In a confusion matrix each column represents the instances in a predicted class, while each row represents the instances in an true class. An example of a confusion matrix is shown in Table. 4.6

There are other tools used to measure the performance of a neural network. A special tool for a two-class problem is the ROC curve (The Receiver Operator Characteristic curve). It combines the concepts of sensitivity [5] and specificity [6] that depend on the arbitrary selection of a decision threshold. The ROC curve is shown to be a simple yet complete empirical description of this decision thresh-

[5]Sensitivity is a statistical measure of how well a binary classification test correctly identifies a condition. In a medical test to determine if a person has a certain disease, the sensitivity to the disease is the probability that, if the person has the disease, the test will be positive. That is, the sensitivity is the proportion of true positives of all positive cases in the population.

[6]Specificity is a statistical measure of how well a binary classification test correctly classifies cases not belonging to that class. In a medical test to determine if a person has a certain disease, the specificity to the disease is the probability that, if the person does not have the disease, the test will be negative. That is, the specificity is the proportion of true negatives of all negative cases in the population.

old effect, indicating all possible combinations of the relative frequencies of the various kinds of correct and incorrect decisions [134].

2.2 Entropy

The concept of entropy originated from thermodynamics is widely known from its second law, first stated by Rudolf Clausius. He introduced the concept of entropy in 1865 during the apogee of steam engines: it specified the maximum energy available for useful work. In a posterior stage, in statistical mechanics, Boltzmann stated his famous equation $S = k \ln W$ describing the entropy (S) as the relation between the number of microstates in a system (W) and its macroscopic properties (k is the Boltzmann constant). Tsallis, in 1988, presented a extension of the concept of entropy, stating that Boltzmann formula was not valid for some rare events [36, 184]:

$$S_q(p) = \frac{1}{q-1}\left(1 - \sum_x p^q(x)\right). \tag{2.26}$$

In this case, p is a probability distribution, and q is a real parameter. In the limit as $q \to 1$, the normal Boltzmann-Gibbs entropy is recovered.

Claude Shannon used the same term, entropy, when he introduced his work about communications over a noisy channel [174]. He studied the statistical structure of a message to be transmitted and the nature of the final destination of the information. The first concepts of information in communications were introduced by Nyquist and Hartley in the 1920's. In 1924 Nyquist showed that the speed W of transmission of information over a telegraph circuit, with a fixed line speed, is proportional to the logarithm of the number m of values used to encode the message: $W = k \log m$ where k is a constant [143]. Later, in 1928, Hartley generalized the concept and introduced the measure of information for communication [74], the amount of information associated with an event x which occurs with probability p, as $I = \log \frac{1}{p}$.

Entropy has been used in a variety of applications, scattered through the large science spectrum. Examples are the applications in information and coding theory, dynamical systems, logic and theory of algorithms, statistical inference and prediction, physical sciences, economics, biology, humanities and social sciences.

2.2.1 Entropy and Information

In a communication system, the information of a certain event is higher as smaller is its probability of occurrence. In simple cases, the amount of information is conveniently measured by the logarithm of the number of available choices [7].

Note that information here is not equivalent and must not be confused with meaning [185]. The concept of information is too complex and cannot be explained with a single definition. Two messages, one of which is heavily loaded with meaning and the other which is pure nonsense, can be exactly equivalent as regards to our use of information. Information, in communication theory, relates not so much to what you *do say*, as to what you *could say*. Information is one's freedom of choice when one selects a message [185]. In this sense it is clearly related to the initial uncertainty associated to the message selection.

When an event is related to a previous one – when a message is produced by consecutive symbols – the probability of occurrence of the various symbols at a certain stage of the process can be dependent on the previous one. The quantity that measures the information of such a process must be expressed in terms of the various probabilities involved: those of getting to certain stages of the message forming process, and the probabilities that, when in those stages, certain symbols will be chosen next. This quantity, moreover, involves the logarithm of probabilities, so that it is a natural generalization of the logarithmic measure for simple cases.

Claude Shannon defined precisely *entropy* as how much choice is involved in the selection of a certain event or how uncertain we are of the outcome. Shannon, in his famous 1948 paper [174], introduced the concept of entropy and mutual

[7]One of the reasons for using the logarithm is because information should be additive: $I(ab) = I(a) + I(b)$.

information, and laid the foundations to the new field of information theory. Since then, some concepts related to Shannon's entropy were presented, like relative entropy that was first defined, in 1951, by Kullback and Leibler [116], and also some different properties were found for these quantities, for example, Fano's inequality [49].

Given a discrete random variable X taking values in the finite set $\mathcal{X} = \{x_1, x_2, ..., x_n\}$ with probabilities $\mathbf{p} = (p_1, p_2, ..., p_n)$, we define the (Shannon) entropy of X to be the expectation of the Hartley's information measure:

$$H(X) = -K \sum_{i=1}^{n} p_i \log_2 p_i \,, \tag{2.27}$$

where K is a positive constant. The entropic measure

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i \,, \tag{2.28}$$

(the constant K is only related to the choice of the unit of measure, the used logarithm) played an important role in information theory as a measure of information, choice and uncertainty.

Note that the entropy of X depends, not on the values of X, but on their probabilities; however we will use, as usual, $H(X)$ as $H(\mathbf{p})$.

Originally Shannon used log_2 and measured entropy in bits. From now on we will use log, representing the natural logarithm.

If X is a random variable with Bernoulli distribution that takes the value 0 with probability $1 - p$ and the value 1 with probability p, the entropy is:

$$H(X) = -p \;\log p - (1-p) \;\log(1-p)). \tag{2.29}$$

As can be seen, for $p = 0$ or 1, there is no uncertainty in the event (X is deterministic) and so is the entropy $H(X) = 0$. If $p = 1/2$, X will have the highest uncertainty and consequently the highest value for the entropy is $H(X) = 1$ (see Fig.2.11).

Entropy possesses the following properties:

1. $H(X) \geq 0$. The equality holds if one of the probabilities is 1 and all the others are 0.
2. $H(X)$ is a continuous function of $\mathbf{p}$.
3. $H(X)$ is symmetric. In other words, the ordering of the probabilities $p_1, p_2, ..., p_n$ does not influence the value of $H(X)$.
4. The entropy of independent variables is additive. If X and Y are two independent variables with probabilities $\mathbf{p}$ and $\mathbf{q}$ then, for the entropy of the join event (X, Y), we have $H(X, Y) = H(X) + H(Y)$.
5. $H(X) \leq \log n$, with equality iff $p_1 = p_2 = ... = p_n = \frac{1}{n}$.

After Shannon other authors have presented other information entropy measures having almost the same properties of the original one. Rényi presented a generalized form of information measure based on a general theory of means, derived from the following axioms:

1. the information of a couple of independent individual events is the sum of their respective information;

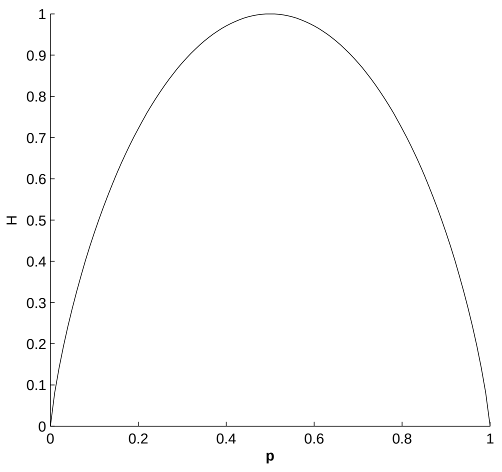


Figure 2.11: The entropy of a Bernoulli variable for different $\mathbf{p}$.

2. the information of a random variable is a (generalized) mean information of the individual event information;

3. the information is additive for independent random variables.

The generalized mean of the real numbers $x_1, x_2, ..., x_n$ with weights $p_1, p_2, ..., p_n$ has the form

$$\varphi^{-1}\left(\sum_{k=1}^{n} p_k\, \varphi(x_k)\right), \tag{2.30}$$

where $\varphi()$ is the Kolmogorov-Nagumo function [113, 139], which is an arbitrary continuous and strictly monotonic function defined on the real numbers.

The entropy measure should be [154, 155]

$$\varphi^{-1}\left(\sum_{k=1}^{n} p_k\, \varphi(I(p_k))\right). \tag{2.31}$$

To meet the additivity condition, $\varphi()$ can either be $\varphi(x) = x$ or $\varphi(x) = 2^{(1-\alpha)x}$. If the first expression is used, 2.31 will become Shannon's entropy 2.28. If the latter expression is used, Rényi's entropy [155] is obtained instead:

$$H_{R\alpha} = \frac{1}{1-\alpha} \log\left(\sum_{k=1}^{n} {p_k}^{\alpha}\right), \qquad \alpha > 0, \alpha \neq 1\,. \tag{2.32}$$

This entropy measure is often known as a *generalized information measure*, *information measure of order* α, or simply α*-order Rényi entropy.*

Rényi's entropy is a family of entropy measures and has the Shannon's entropy (H_S) as a special case. The relation between both entropies is defined by:

$$\begin{cases} H_{R\alpha} \geq H_S \geq H_{R\beta} & \text{if } 0 < \alpha < 1 \text{ and } \beta > 1, \\ \lim\limits_{\alpha \to 1} H_{R\alpha} = H_S & \end{cases}. \tag{2.33}$$

Rényi's entropy is the only entropy measure that satisfies the above three axioms (including, of course, Shannon's entropy as a particular case) [98]. However, there are other measures of information that do not satisfy all the axioms but are still useful for some applications. The Havrda and Charvat's entropy [79]:

$$H_{H\alpha} = \frac{1}{1-\alpha} \log\left(\sum_{k=1}^{n} {p_k}^{\alpha} - 1\right), \qquad \alpha > 0, \alpha \neq 1, \tag{2.34}$$

is an example of an entropy measure, similar to Rényi's entropy but with different scaling, that does not satisfy the additivity axiom but is still equivalent to Rényi and Shannon entropies with regard to entropy maximization [104]. Another example of an entropy measure is also $H_\infty = -\log\big(\max_k(p_k)\big)$ [104].

2.2.1.1 Conditional, Joint and Mutual Information Measures

In the previous subsection we have defined the entropy of a single discrete random variable. The following measures extend the definition to a pair of discrete random variables.

Consider two discrete random variables X and Y taking values in the finite sets $\mathcal{X} = \{x_1, x_2, ..., x_n\}$ and $\mathcal{Y} = \{y_1, y_2, ..., y_n\}$ with probabilities $\mathbf{p} = (p_1, p_2, ..., p_n)$ and $\mathbf{q} = (q_1, q_2, ..., q_m)$ respectively. We can see (X, Y) as a compound probabilistic experiment with outcome (x_i, y_i). Let us consider $r(x_i, y_i)$ as the joint probability, the probability that the compound experiment (X, Y) will yield (x_i, y_i) as outcome.

As a straightforward generalization of 2.28 the joint Shannon's entropy of (X, Y) is defined as ([35]):

$$H(X,Y) = -\sum_{i=1}^{n}\sum_{j=1}^{m} r(x_i, y_j)\log\left[r(x_i, y_j)\right]. \tag{2.35}$$

A straightforward extension is also the definition of the conditional entropy of a certain r.v. Y, related to the probability of Y under the condition that outcome x_i has occurred.

$$H(Y|x_i) = -\sum_{j=1}^{m} q(y_j|x_i)\log\left[q(y_j|x_i)\right]. \tag{2.36}$$

Note that we use the conditional probabilities $q(y_j|x_i)$, $j = 1, 2, ..., m$, instead of probabilities $q(y_j)$, $j = 1, 2, ..., m$.

The average conditional entropy of Y given X is obtained by averaging $H(Y|x_i)$ over all the x_i values as:

$$\begin{aligned}\sum_{i=1}^{n} p(x_i) H(Y|x_i) &= -\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i) q(y_j|x_i) \log\left[q(y_j|x_i)\right]; \\ H(Y|X) &= -\sum_{i=1}^{n}\sum_{j=1}^{m} r(x_i, y_j) \log\left[q(y_j|x_i)\right].\end{aligned}$$

The minimum and maximum values for the conditional entropy of Y given X, $H(Y|X)$ is given by the following inequalities:

1. $H(Y|X) \geq 0$;

2. $H(Y|X) \leq H(Y)$, with equality only if X and Y are independent.

The main conclusion about the second inequality is that, on average, the information about X leads to a reduction of the uncertainty of event Y. In case X and Y are independent, knowing X value does not help reducing the initial uncertainty of Y. The same properties hold for $H(X|Y)$. For all r.v. X and Y,

$$\begin{aligned}H(X,Y) &= H(X) + H(Y|X) \\ &= H(Y) + H(X|Y). \end{aligned} \tag{2.37}$$

Using 2.37 and the previous inequalities one easily derives:

$$H(X,Y) = H(X) + H(Y|X) \leq H(X) + H(Y), \tag{2.38}$$

with equality only if events X and Y are independent. One can say that, if an absolute dependence exists between the outcomes of Y and X (if Y is known after knowing X), the conditional entropy $H(Y|X) = 0$ and $H(X,Y) = H(X)$.

Relative entropy and mutual information are two further concepts related with entropy which are very important in information theory. Relative entropy, also known as the Kullback-Leibler divergence [116], is a measure of the dissimilarity between two distributions and is defined as the expectation of the logarithmic likelihood ratio. Relative entropy is thus defined as:

$$D(p|q) = \sum_{x} p(x) \log \frac{p(x)}{q(x)}, \tag{2.39}$$

where $p(x)$ and $q(x)$ are two probability mass functions. Although sometimes mentioned as "Kullback-Leibler distance" this is not a true distance measure between two distributions since it is not symmetric and does not satisfy the triangular inequality. The relative entropy is used as a measure of inefficiency of assuming that the distribution is q when the true distribution is p [35].

Mutual information can be seen as a dependency measure between two random variables Y and X, or the amount of information that one random variable contains about another random variable. Mutual information is thus defined as:

$$\begin{aligned} I(X,Y) &= H(Y) - H(Y|X) \\ &= -\sum_{i=1}^{n}\sum_{j=1}^{m} r(x_i,y_j)\log\frac{r(x_i,y_j)}{p(x_i)q(y_j)}\,. \end{aligned} \tag{2.40}$$

When X and Y are independent $I(X,Y)=0$. $I(X,Y)$ is symmetric, that is

$$I(X,Y) = I(Y,X) = H(X) - H(X|Y). \tag{2.41}$$

The relationship between the presented information measures (different entropies and mutual information) is shown in the Venn diagram of Fig.2.12. The intersection of the two circles will not occur in the case of independent r.v. X and Y.

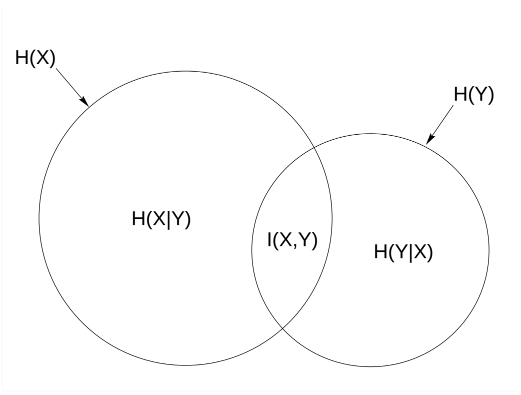


Figure 2.12: Relationship between information measures.

The following relationships between information measures can be easily derived from the Venn diagram:

1. $H(X|Y) \leq H(X)$ and $H(Y|X) \leq H(Y)$
2. $I(X,Y) \leq H(Y)$ and $I(X,Y) \leq H(X)$
3. $I(X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$
4. $H(X,Y) = H(X|Y) + I(X,Y) + H(Y|X) = H(Y) + H(X|Y) = H(X) + H(Y|X)$
5. $H(X,Y) \leq H(X) + H(Y)$

Until now we have just presented information measures and related properties for discrete random variables. The equivalent formulas for continuous random variables are obtained in a straightforward way, substituting summations by integrals and probability mass functions by probability density functions. The entropies of continuous r.v. are sometimes known as differential entropies. This way, Shannon's (differential) entropy is:

$$H(X) = -\int_{-\infty}^{+\infty} f(x) \log f(x)\, dx, \tag{2.42}$$

where $f(x)$ is the probability density function (pdf) of r.v. X. Analogously, the α-order Rényi differential entropy is:

$$H_{R\alpha}(X) = \frac{1}{1-\alpha} \log \left(\int_{-\infty}^{+\infty} f^{\alpha}(x)\, dx \right), \qquad \alpha > 0, \alpha \neq 1. \tag{2.43}$$

The second-order Rényi's differential entropy,

$$H_{R2}(X) = -\log \left(\int_{-\infty}^{+\infty} f^{2}(x)\, dx \right), \tag{2.44}$$

will be used later and will be referred to as the Rényi's quadratic entropy due to the quadratic form of the pdf.

Differential entropy follows some important extremal properties:

1. If the density f is concentrated on a limited interval $[a;b]$, then the differential entropy is maximal iff f is uniform on $[a;b]$, and then $H(f) = 0$.

2. If the density is concentrated on the positive half line and has a fixed expectation, then the differential entropy takes its maximum for the exponential distribution.

3. If the density has fixed variance, then the differential entropy is maximum for the Gaussian density.

All the properties presented earlier, such as joint and conditional entropies and mutual information, are also valid for differential entropies [35].

2.2.2 Entropy Estimation

How can entropy be estimated from data? One might think that this problem has already been conclusively studied and understood. However, taking into account all the works presented since Shannon's 1948 paper, it still is a currently researched subject especially in what concerns differential entropy.

Several authors have presented explicit expressions for the entropy of known continuous probability distributions. An overview of some formulas for univariate densities can be found in [117] and for multivariate densities in [3, 64]. The entropy estimation problem is rather more difficult for unknown distributions because one must estimate the probability density function and, when directly applying formula 2.42, to use numerical integration. Before discussing entropy estimation, we will present in the following two subsections the most used nonparametric methods to estimate the probability density function of a continuous random variable: the histogram-based estimator and the Parzen window estimator. See [129] for a review on nonparametric density estimation.

2.2.2.1 Histogram-based Density Estimators

According to [181], the first histogram appeared in 1661, due to John Graunt, a London haberdasher, as an attempt to summarize the information collected by the parish priests of the Church of England about the last 100 years births and deaths, as was ordered by king Henry VIII, concerned with the decrease of the

English population caused by the plague. This first histogram came about from the need to summarize the mountains of collected data.

Let us consider a sample $\{x_1, x_2, ..., x_n\}$ of observations of an i.i.d. random variable $X \in \mathbb{R}$, from an unknown absolutely continuous probability density function $g(x)$.

The histogram-based density estimator is used to estimate the truncated [8] density of $g(x)$, $f(x)$ (if $g(x)$ has infinite support) as

$$f(x) = \begin{cases} \frac{g(x)}{\int_a^b g(t)dt} & x \in [a,b] \\ 0 & otherwise \end{cases} . \tag{2.45}$$

Let us consider the partition of the interval $[a,b]$ by $a = t_0 < t_1 < ... < t_i < ... < t_m = b$ and denote

$T_i = [t_i, t_{i+1}[$

$q_i = \sum_{k=1}^{n} I_{x_k \in T_i}$

$l(T_i) = t_{i+1} - t_i$

The histogram is built by assigning to each bin a height proportional to the probability:

$$p(t) = \begin{cases} q_i/n & t \in T_i; \\ q_{m-1}/n & t = b; \\ 0 & t \notin [a,b] \end{cases} . \tag{2.46}$$

The estimated density given by the histogram is therefore obtained by:

$$\hat{f}_H(t) = \begin{cases} p(t)/l(T_i) & t \in T_i; \\ p(t)/l(T_{m-1}) & t = b; \\ 0 & t \notin [a,b] \end{cases} . \tag{2.47}$$

If f is bounded and has continuous derivatives up to order three, except at the endpoints of $[a,b]$, and we consider equal spacing, $t_{i+1} - t_i = 2h(n) \equiv 2h_n$,

[8] Given the finiteness of the available data sample only a truncated pdf estimation can be reliably performed.

then, if $n \to \infty$ and $h_n \to 0$ such that $nh_n \to \infty$, for $x \in [a, b]$, the following holds [9] (see proof in [181]):

$$MSE\left(\hat{f}_H(x)\right) = E\left[\left(\hat{f}_H(x) - f_H(x)\right)\right] \to 0. \qquad (2.48)$$

In other words, $\hat{f}_H(x)$ converges in the L^2-norm to $f(x)$ and, therefore, it is a consistent estimator for $f(x)$.

2.2.2.2 Parzen Window Density Estimator

The Parzen window estimator is a generalization of the shifted-histogram or Rosenblatt's kernel estimator. Rosenblatt's approach is simply a histogram which, for estimating the density at x, has been shifted so that x lies at the center of a mesh interval. The Rosenblatt estimator is therefore given by ([160])

$$\hat{f}_n(x) = \frac{\sharp \text{ sample points in } (x - h_n, x + h_n)}{2nh_n}, \qquad (2.49)$$

where h_n is a real valued number constant for each n, i.e.,

$$\hat{f}_n(x) = \frac{F_n(x + h_n) - F_n(x - h_n)}{2h_n}, \qquad (2.50)$$

with

$$F_n(x) = \frac{\sharp \text{ sample points} \le x}{n}, \qquad (2.51)$$

the empirical distribution of the data.

The Rosenblatt estimator, as the histogram-based estimator, $\hat{f}_H$, is also a consistent estimate of $f(x)$ [181].

One can also represent Rosenblatt's shifted histogram estimator as:

$$\hat{f}_n(x) = \frac{1}{n}\sum_{j=1}^{n} \frac{1}{h_n} w\left(\frac{x - x_j}{h_n}\right), \qquad (2.52)$$

where $w(u) = \begin{cases} \frac{1}{2} & \text{if} \quad |u| < 1 \\ 0 & otherwise \end{cases}$ is a rectangular kernel function.

[9] Condition $nh_n \to \infty$ is used to guarantee that n converges more rapidly to ∞ than h_n to 0. These two parameters must be related in such a way that, when n grows, it must grow faster than the decreasing of h_n.

A global optimal value for h_n is presented in [181] and can be obtained from the integrated mean square error:

$$h_n = \left[\frac{9}{2\int (f''(x))^2 dx}\right]^{1/5} n^{-4/5}. \tag{2.53}$$

Note that there is a faster decrease in h_n with the growth of n in such a way that condition $nh_n \to \infty$ is satisfied.

Although Rosenblatt suggested generalizing 2.52 to estimators using different bases (kernels) than step (rectangular) functions, the detailed explanation and study of kernel estimators is due to Parzen [146]. Parzen considered the estimator for $f(x)$ as

$$\hat{f}_n(x) = \int_{-\infty}^{\infty} \frac{1}{h_n} K\left(\frac{x-y}{h_n}\right) dF_n(y) \simeq \frac{1}{nh_n}\sum_{j=1}^{n} K\left(\frac{x-x_j}{h_n}\right), \tag{2.54}$$

where

$$\begin{cases} \int_{-\infty}^{\infty} |K(y)\, dy| < \infty \\ \underset{-\infty<y<\infty}{Sup} |K(y)| < \infty, \\ \lim\limits_{y\to\infty} |y\, K(y)| = 0 \end{cases} \tag{2.55}$$

and

$$\begin{cases} K(y) \geq 0 \\ \int_{-\infty}^{\infty} K(y)\, dy = 1 \end{cases} . \tag{2.56}$$

If K is a Borel function [10], the kernel estimator $\hat{f}_n$ in 2.54 subject to 2.55 and 2.56 is asymptotically unbiased if $h_n \to 0$ as $n \to \infty$, i.e,

$$\lim_{n} E\left(\hat{f}_n(x)\right) = f(x). \tag{2.57}$$

The estimator $\hat{f}_n$ in 2.54 subject to 2.55 and 2.56 is consistent if we add the additional constrain $\lim\limits_{n\to\infty} nh_n \to \infty$. Proofs of the previous theorems can be found in [181].

[10] Borel functions, also called measurable functions, are well-behaved functions between measurable spaces.

A global optimal value for h_n obtained from the minimization of the integrated mean square error is:

$$h_n = n^{-\frac{1}{2r+1}}\alpha(K) + \beta(f) \tag{2.58}$$

with

$$\alpha(K) = \left[\frac{\int K^2(y)\,dy}{2r\left(\int y^r K(y)dy/r!\right)^2}\right]^{1/(2r+1)} \qquad \text{and}$$

$$\beta(f) = \left[\int |f^{(r)}(y)|^2\,dy\right]^{-1/(2r+1)},$$

where r is the characteristic exponent [11] of the kernel. If K is a probability density r cannot be higher than 2, with $r = 2$ being the most important case. Examples of kernels with characteristic exponents of 2 are the Gaussian kernel, the double exponential or any other symmetric kernel K having $x^2K(x) \in L^1$. Since we assume the functional form of K to be given, we can evaluate $\alpha(K)$ more or less easily. The determination of $\beta(f)$ is fraught with difficulty because $f(y)$ is unknown. Examples of values of $\alpha(K)$ for $r = 2$ are shown in Table 2.1.

Table 2.1: α values for different kernels with $r = 2$.

K		$\alpha(K)$
$K(y) = 1/2$	$\lvert y\rvert \leq 1$	1.3510
$K(y) = \frac{15}{16}(1-y^2)^2$	$\lvert y\rvert \leq 1$	2.0362
$K(y) = \frac{1}{\sqrt{2\pi}}\exp^{-y^2/2}$	$\lvert y\rvert < \infty$	0.7764

In Parzen window estimation we are limited by the fact that $\beta(f)$ is generally unknown. One could consider to iteratively improve an estimation of $\beta(f)$. As an example we mention that, for a Gaussian density with standard deviation σ,

[11] In fact, r is the characteristic exponent of k, the Fourier transform of the kernel K that satisfies conditions 2.55 and 2.56. If there exists a positive r, such that $k_r = \lim\limits_{u\to 0}\left[\frac{1-k(u)}{|u|^r}\right]$ is nonzero and finite, r is called the characteristic exponent of k.

we have:

$$\int |f''(y)|^2 dy \approx 0.212\sigma^{-5} \Rightarrow \beta(f) \approx 1.3637 \Rightarrow h_n \approx 1.06\sigma n^{-1/5}. \qquad (2.59)$$

(For now on and for the sake of simplicity we will use $\hat{f}(x)$ and h for $\hat{f}_n(x)$ and h_n)

Aware of the fact that kernel estimators are not, in general, robust against poor choices of h [181], some authors have suggested some values for this parameter based on formula 2.58. Examples are the ones proposed by Silverman, both for unidimensional cases [179]: the mentioned value $h = 1.06\sigma n^{-1/5}$ (formula 2.59), and also

$$h = 0.9An^{-1/5} \qquad \text{where} \qquad A = min\left(\sigma, \frac{IQR}{1.34}\right), \qquad (2.60)$$

where IQR is the interquartile range. Also the one proposed by Bowman and Azzalini [24] for multidimensional cases and assuming normal distributions:

$$h = \sigma\left(\frac{4}{(m+2)n}\right)^{\frac{1}{m+4}}, \qquad (2.61)$$

where m is the vector dimension (for $m = 1$, $h = \sigma(4/3n)^{0.2}$, similar to formula 2.59).

However, the choice of h (also known as *smoothing parameter* or *bandwidth*) is always limited by the bias-variance tradeoff: the bias can be reduced at the expense of the variance, and vice versa. The bias of an estimate is the systematic error incurred in the estimation and the variance of an estimate is related to the random error incurred in the estimation. The bias-variance dilemma applied to the choice of h simply means that a large h will reduce the differences among the estimates of $\hat{f}(x)$ for different data sets (the variance) but it will increase the bias of $\hat{f}(x)$ with respect to the true density. A small h will reduce the bias of $\hat{f}(x)$, at the expense of a larger variance in the estimates $\hat{f}(x)$.

The Parzen window estimator for multiple dimensions is:

$$\hat{f}(x) = \frac{1}{n\,h^d}\sum_{j=1}^{n} K\left(\frac{x - x_j}{h}\right). \qquad (2.62)$$

We will return to the subject of the optimal value for h when discussing Rényi's entropy estimation in the next subsection.

2.2.2.3 Differential Entropy Estimation

After one of the first works in entropy estimation, in 1956, for discrete distributions [12], this problem, less complex than the estimation for continuous distributions, was also analysed by several other authors [10,14,65,87,114,144,145,172]; these works presented estimation methods, convergence properties and studied the estimators complexity and their statistical properties.

Regarding continuous distributions, as we said earlier in this section, some authors have presented explicit formulas for the entropy of known continuous probability distributions. For unknown distributions the first proposed estimators for Shannon's entropy were presented in the seventies by Dmitriev [40], Ahmad [2] and Vasicek [186]. Other entropy estimators, mostly based on the previous ones, were presented in the following years [19,34,73,115,136,188].

Dmitriev [40] was the first to propose the integral estimator of the form

$$H_n = -\int_{A_n} f_n(x)\log_2 f_n(x)\,dx, \tag{2.63}$$

to estimate Shannon's entropy for $d = 1$. Ahmad [2] estimated the entropy using the resubstitution estimator

$$H_n = -\frac{1}{n}\sum_{i=1}^{n} \ln f_n(X_i) \tag{2.64}$$

and showed the consistency of this estimator under certain conditions. Other entropy estimators like the splitting data estimator and the cross-validation estimator, were proposed by Györfi [70–72] and by Ivanov [88] and Hall [73] respectively. These authors also studied the consistency and convergence criteria for these estimators.

2.2.2.4 Rényi's Quadratic Entropy Estimation

Most of the work done in the estimation of entropy is related to Shannon's entropy. However, recent research works use estimators of Rényi's entropy with several applications in learning systems. A particular form of Rényi's entropy is the quadratic one, because, in conjunction with the Parzen Window probability density function estimation with gaussian kernel, it can be estimated in a non-parametric and very practical way. The only estimation involved is the pdf estimation. Estimation of Rényi's quadratic entropy with Parzen window and gaussian kernel was proposed in [198]. Posteriorly, the estimation of the general α-order Rényi's entropy was presented in [45].

Rényi's quadratic entropy H_{R2} can be estimated in the following way:

Let us consider a sample $\{x_1, x_2, ..., x_n\}$ of observations of an i.i.d. random variable $X \in \mathbb{R}$. Let us remember that the Parzen window method estimates the pdf $f(x)$ as

$$f(x) = \frac{1}{nh}\sum_{i=1}^{n} K(\frac{x - x_i}{h}). \tag{2.65}$$

Using a simple Gaussian kernel

$$G(x,1) = \frac{1}{(2\pi)^{\frac{1}{2}}} exp\left(-\frac{1}{2}x^T x\right), \tag{2.66}$$

the estimated pdf $f(x)$ using Parzen window and Gaussian kernel is:

$$f(x) = \frac{1}{nh}\sum_{i=1}^{n} G\left(\frac{x - x_i}{h}, 1\right) = \frac{1}{n}\sum_{i=1}^{n} G\left(x - x_i, h^2\right). \tag{2.67}$$

Noting that the integral of the product of two Gaussians is exactly given by a Gaussian function whose variance is the sum of the variances of the two original Gaussian functions ([197]), Rényi's quadratic entropy can be estimated by

$$\begin{aligned}\hat{H}_{R2}(x) &= -\log \int_{-\infty}^{+\infty} \left(\frac{1}{n}\sum_{i=1}^{n} G(x - x_i, h^2)\right)^2 dx \\ &= -\log\left[\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} G(x_i - x_j, 2h^2)\right]. \end{aligned}\tag{2.68}$$

One can use other kernel functions in this estimator not achieving, however, the same convenient evaluation of the integral.

Rényi's quadratic entropy estimation will be treated with more detail in the following chapter.

The α-order Rényi's entropy (2.43) can be written with an expectation operator as:

$$H_{R\alpha}(X) = \frac{1}{1-\alpha} \log \int_{-\infty}^{+\infty} f^{\alpha}(x)\, dx = \frac{1}{1-\alpha} \log E\left[f^{\alpha-1}(x)\right], \tag{2.69}$$

and approximating this expectation operator by the sample mean, we get,

$$H_{R\alpha}(X) \approx \frac{1}{1-\alpha} \log \frac{1}{n} \sum_{j=1}^{n} f^{\alpha-1}(x_j). \tag{2.70}$$

Substituting the Parzen window estimator 2.65 in 2.70, the nonparametric estimator for the α-order Rényi's entropy is:

$$\begin{aligned} \hat{H}_{R\alpha}(X) &= \frac{1}{1-\alpha} \log \frac{1}{n} \sum_{j=1}^{n} \left[\frac{1}{n} \sum_{i=1}^{n} K_h(x_j - x_i)\right]^{\alpha-1} \\ &= \frac{1}{1-\alpha} \log \frac{1}{n^{\alpha}} \sum_{j=1}^{n} \left[\sum_{i=1}^{n} K_h(x_j - x_i)\right]^{\alpha-1}, \end{aligned} \tag{2.71}$$

where $\frac{1}{n}\sum_{i=1}^{n} K_h(x_j - x_i)$ is the same as $\frac{1}{nh}\sum_{i=1}^{n} K(\frac{x_j - x_i}{h})$.

In [45] the consistency of this estimator, on the condition of consistency of the related Parzen windowing and sample mean, is proved.

Chapter 3

Error Entropy Minimization Algorithm

We start this chapter with an overview of the application of information-theoretic concepts in learning systems and we will present the error entropy minimization algorithm that was used for regression. In order to perform neural network classification following the same approach, we developed the error entropy minimization algorithm for classification. Further on, we present several optimization procedures, including several complements in the algorithm, in order to obtain a faster learning. In this chapter, we also present some experiments showing the results obtained with the new algorithm and also with the implemented optimization procedures. These experiments show the validity of the proposed error entropy minimization (EEM) algorithm.

3.1 Entropy in Learning Systems

Since the introduction by Shannon [174] of the concept of entropy, and the posterior generalization made by Rényi [154], entropy and information theory concepts have been applied to learning systems. As a matter of fact, entropy and relative concepts have several applications in learning systems. Some applications are based on finding the mutual information and the consequent relations between the distributions of the variables involved in a particular problem. Linsker [122] proposed the Infomax principle that consists in maximizing the mutual information between the input and the output of a neural network. Mutual information gives rise to either unsupervised or supervised learning rules depending on how the problem is formulated. We can have unsupervised learning when we manipulate the mutual information between the outputs of the learning system or between its input and output. Examples of these approaches are independent component analysis (ICA) and blind source separation [9, 18]. If the goal is to maximize the mutual information between the output of a mapper and an external desired response, then learning becomes supervised. Figure 3.1 shows a block diagram of a unifying scheme for learning, based on the mutual information criterion.

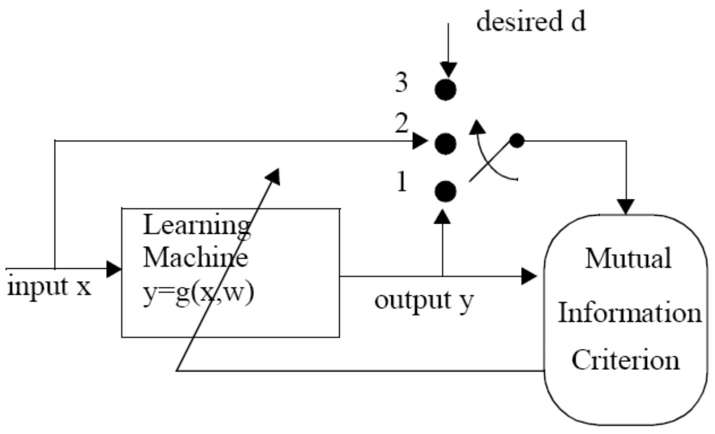


Figure 3.1: Unifying learning models with the mutual information criterion (source [152]).

Depending on the position of the switch, learning belongs to the unsupervised type (position 1 and 2) or supervised type (position 3). Position 1 corresponds to ICA or blind source separation and position 2 to Linsker's Infomax criterion. In position 3, by maximizing the mutual information between the output of a mapper and an external desired response, the learning becomes supervised.

Mutual information is also applied in pattern recognition and classification. Fano's inequality [49], mentioned earlier in Chapter 2, shows that maximizing mutual information decreases the lower bound of the probability of classification error. It relates the probability of error to the conditional entropy. If the goal is to estimate a variable X with a discrete probability mass $p(x)$ by calculating an estimate from another random variable Y characterized by $p(x|y)$, Fano inequality states that

$$H(P_e) + P_e \log(|\mathcal{X}| - 1) \geq H(X|Y), \tag{3.1}$$

that can be weakened to:

$$1 + P_e \log(|\mathcal{X}|) \geq H(X|Y), \tag{3.2}$$

where $P_e = P(x \neq \hat{x})$. Applying 2.41 in 3.2 we get:

$$P_e \geq \frac{H(X) - I(X,Y) - 1}{\log(|\mathcal{X}|)}. \tag{3.3}$$

Since the entropy of X and the $\log(|\mathcal{X}|)$ depend only on the data, we can see that, in order to reduce the lower bound of the probability of error, one must maximize the mutual information between x and y.

Entropy and mutual information were used in a variety of real problems, such as noise detection [190], image alignment [191], cryptology [25], time series prediction [46, 149], channel equalization [163] or blind source separation and ICA [19, 23, 176]. In the last years, Príncipe and his co-workers, presented a series of works where they successfully applied Rényi's entropy and other derived optimality criteria to a huge variety of real problems, some of them related to blind source separation, dimensionality reduction, feature extraction or time series prediction [45, 46, 55, 62, 137, 149, 150, 152, 197, 198]. In [45] it is stated that Príncipe was the first to introduce the terminology "information theoretic learning" (ITL) into the adaptive systems literature.

We may find some examples of applications of entropy in the specific field of neural networks. Works presented in this subject show that information theory concepts may help us to build and tune neural networks to specific problems. Examples of these applications are the works that try to use entropy to determine and define the complexity of the neural network by defining bounds for it [17, 41, 204] or just by generating the neural network based on entropy [85, 183] or yet by performing neural network architecture optimization and pruning [142, 147]. Entropy and information theory can also be combined with neural networks to solve some real problems [28, 180]. Other examples of the relation between entropy and neural networks are the works of Schraudolph [171] and Viola [190].

With the specific goal of performing supervised information-theoretic learning with neural networks, the following approaches have been proposed:

- CIP (Cross Information Potential) - The CIP tries to establish the relation between the pdfs of two variables. These variables could be the output of the network and the desired targets or the output of each layer and the desired targets [199].

- The entropy maximization of the output of the network and simultaneously the minimization of the entropy of the output of the data that belongs to a specific class. This method was proposed in [77], as a way of performing supervised learning without numerical targets.

- MEE (Minimum Error Entropy) - This method consists of the minimization of the error entropy between the outputs of the network and the desired targets. This approach was proposed in [46] and used to make time series prediction.

We made experiments with these three methods with the goal of performing supervised classification. None of them has shown to be appropriate for that task. This led us to develop a new approach for classification problems that we describe in Section 3.2

3.1.1 The Error Entropy Minimization Algorithm for Regression

As we have seen in Chapter 2, Rényi's quadratic entropy can be estimated in a very efficient way. This entropic measure was used in [46, 47] to estimate the entropy of the errors between the output and the desired targets of an adaptive system. In this case, the authors have applied Rényi's quadratic entropy in time delay neural networks of various sizes to perform short-term prediction of Mackey-Glass chaotic time series and nonlinear system identification.

Consider $e = d - y$ to be the error between the desired and the actual output of an adaptive system. The global minimum of Shannon's entropy of the error e is achieved when the pdf of the error is a Dirac-δ function [46], meaning that the global minimum is achieved when all errors are equal.

Regarding Rényi's quadratic entropy of e:

$$\begin{aligned}\hat{H}_{R2}(e) &= -\log \int_{-\infty}^{+\infty} \left(\frac{1}{n} \sum_{i=1}^{n} G(e - e_i, h^2) \right)^2 dx \\ &= -\log \left(\frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} G(e_i - e_j, 2h^2) \right) = -\log V(e), \qquad (3.4)\end{aligned}$$

it was shown that it has the local minimum $e = 0$ and that the global minimum of this estimator is also $e = 0$. In this case the minimum is obtained when all the errors have the same value. Principe [150] calls $V(.)$ the *information potential* in analogy with the potential field in physics.

In regression problems, this approach can lead to a situation where the final solution, the final weight values, may not correspond to a zero-mean error solution. Actually, in [46] it is stated that:

> "One important point to note in training with entropy is that since entropy does not change with the mean of the distribution, the algorithm will converge to a set of optimal weights, which may not yield zero-mean error. However, this can be easily corrected by properly modifying the

bias of the output processing element of the MLP to yield zero mean error over the training data set just after training ends."

In the next section we present the conditions to apply a similar algorithm of error entropy minimization to classification problems.

3.2 The EEM Algorithm for Supervised Classification

As we saw on the previous section, the minimization of the error entropy was already used in learning systems, mainly on regression and time series prediction. Although in [45] is said that "Due to the property that the entropy estimator is invariant to the mean of the underlying density of the samples as is the actual entropy, in supervised learning entropy cannot be used to force the mean of the error signal to zero", we will show in this section how to use the minimization of the error entropy in classification problems [164].

We will perform classification tasks, using as cost function Rényi's quadratic entropy of the error between the output of the neural network and the desired targets: this yields the Error Entropy Minimization algorithm, EEM. To perform the neural network learning, we apply the back-propagation algorithm and consequently the gradient descent method for entropy minimization. This same algorithm with Shannon's entropy was proposed posteriorly in [178], also with good results when comparing it with MSE and Cross-entropy.

Let us see how to estimate Rényi's quadratic entropy of multi-dimensional random variables.

Let $a = a_i \in \mathbb{R}^m$, $i = 1, ..., N$, be a set of samples from the output $Y \in \mathbb{R}^m$ of a mapping $\mathbb{R}^n \mapsto \mathbb{R}^m : Y = g(w, x)$, where w is a set of Neural Network weights. The Parzen window method estimates the pdf $f(y)$ as

$$f(y) = \frac{1}{Nh^m} \sum_{i=1}^{N} K\left(\frac{y - a_i}{h}\right), \tag{3.5}$$

where N is the number of data points, K is a kernel function, and h the bandwidth or smoothing parameter. The Gaussian kernel

$$G(y;0,\Sigma) = \frac{1}{(2\pi)^{\frac{m}{2}}|\Sigma|^{\frac{1}{2}}} exp\Big(-\frac{1}{2}y^T\Sigma^{-1}y\Big), \tag{3.6}$$

will be used in his simple form, a spherical symmetric Gaussian kernel with zero mean and diagonal covariance matrix $\Sigma = \mathbf{I}$, where $\mathbf{I}$ is the $m \times m$ identity matrix:

$$G(y;0,\mathbf{I}) = \frac{1}{(2\pi)^{\frac{m}{2}}|\mathbf{I}|^{\frac{1}{2}}} exp\Big(-\frac{1}{2}y^T\mathbf{I}^{-1}y\Big). \tag{3.7}$$

The estimated pdf $f(y)$ using Parzen window and simplified Gaussian kernel is:

$$\begin{aligned} f(y) &= \frac{1}{Nh^m}\sum_{i=1}^{N} G\left(\frac{y-a_i}{h}, \mathbf{I}\right) \\ &= \frac{1}{Nh^m}\sum_{i=1}^{N} G\left(\frac{y}{h}; \frac{a_i}{h}, \mathbf{I}\right). \end{aligned} \tag{3.8}$$

Substituting 3.8 in 2.44 Rényi's quadratic entropy, can be estimated by

$$\begin{aligned} &\hat{H}_{R2}(y) = \\ &= -\log \int_{-\infty}^{+\infty} \left[\frac{1}{Nh^m}\sum_{i=1}^{N} G\left(\frac{y}{h}; \frac{a_i}{h}, \mathbf{I}\right)\right]^2 dy \\ &= -\log \frac{1}{N^2h^{2m}} \int_{-\infty}^{+\infty} \left[\sum_{i=1}^{N} \frac{1}{(2\pi)^{\frac{m}{2}}|\mathbf{I}|^{\frac{1}{2}}} exp\Big(-\frac{1}{2}(\frac{y-a_i}{h})^T\mathbf{I}^{-1}(\frac{y-a_i}{h})\Big)\right]^2 dy \\ &= -\log \frac{1}{N^2h^{2m}} \int_{-\infty}^{+\infty} \left[\sum_{i=1}^{N} \frac{h^m}{(2\pi)^{\frac{m}{2}}(|h^2\mathbf{I}|)^{\frac{1}{2}}} exp\Big(-\frac{1}{2}(y-a_i)^T\ (h^2\mathbf{I})^{-1}\ (y-a_i)\Big)\right]^2 dy \\ &= -\log \frac{1}{N^2} \int \left[\sum_{i=1}^{N} G(y;a_i,h^2\mathbf{I})\right]^2 dy. \end{aligned}$$

Given that ([197]):

$$\int G(x;a,A)G(x;b,B) = G(a-b;0,2AB), \tag{3.9}$$

Rényi's quadratic entropy is estimated by:

$$\hat{H}_{R2}(y) = -\log\left(\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} G(a_i - a_j; 0, 2h^2\mathbf{I})\right) = -\log V(a). \qquad (3.10)$$

For simplicity and since the log is a monotonic increasing function, only $V(a)$ will be used for entropy minimization purposes.

To apply the gradient descent method, we need to compute the derivative of $V(a)$ with respect to a:

$$\begin{aligned}\frac{\partial V}{\partial a} &= \frac{\partial}{\partial a}\left(\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} G(a_i - a_j; 0, 2h^2\mathbf{I})\right)\\ &= -\frac{1}{2N^2h^2}\sum_{i=1}^{N}\sum_{j=1}^{N} G(a_i - a_j; 0, 2h^2\mathbf{I})(a_i - a_j). \qquad (3.11)\end{aligned}$$

Let us now see how to use Rényi's quadratic entropy as cost function in a neural network classification problem (Fig. 3.2).

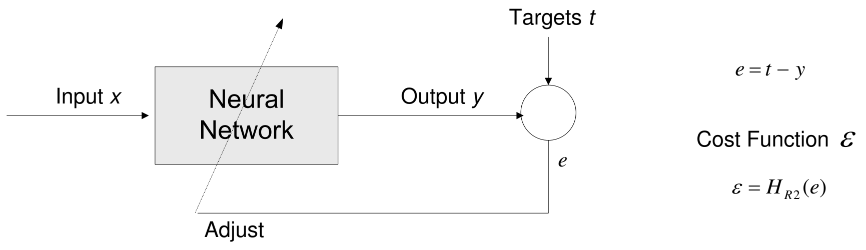


Figure 3.2: Neural network learning with entropic cost function.

Let $d \in \mathbb{R}^m$ be the desired targets and Y the network output from the classification problem and $e_i = d_i - y_i$ the error for each data sample i of a given data set. The error entropy minimization approach [46] used in time series prediction, states that Rényi's Quadratic Entropy of the error, with pdf approximated by Parzen window with Gaussian kernel, has minima along the line where the error is constant over the whole data set. Also the global minimum of this entropy is achieved when the pdf of the error is a Dirac delta function.

Taking the quadratic entropy of the error

$$\hat{H}_{R2}(e) = -\log\left(\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} G(e_i - e_j; 0, 2h^2\mathbf{I})\right) = -\log V(e), \tag{3.12}$$

we clearly see that this entropy will be minimum when the differences of all the error pairs $(e_i - e_j)$ are zero. This means that the errors are all the same. In classification problems with separable classes, the goal is to get all the errors equal to zero, meaning that we don't get any errors in the classification. In classification problems with non separable classes, the goal is to achieve the Bayes error.

In the following we prove that, in classification problems, by imposing some conditions to the output range and target values, the EEM algorithm makes the error convergent to zero. The objective is to minimize the entropy of the error $e = d - y$ and, as stated above, to achieve the goal of $e = 0$ for all data samples.

Corollary 1. *Consider a two class supervised classification problem with a unidimensional output vector. Let $y \in [r, s]$ be the output of the network and $d \in \{a, b\}$ be the target vector of the desired output. If $r = a$, $s = b$ and $a = -b$ then the application of the EEM algorithm forces the errors on each data point to be equal to zero.*

Proof. Define the targets as $d \in \{-a, a\}$ and consider the output of the network as $y \in [-a, a]$. The errors are given by $e = d - y$.

If the true target for a given input x_i is $\{a\}$ then the error e_i varies in $P = [0, 2a]$.

If the true target for a given input x_j is $\{-a\}$ then the error e_j varies in $Q = [-2a, 0]$.

Since the minimization of the entropy of the error makes the errors all have the same value, r, we get $e_i = e_j = r$.

But r must be in P and Q. Since $P \cap Q = \{0\}$ follows that $r = 0$ and $e_i = e_j = 0$.

□

A similar proof can be made for multidimensional output vectors.

In the EEM algorithm, the error entropy estimation is different according to the vector e dimension. This dimension depends on the number of MLP output neurons that depend on the number of classes, C, and their coding. If we use binary coding [1], the dimension of vector e is $\lceil log_2 C \rceil$, whereas if we use the one-out-of-C coding, the number of output neurons, as well as the dimension of vector e, is equal to the number of classes [2]. In the experiments that we performed we obtained good results for both codings, but we suggest the use of binary coding only when the number of classes is a power of 2; otherwise, we get an excessive number of outputs compared with the number of classes.

In Fig. 3.3 we show examples of the support space of the error vector distribution for two-class and three-class problems (output vector dimension corresponds to the one-out-of-C encoding).

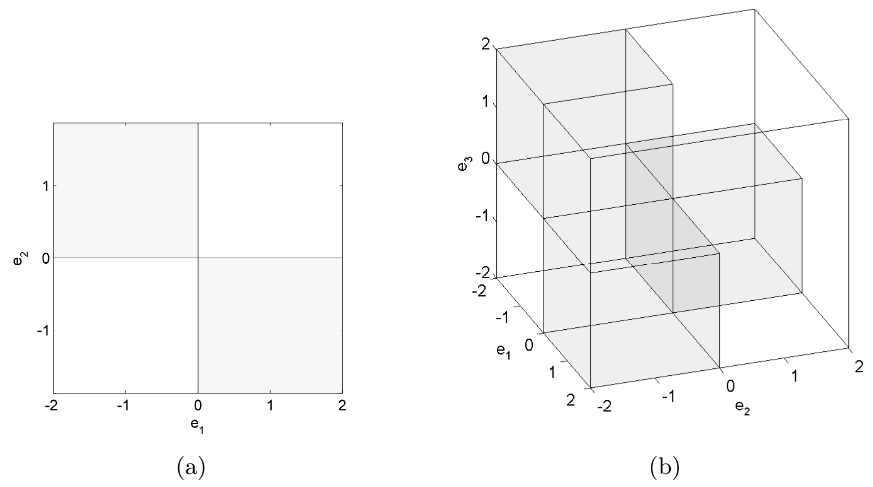


Figure 3.3: The support space (shadowed regions) for the error distribution in a (a) 2-dimensional and (b) 3-dimensional outputs (a two-class and a three-class problem).

To apply the EEM algorithm the entropy gradient at each point is back-propagated into the MLP using the back-propagation algorithm (the same used by the MSE algorithm). The update of the neural network weights is performed

[1]When using binary coding we can use a single output neuron to solve a two-class problem. The single output y defines, for example, class 1 if $y < 0$ and class 2 if $y \geq 0$. Targets are encoded as -1 and 1. For instance a four-class problem only needs 2 outputs.

[2]In the one-out-of-C coding the target vectors are encoded such that $d = [-1, ..., 1, ..., -1]$, where the 1 appears at the kth component for a pattern belonging to class C_k.

using $\Delta w = \pm\eta\frac{\partial V}{\partial w}$. The $\pm$ means that we can either maximize $(+)$ or minimize $(-)$ the entropy.

As we have seen, by minimizing Rényi's Quadratic Entropy of the error, applying the back-propagation algorithm, we find the weights of the neural network that yield good results in classification problems as we show in the following experiments. This algorithm represents a new way of performing supervised classification by using as cost function the entropy of the error between the output of the MLP and the desired targets:

$$E_H = H_{R2}(e). \tag{3.13}$$

Some aspects in the implementation of the algorithm will be studied in detail in Section 3.3; for example, how to choose h and η and make their values adjust during the training phase to improve the classification performance.

3.2.1 Preliminary Experiments

We made several preliminary experiments, using multi-layer perceptrons, to show the application of the EEM algorithm to data classification and we have compared the results with the MSE. The learning rate η and the smoothing parameter h were experimentally selected; however, we will present later several optimization procedures that overcome the need for exhaustive experiments to obtain the ideal values for these parameters.

In the first experiment we created a data set consisting of 200 data points, constituting 4 separable classes (Fig. 3.4).

Several [2:n_h:4] MLP's [3] were trained and tested 40 times, 150 epochs, using EEM and also MSE. We made n_h (the number of neurons in the hidden layer) vary from 3 to 6. We used the 2-fold cross validation method. The results of the first experiment are shown in Table 3.1. The last row (STD) presents the

[3]We used in this experiment the one-out-of-C coding, and this is the reason for having 4 outputs for a 4-class problem.

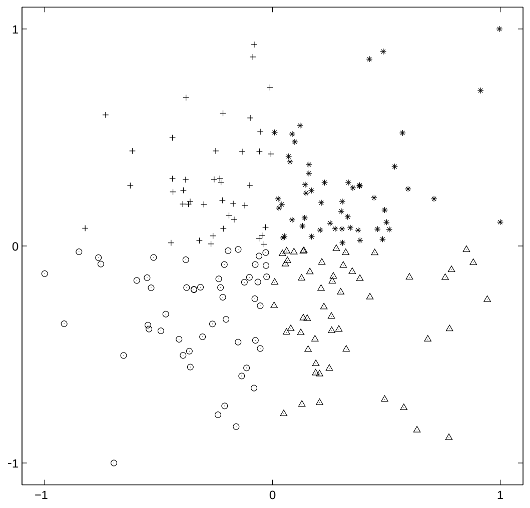


Figure 3.4: Data set for the first experiment with the EEM algorithm.

standard deviation of the errors over the different n_h sessions. The classification errors are smaller in EEM than in MSE.

Table 3.1: The test error and standard deviations for the first experiment. Last row, STD, represents the standard deviation of the mean errors obtained with all n_h values for EEM and MSE.

n_h	EEM	MSE
3	2.43(1.33)	2.93(1.46)
4	2.20(1.20)	2.55(1.24)
5	**2.01**(1.09)	2.64(1.13)
6	2.09(1.02)	2.91(1.73)
STD	**0.18**	0.19

In Fig. 3.5, Fig. 3.6 and Fig. 3.7 we depicted the errors produced in steps 1, 10 and 40 respectively, of this first experiment. Since we have a neural network with four outputs, the error vectors e form a 100×4 matrix. This matrix of errors is represented in each figure by a 4×4 matrix of axes with scatter plots of each column of the matrix against the other columns. Diagonals are the histograms of each column of the matrix.

Analyzing these graphs we can see that the errors are located (more visible in the first iteration) in the three support regions, as depicted in Fig. 3.3b and that the error vectors converge to the origin $(0, 0, 0, 0)$ during the experiment.

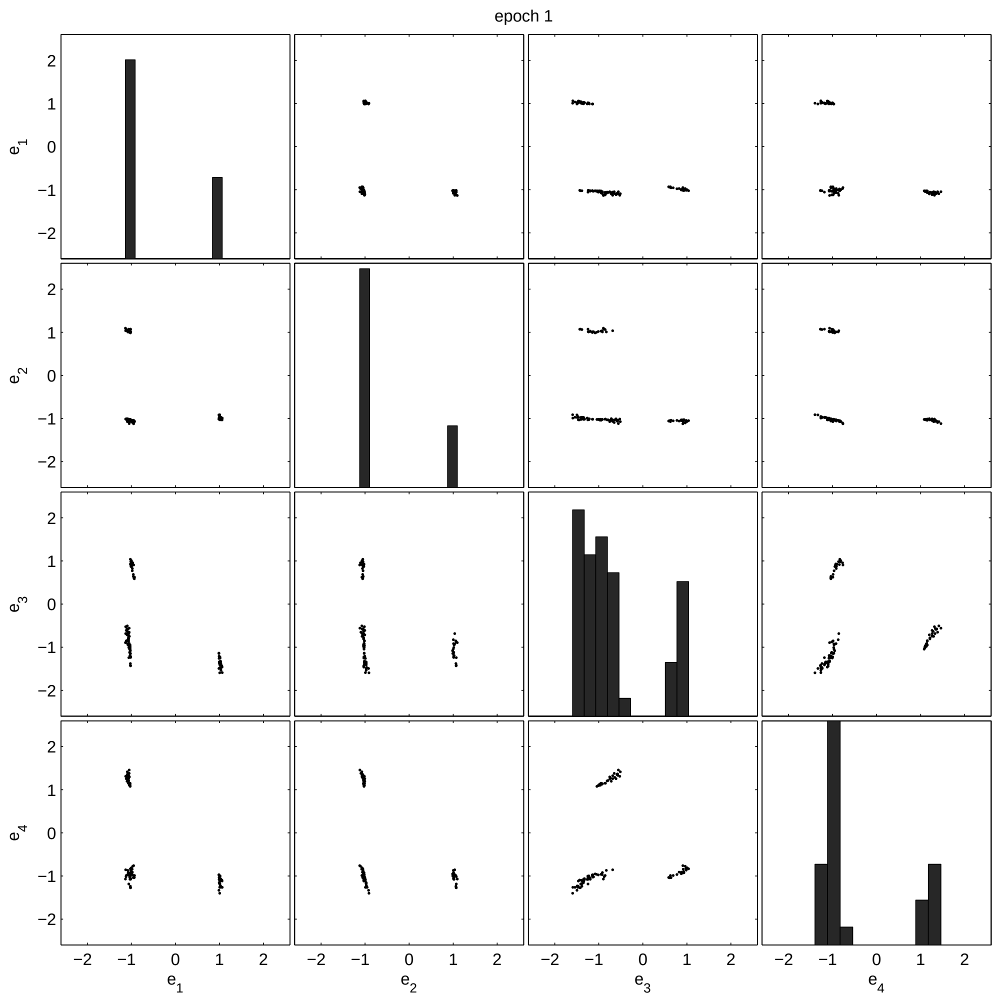


Figure 3.5: Errors in the first iteration of an experiment with data set of Fig. 3.4.

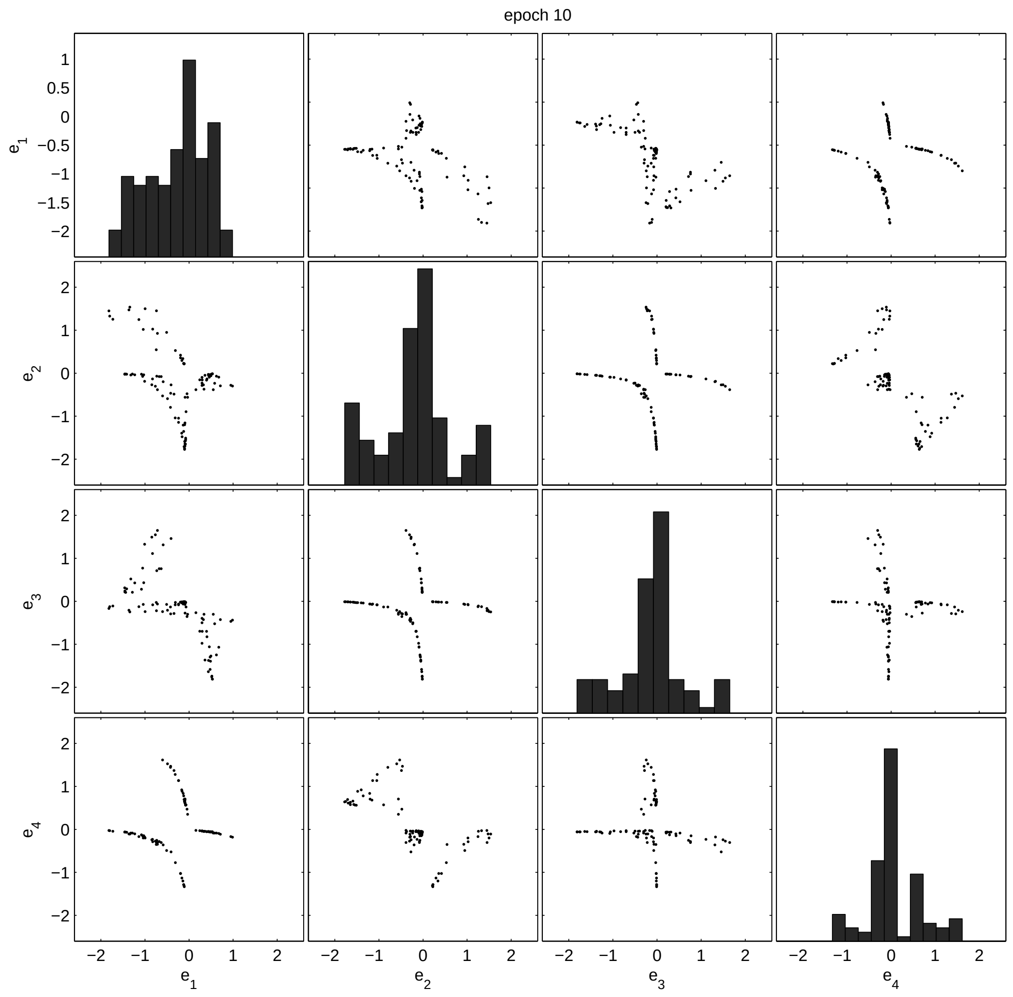


Figure 3.6: Errors in the tenth iteration of an experiment with data set of Fig. 3.4.

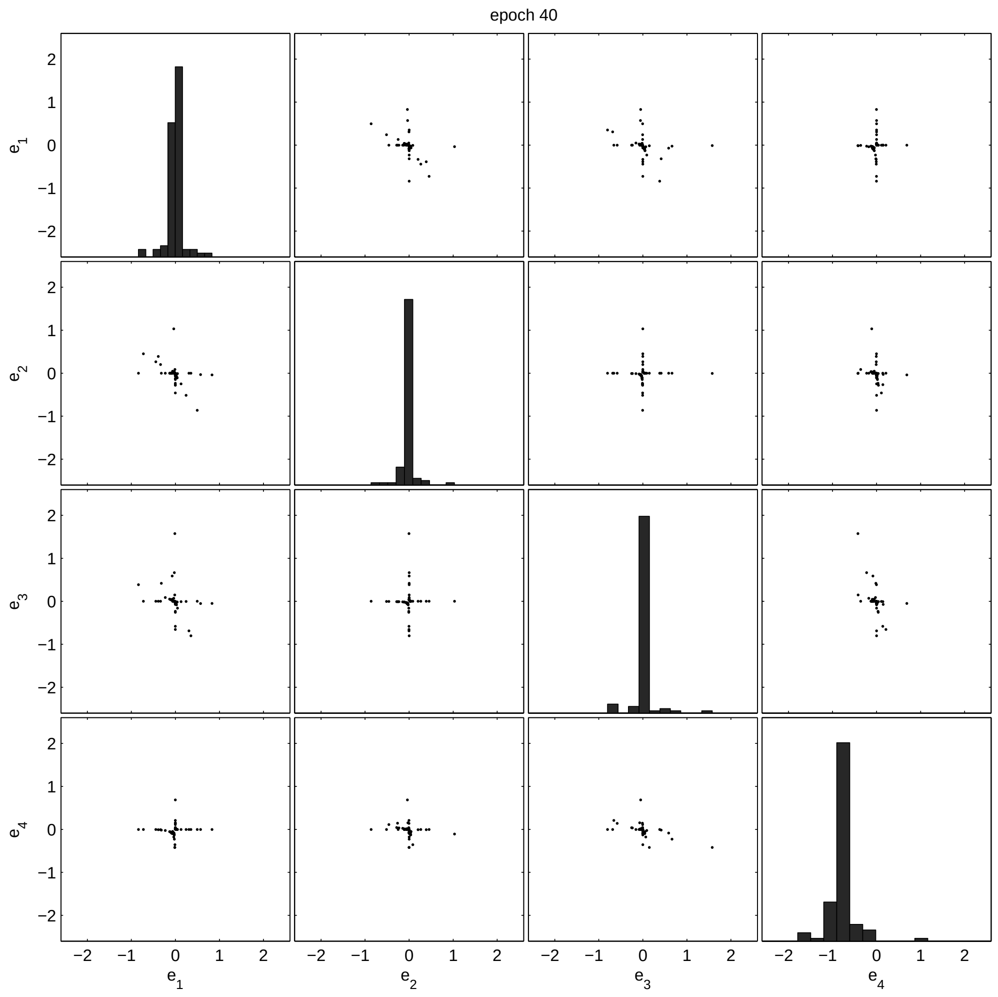


Figure 3.7: Errors in the fortieth iteration of an experiment with data set of Fig. 3.4.

In the following experiments, we used the data sets Diabetes, Wine and Iris (Appendix A contains a summary of the characteristics of all the real data sets used in this work).

Several MLP's with one hidden layer were trained and tested 20 times, 150 epochs, for EEM and also for MSE. The 2-fold cross validation was used. The results of these experiments are presented in Table 3.2.

Table 3.2: The error results of the second set of experiments. Last row, STD, represents the standard deviation of the mean errors obtained with all n_h values for EEM and MSE.

	Diabetes		Wine		Iris	
n_h	EEM	MSE	EEM	MSE	EEM	MSE
2	23.80(0.94)	28.40(4.87)	3.62(1.30)	9.72(10.60)		
3	23.94(0.97)	27.25(4.72)	3.81(1.00)	4.27(3.77)	4.36(1.12)	4.72(4.75)
4	23.99(1.52)	26.42(4.53)	**1.94**(0.72)	3.03(1.08)	4.43(1.30)	4.75(1.27)
5	23.80(1.04)	25.10(1.80)	2.50(1.01)	3.20(1.83)	4.38(1.34)	4.15(1.32)
6	24.10(1.33)	24.70(1.80)	2.47(1.20)	3.06(1.43)	4.30(1.16)	**3.97**(1.05)
7	24.10(0.90)	24.40(1.06)	2.44(1.00)	2.39(1.50)	4.41(1.42)	5.18(4.74)
8	23.90(0.71)	23.90(1.18)	2.16(0.92)	2.92(1.07)	4.31(1.27)	4.65(1.32)
9	24.30(1.42)	24.00(0.95)	2.22(0.83)	2.50(1.35)		
10	**23.60**(0.86)	24.10(1.20)	2.31(0.51)	2.95(1.29)		
11	24.02(1.00)	27.41(5.19)				
12	24.93(3.24)	27.64(5.04)				
STD	**0.35**	1.69	**0.65**	2.29	**0.05**	0.44

The results show, in almost all experiments, a small, but better performance of the EEM algorithm. They also show, especially in the second set of experiments, that the variation of the error along n_h is smaller in the EEM than in the MSE. This can be seen in the last row, STD, the standard deviation of the mean errors obtained with all n_h values for EEM and MSE. This could mean that the relation between the complexity of the MLP and the results of the EEM algorithm is not so tight as for the MSE algorithm. In other words, we obtained some empirical evidence that EEM generalizes better than MSE. This is also hinted by the systematically lower standard deviation of the errors of the EEM when compared with MSE. These findings can be understood taking into account that entropy is better at characterizing the pdf of the errors than simply MSE, which only characterizes their variance.

3.3 Optimization of the EEM Algorithm

We performed several adaptations to the EEM algorithm, trying to achieve a faster convergence and a better performance. We made some studies focused on the learning rate parameter [169], on the smoothing parameter [167] and on combining batch and online training [166]. In [138] we can find some popular techniques for parameter optimization, applied to information theoretic learning, particularly in unsupervised feature extraction and frequency-doubling problems.

We started the optimization of EEM algorithm by trying to make the kernel smoothing parameter (kernel window size) h an updated variable along the training process, namely proportional to the error variance. This strategy was based on the fact that, as we approach the optimal solution, the errors tend to zero (m-tuples of zeros) and so it makes sense to decrease h since the points are all close to each other. However, the estimates of the error entropy and its derivative depend on the values of h; smaller h originates higher entropy estimates. If we reduce the value of h from one iteration to another, by that simple fact, the value of the entropy is higher, the opposite to our objective of minimization of the entropy of the error in consecutive iterations. If, at each algorithm iteration, we manage to minimize the entropy function, we can, at least theoretically, get an optimal solution. The problem found when using a variable h was that, in the proximity of the minimum training error, the algorithm became very unstable, loosing the capability of convergence. In Fig. 3.8 we show the behavior of the training curve when using a variable h, proportional to the error variance. We can see that h varies with the training error but, at a certain stage (around epoch 100 and training error 10%), the algorithm loses stability never returning to a steady state.

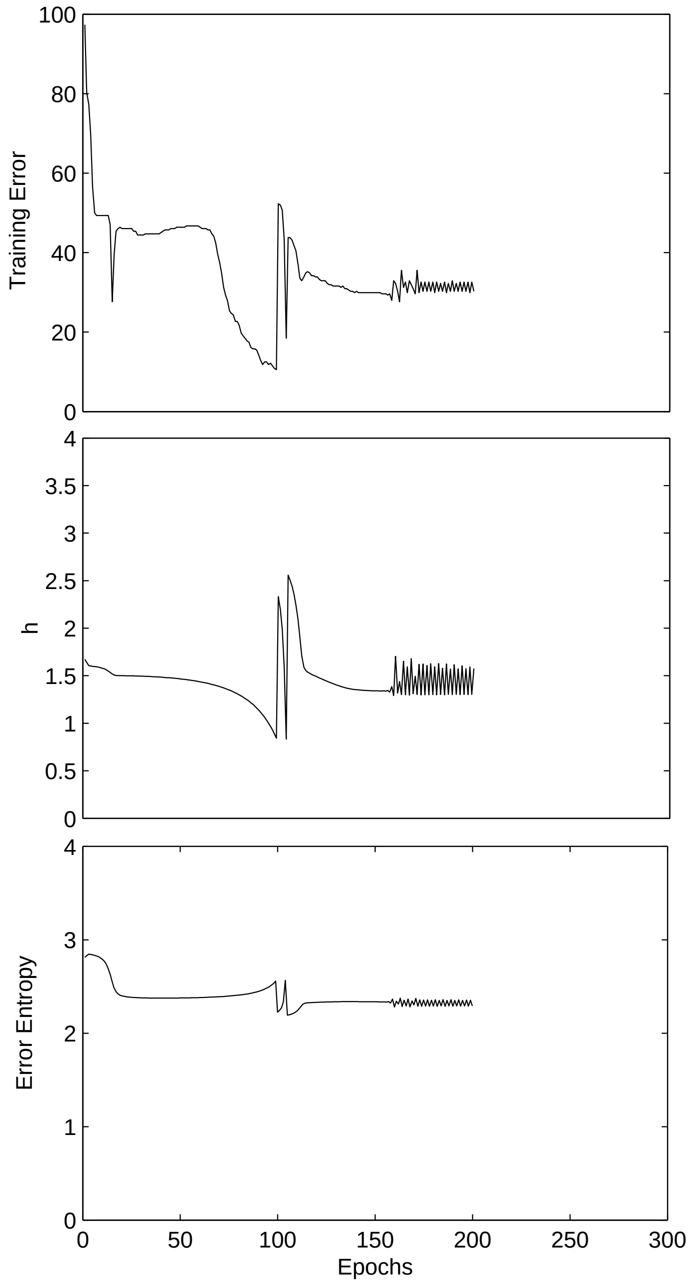


Figure 3.8: The observable instability of EEM algorithm when using variable h in an experiment with data set 2VowelsPB.

After performing several experiments with different approaches trying to overcome this limitation, and having observed the same behavior we left behind the possibility of using a variable h along the learning process and we started the improvement of EEM algorithm by implementing an adaptive learning rate η and a fixed smoothing parameter h.

3.3.1 Adaptive Learning Rate

The second implemented optimization procedure was an adaptive learning rate η. As we have seen in Chapter 2, several authors have shown that, by adapting the η value along the learning process, one can get a better and faster convergence in a neural network using mean squared error as cost function. With the experiments to be described later, we tried to grasp on what conditions we could apply an adaptive learning rate to a multi-layer perceptron trained with the EEM algorithm. To get a faster convergence, still obtaining good classification results, we first planed to adjust η as a function of the error entropy similarly to adjusting it as a function of the MSE. We will see how the variation of the learning rate along the training process can yield good results.

As we saw in Section 3.2, the gradient of Rényi's Quadratic Entropy of the error is back-propagated into the MLP in the same way as with the MSE algorithm. The update of the neural network weights is performed using $\Delta w = \pm\eta\frac{\delta V}{\delta w}$.

The variability of the learning rate follows the simple but effective rule mentioned in Section 2.1.4.4: if the error entropy decreases between two consecutive epochs of the training process, then the algorithm produces an increase in the learning rate parameter. Similarly, if the error entropy increases between two consecutive epochs, then the algorithm produces a decrease in the learning rate parameter and, furthermore, it restarts the update step, i.e. we recover and use the previous "good" values of all the neural network parameters. In Section 3.3.3 we also implemented with good results two other different rules: the Silva and Almeida's rule [177] and the resilient backpropagation (RProp) algorithm [156].

The rule for learning rate updating is:

$$\eta^{(n)} = \begin{cases} \eta^{(n-1)}u & \text{if} \quad H_{R2}^{(n)} < H_{R2}^{(n-1)} \\ \eta^{(n-1)}d \ \wedge \text{restart} & \text{if} \quad H_{R2}^{(n)} \geq H_{R2}^{(n-1)} \end{cases}, u > 1,\, d < 1, \tag{3.14}$$

where $\eta^{(n)}$ and $H_{R2}^{(n)}$ are, respectively, the learning rate and Rényi's Quadratic entropy of the error at the nth iteration and u and d are the increasing and decreasing factors.

We performed several experiments in order to find good values for u and d. In one of these tests, that we present here, we used the bi-dimensional 2VowelsPB data set.

In Fig. 3.9 we show an example of the training phases with fixed learning rate, FLR (dotted lines), and with rule 3.14 variable learning rate, VLR (solid lines), of two experiments that have produced the smallest classification errors. The use of VLR produces a continuous decreasing entropy curve and a minimum training error is achieved.

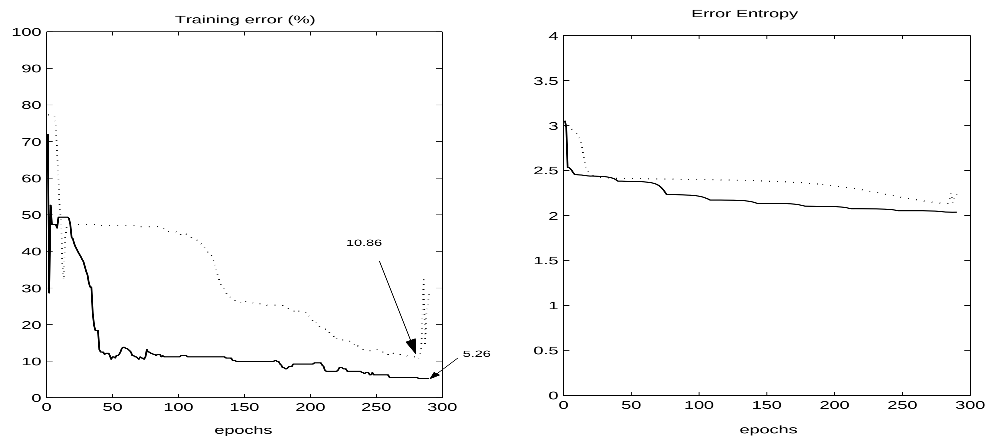


Figure 3.9: Two best results for FLR (doted) and VLR (solid).

In Table 3.3 we present the results of an experiment made with different values for u and d. The column (restart) indicates the number of times that the algorithm restarts the update step. These experiments suggest that, if the algorithm produces an increase on the entropy, then the learning rate should be decreased by a considerable factor. Based in the several tests that we have performed and in the fact that our errors are always limited to a restricted set, due to the conditions mentioned in Section 3.2, we found out that d and u should have values around 0.2 and 1.2, respectively. The solid line in Fig. 3.9 represents a case with $d = 0.2$ and $u = 1.2$.

Table 3.3: Results for different values for u and d.

u	d	*restart*	Training Error
1.2	0.2	36	5.26
1.2	0.4	65	5.26
1.2	0.6	112	5.59
1.2	0.8	256	5.92
1.4	0.2	64	5.59
1.4	0.4	115	5.26
1.4	0.6	197	24.34
1.4	0.8	465	5.59
1.6	0.2	90	5.59
1.6	0.4	154	5.26
1.6	0.6	279	5.59
1.6	0.8	640	5.26
1.8	0.2	112	5.59
1.8	0.4	193	5.59
1.8	0.6	352	5.59
1.8	0.8	801	5.26

3.3.1.1 Experiments: EEM-VLR versus MSE-VLR

We made a first experiment, using multilayer perceptrons (MLP), to show the application of the Error Entropy Minimization with Variable Learning Rate (EEM-VLR) algorithm to data classification and compare it with Mean Square Error with Variable Learning Rate (MSE-VLR) algorithm. In this experiment we used the same data set 2VowelsPB. Several $[2 : n_h : 4]$ MLP's were trained and tested

40 times, 300 epochs, using the EEM-VLR and also the MSE-VLR. We made n_h vary from 3 to 20. The 2-fold cross validation method was used. The results of this experiment are shown in Table 3.4.

Table 3.4: Classification errors for EEM-VLR and MSE-VLR.

n_h	EEM-VLR	MSE-VLR
3	9.68(3.31)	24.46 (9.84)
4	**8.18**(2.30)	17.49 (9.10)
5	8.54(2.81)	15.86 (8.61)
6	8.73(2.51)	13.80 (7.61)
7	9.22(4.60)	14.35 (8.26)
8	8.67(2.60)	12.29 (6.58)
9	9.03(2.51)	12.46 (7.48)
10	8.77(1.88)	11.95 (6.64)
11	9.90(3.55)	11.34 (6.66)
12	9.30(2.56)	10.67(6.01)
13	10.16(2.83)	9.44(3.84)
14	10.01(2.50)	9.46(3.63)
15	10.14(2.16)	8.61(1.30)
16	11.50(5.54)	9.22(3.48)
17	10.72(1.94)	9.77(4.97)
18	12.68(4.47)	10.61(6.18)
19	12.62(4.10)	9.71(4.76)
20	12.94(5.80)	9.10(3.88)

We see in Table 3.4 that EEM-VLR algorithm produces better results when compared to the MSE-VLR algorithm. The smallest error is 8.18. The MSE-VLR algorithm produced better results only for larger values of the number of neurons in the hidden layer. However, this could be due to over-fitting, since we used a fixed number of epochs (no early stopping). We also see that similar results are achieved with less complex MLP's using the EEM-VLR algorithm (Fig. 3.10). This may suggest that, with EEM, we need less complex neural networks, compared to MSE in order to solve a particular classification problem.

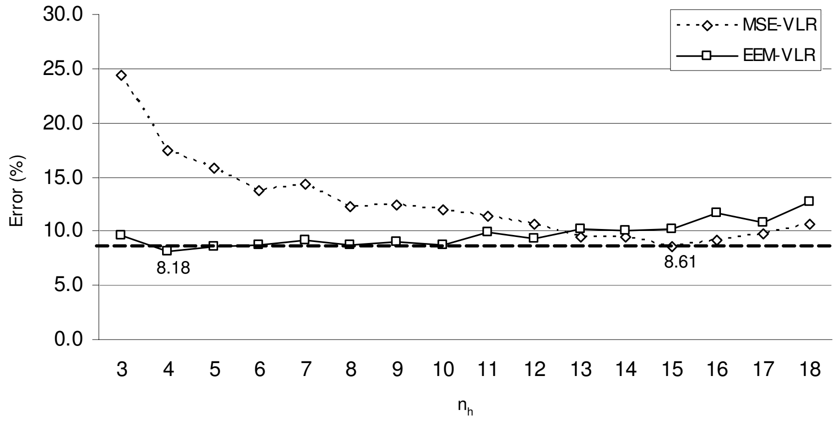


Figure 3.10: Results comparison between EEM-VLR and MSE-VLR for data set 2VowelsPB.

Two more experiments were made applying the two algorithms to the data sets Diabetes and Wine.

Several MLP's were trained and tested 20 times, 120 epochs, with $d = 0.2$ and $u = 1.2$. Again the 2-fold cross validation was used. The initial learning rate value for our experiments is usually around 0.1, however one can use a different value by observing the behaviour of the training curve[4]. The results of these experiments are shown in Table 3.5. Again, the best results (bold), in this two classification problems, were achieved with the EEM-VLR algorithm.

We present in Fig. 3.11 the results comparison between EEM-VLR and MSE-VLR with data sets Diabetes and Wine and we can see, as in the previous experiment, that with EEM less complex MLP's are needed to get similar results when solving these two classification problems.

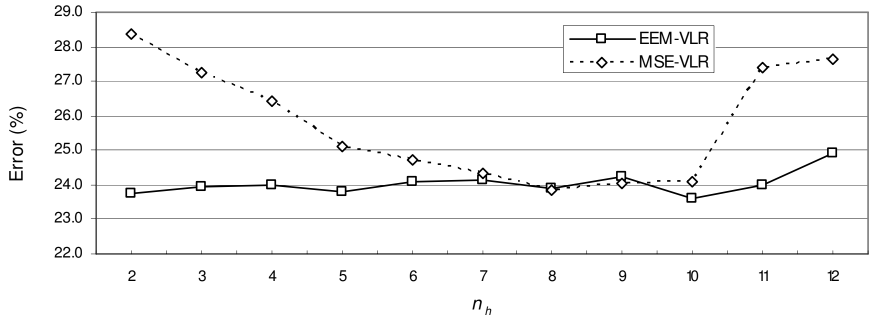
[4]If a very small value is used for a particular problem we will get a learning curve with an initial very flat region indicating that the weights are updated by a very small amount. If we use a high value for the learning rate, the training curve will have a initial very fast decrease or, if an extremely high value is used, the algorithm can even fail to converge.

Table 3.5: Classification errors for the EEM-VLR and the MSE-VLR algorithms.

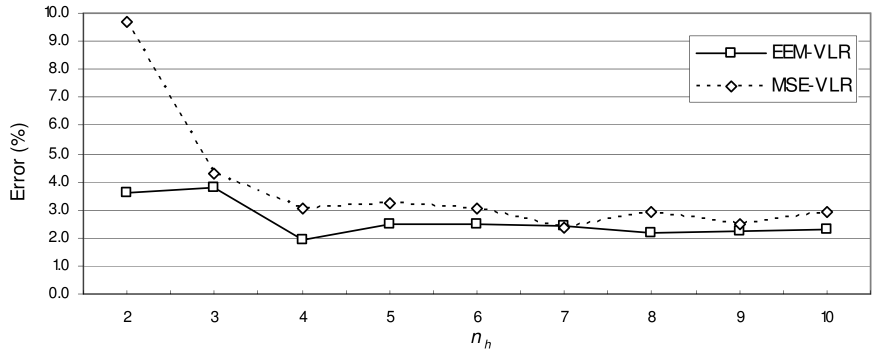
	Diabetes		Wine	
n_h	EEM-VLR	MSE-VLR	EEM-VLR	MSE-VLR
2	23.80(0.94)	28.40(4.87)	3.62(1.3)	9.72(10.6)
3	23.94(0.97)	27.25(4.72)	3.81(1.00)	4.27(3.77)
4	23.99(1.52)	26.42(4.53)	**1.94**(0.72)	3.03(1.08)
5	23.79(1.04)	25.12(1.80)	2.50(1.01)	3.20(1.83)
6	24.07(1.33)	24.73(1.80)	2.47(1.20)	3.06(1.43)
7	24.12(0.90)	24.35(1.06)	2.44(1.00)	2.39(1.50)
8	23.90(0.71)	23.87(1.18)	2.16(0.92)	2.92(1.07)
9	24.26(1.42)	24.04(0.95)	2.22(0.83)	2.50(1.35)
10	**23.62**(0.86)	24.08(1.20)	2.31(0.51)	2.95(1.29)
11	24.02(1.00)	27.41(5.19)		
12	24.93(3.24)	27.64(5.04)		

3.3.2 The Smoothing Parameter

Having performed the first improvement of the EEM algorithm by using an adaptive learning rate during the training process, we searched for further improvement by studying the influence of the value of the smoothing parameter h in the performance of the neural network. As a matter of fact, from the many experiments performed, we came to recognize that the smoothing parameter is the most important factor and the one that has more influence in the final results of a classification problem, when using the EEM algorithm. The choice of the smoothing parameter in the Parzen Window estimation of the probability density function for the computation of the entropy and its gradient is a difficult issue of the EEM algorithm. In the following subsection we present a formula yielding the value of the smoothing parameter depending on the number of data samples and on the neural network output dimension. Several experiments with real data sets were made in order to show the validity of the proposed formula.



(a) Classification errors for data set Diabetes.



(b) Classification errors for data set Wine.

Figure 3.11: Results comparison between EEM-VLR and MSE-VLR for data sets Diabetes and Wine.

3.3.2.1 Tuning the Smoothing Parameter

One of the problems of pdf estimation using the Parzen Window method, besides the choice of the kernel, is the choice of the smoothing parameter h. In the EEM algorithm the value of h depends on: the different codings of the number of classes; the number of data samples; the dimension m of the vector e.

Let us remind that for continuous $f(x)$, the estimated density function will converge to the true density as $N \to \infty$ when:

$$h \to 0 \qquad \text{and} \qquad Nh \to \infty. \tag{3.15}$$

We have also seen in Chapter 2 that, for multidimensional cases, assuming normal distributions and using the normal kernel, Bowman and Azzalini [24] proposed the formula:

$$h_{op} = s\left(\frac{4}{(m+2)N}\right)^{\frac{1}{m+4}}, \tag{3.16}$$

where s is the sample standard deviation, N is the number of samples and m is the dimension of vector x. An important fact that impedes, in our case, the use of formula 3.16 is that our algorithm uses the entropy of e as a control variable, i.e., the algorithm progresses only if the entropy at a given iteration is smaller than at the previous one. Since the entropy value is proportional to the smoothing parameter value used to compute it, if one uses a value for h proportional to the variance of e, one might be increasing, by this simple fact, the entropy value and the algorithm fails to converge to a minimum. This means that we are limited to use a value for h that does not depend on s. Considering that the variable e takes values, in the unidimensional case, in the interval $[-2, 2]$, and that the maximum standard deviation in this case is 2, we considered using this value to replace the standard deviation in formula 3.16:

$$h_{op} = 2\left(\frac{4}{(m+2)N}\right)^{\frac{1}{m+4}}. \tag{3.17}$$

Note that, in the EEM algorithm, we only need to compute the entropy and its gradient; we do not need to estimate the probability density function of e. This is a relevant fact because, in the gradient descent method, more important than computing with extreme precision the gradient is to get with relative precision its direction. Also, the tentative of estimating with extreme accuracy the probability density function by using very small values of h, causes the estimations of entropy to have high variability in consecutive epochs. This fact can also lead to the occurrence of local minima. Given these considerations, we do not hesitate in using h values higher than the ones usually proposed for pdf estimation. Taking into account the experimental results with several data sets we tried to formulate a rule that yields higher values of h for smaller data sets than those obtained with formula 3.17 and still inducing the same behavior. We then arrived at the following formula with behavior similar to 3.17:

$$h_{op} = 25\sqrt{\frac{m}{N}}. \tag{3.18}$$

Notice the decreasing behavior with N and increasing behavior with m as desired. A comparison between the values of h obtained with formulas 3.17 and 3.18 for different values of m is shown in Fig. 3.12.

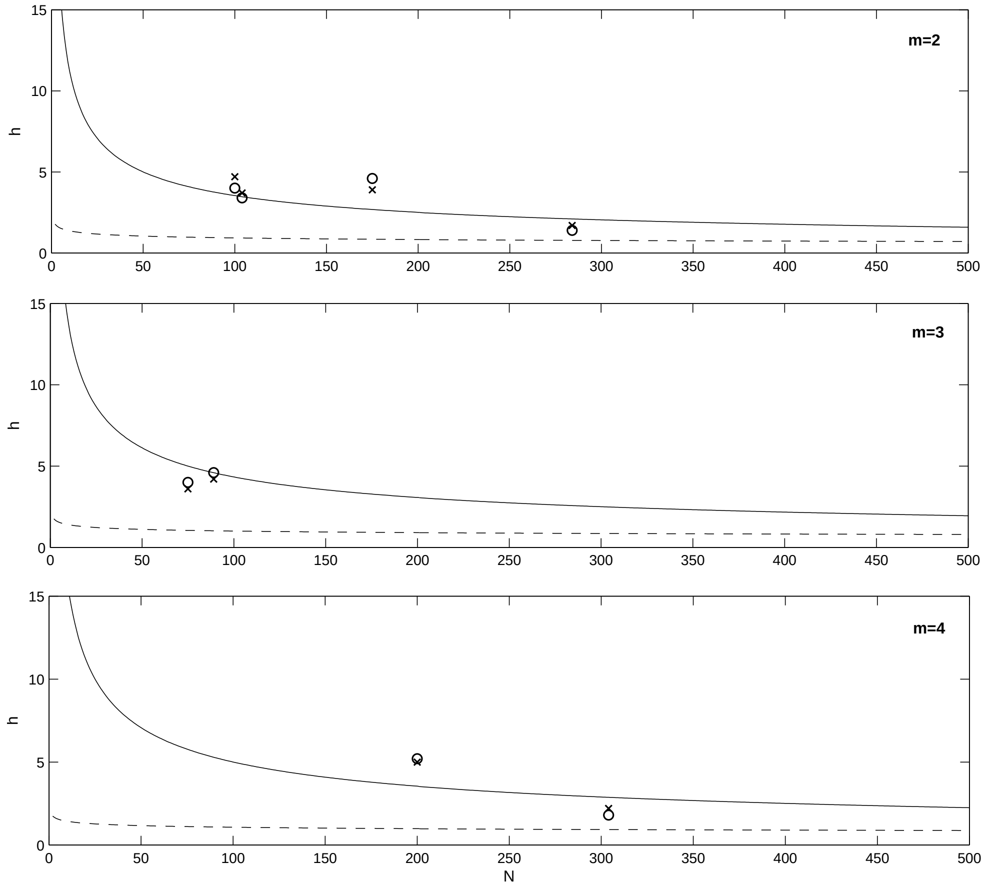


Figure 3.12: Value of h for formulas 3.17 (dashed) and 3.18 (solid) for m=2, 3 and 4. (Marked points refer to experiments with data sets summarized in Table 3.15).

In the several experiments that we have performed using formula 3.18, the results were very satisfactory, as we will see in the next paragraph.

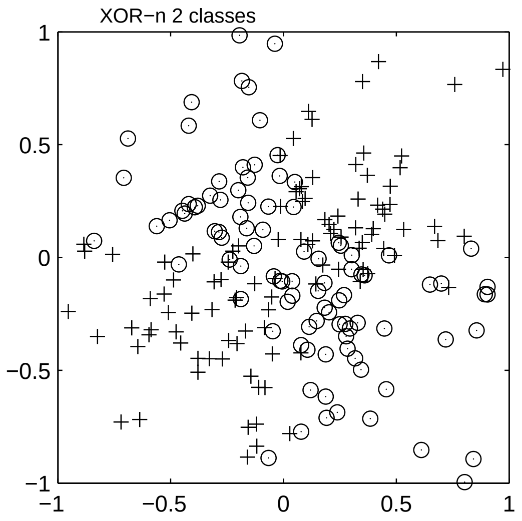
Experiments We now describe experiments using several different real data sets with different number of samples and different number of classes, namely the Ionosphere, Sonar, Wdbc, Iris, Wine and 2VowelsPB data sets.

We also produced an artificial data set and used it as a 2-class and as a 4-class classification problem in an attempt to establish the influence of the number of classes in the value of the smoothing parameter. The two versions of the artificial data set (that we call XOR-n), similar to an XOR problem, but with some noise added, are shown in Fig. 3.13.

In all experiments we used [I:n_h:O] MLP's, where I is the number of input neurons, n_h is the number of neurons in the hidden layer and O is the number of output neurons. We applied the 2-fold cross validation method using half of the data for training and half for testing. The experiments for each data set were performed varying the number of neurons in the hidden layer, the value of the smoothing parameter and using different number of epochs. The results are shown in Tables 3.6, 3.7, 3.8 and 3.9. Each result is the mean error of 20 repetitions. For each number of epochs we highlighted the 10 best results in order to get the needed guidance about the optimum value for the smoothing parameter.

In Tables 3.6 and 3.7 we show the results of the classification errors for the XOR-n data sets. Comparing the two tables, we can see that an increased number of classes demands an increased value of the smoothing parameter. In the first case, (2-class problem), the optimum value for h is about 4.0 and in the second case, (4-class problem), the optimum value for h is about 4.8.

The classification errors for the real data sets are shown in Tables 3.8, 3.9, 3.10, 3.11, 3.12, 3.13 and 3.14. For each data set we performed experiments with different number of epochs. In the results we highlighted (bold), for each number of epochs of each data set, the 10 smallest classification errors and, for each data set, we underlined the best classification results.



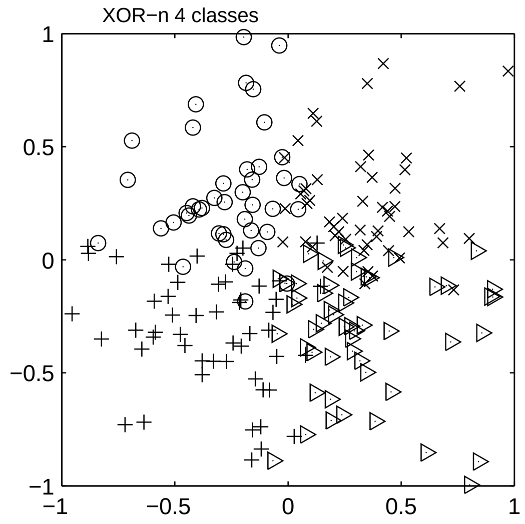


Figure 3.13: Artificial data set for the first two experiments.

Table 3.6: Errors (%) for XOR-n data set, 2 classes (resp. 80, 120 and 160 epochs from top to bottom).

	h										
n_h	2.0	2.4	2.8	3.2	3.6	4.0	4.4	4.8	5.2	5.6	6.0
2	36.50	35.43	37.90	36.28	35.93	35.40	35.33	35.63	34.45	36.35	34.20
4	28.25	30.38	27.18	26.33	26.30	25.75	27.75	24.70	28.60	25.68	27.53
6	24.40	26.43	24.45	26.78	27.10	26.83	27.25	24.98	26.75	23.75	23.70
8	**23.10**	23.93	26.65	23.88	**22.43**	24.93	**22.30**	23.73	**23.05**	24.93	**22.38**
10	**23.38**	27.00	26.38	25.65	27.08	$\underline{\textbf{21.10}}$	26.65	25.33	$\underline{\textbf{21.20}}$	25.50	25.55
12	27.30	28.15	26.48	23.68	27.20	24.80	**22.20**	**22.65**	24.68	24.10	23.60
2	35.10	35.78	34.00	33.68	33.83	34.95	33.20	33.98	32.93	33.78	33.98
4	24.30	29.68	26.25	25.05	24.65	25.18	23.38	24.10	**22.60**	23.40	25.18
6	24.43	25.08	25.00	24.80	24.53	$\underline{\textbf{21.20}}$	24.10	23.68	23.63	**22.75**	23.33
8	**22.75**	23.88	25.45	28.55	25.03	**22.88**	25.80	23.88	24.08	23.80	**22.75**
10	24.48	25.75	26.75	26.63	24.70	26.23	23.30	23.43	24.20	24.93	**22.63**
12	23.55	26.38	27.13	25.63	26.43	$\underline{\textbf{22.15}}$	24.15	**23.00**	25.80	**23.25**	24.05
2	36.28	34.68	35.10	33.65	34.90	33.78	34.03	33.68	35.10	33.03	34.70
4	27.03	26.28	27.10	26.58	$\underline{\textbf{22.00}}$	25.63	24.40	24.00	$\underline{\textbf{21.90}}$	$\underline{\textbf{21.33}}$	$\underline{\textbf{22.03}}$
6	**22.78**	24.93	24.40	24.75	26.48	$\underline{\textbf{21.28}}$	23.58	23.38	24.45	23.18	$\underline{\textbf{22.08}}$
8	23.43	24.30	27.43	25.25	24.18	24.33	25.53	**22.55**	23.25	24.33	24.33
10	23.70	26.60	25.50	26.60	24.98	25.80	23.73	**22.78**	23.38	23.60	22.95
12	**22.55**	27.05	26.55	25.83	25.00	23.48	26.73	23.70	23.00	23.60	23.33

Table 3.7: Errors (%) for XOR-n data sets, 4 classes (resp. 80, 120 and 160 epochs from top to bottom).

	h										
n_h	2.0	2.4	2.8	3.2	3.6	4.0	4.4	4.8	5.2	5.6	6.0
2	19.00	17.65	18.90	17.75	19.10	$\underline{\textbf{17.38}}$	18.18	18.68	18.83	18.68	$\underline{\textbf{17.40}}$
4	19.08	18.13	17.98	18.18	18.50	**17.73**	18.78	17.98	**17.70**	18.33	19.10
6	18.13	19.10	17.90	17.73	17.95	18.65	18.00	18.48	$\underline{\textbf{17.30}}$	$\underline{\textbf{17.65}}$	18.10
8	19.30	18.73	**17.85**	18.30	18.30	18.20	**18.13**	19.18	19.53	18.40	19.53
10	18.88	$\underline{\textbf{17.40}}$	19.15	17.75	18.53	19.13	18.08	18.50	18.70	19.15	**17.73**
12	19.10	18.83	18.23	18.58	18.30	18.38	18.70	18.25	18.70	19.13	18.45
2	18.63	20.38	18.33	18.28	18.20	18.48	17.68	17.60	$\underline{\textbf{17.38}}$	$\underline{\textbf{17.50}}$	17.85
4	19.43	19.28	18.98	18.23	**18.05**	18.23	18.18	18.60	**18.08**	18.23	18.08
6	18.63	18.55	19.30	**17.75**	18.95	18.80	18.98	18.93	17.80	$\underline{\textbf{17.68}}$	17.85
8	20.18	18.95	18.48	18.15	18.73	18.70	18.48	18.35	**17.95**	18.65	18.18
10	19.70	19.73	18.88	18.53	**17.75**	18.15	18.05	18.08	18.63	18.78	$\underline{\textbf{17.68}}$
12	20.63	20.45	18.38	18.65	18.80	**17.78**	19.15	18.50	18.78	19.15	18.55
2	18.35	18.60	17.93	18.88	18.08	$\underline{\textbf{17.33}}$	18.43	18.38	$\underline{\textbf{17.40}}$	17.60	18.18
4	19.45	18.63	19.50	19.38	**17.95**	18.85	20.08	19.13	19.68	**18.45**	18.70
6	19.80	19.00	19.75	19.68	18.90	19.03	19.28	19.30	**18.85**	**18.33**	19.45
8	19.20	19.30	19.60	19.35	18.90	**18.28**	19.60	19.05	**18.78**	19.30	19.03
10	**18.83**	19.85	19.35	19.38	19.90	19.65	19.45	**18.98**	19.98	19.58	19.38
12	19.48	19.08	19.38	19.58	19.05	19.28	19.53	19.18	19.33	19.18	19.08

Table 3.8: Errors (%) for data set Ionosphere (resp. 40, 60 and 80 epochs).

	h									
n_h	1.4	1.8	2.2	2.6	3	3.4	3.8	4.2	4.6	5
4	33.84	**12.47**	12.71	13.07	12.70	13.10	12.73	13.09	12.96	12.71
8	19.54	12.70	12.86	**12.40**	12.97	12.80	12.79	**12.39**	12.81	12.61
12	20.04	13.21	12.81	$\underline{\textbf{12.37}}$	12.74	12.80	12.70	12.67	$\underline{\textbf{12.06}}$	12.90
16	15.29	13.13	13.20	12.82	12.77	12.67	$\underline{\textbf{12.26}}$	12.73	12.80	12.70
20	16.26	12.69	13.03	12.93	12.70	12.64	**12.46**	**12.59**	**12.43**	12.69
24	17.13	12.61	13.01	13.17	13.23	12.84	$\underline{\textbf{12.26}}$	12.97	12.63	12.87
4	12.70	13.13	13.41	13.07	13.17	12.91	13.07	13.06	12.77	13.11
8	12.76	13.19	13.06	12.83	13.03	**12.50**	13.10	13.29	12.70	**12.44**
12	12.97	12.91	12.88	13.16	13.23	12.77	13.36	12.82	12.61	**12.44**
16	13.24	13.01	12.91	12.94	12.86	12.66	12.89	**12.46**	**12.44**	$\underline{\textbf{12.29}}$
20	13.54	12.80	12.80	13.16	12.84	12.94	$\underline{\textbf{12.24}}$	**12.41**	$\underline{\textbf{12.29}}$	13.16
24	13.27	13.14	12.83	12.71	13.19	12.60	12.87	**12.41**	12.96	12.87
4	13.07	13.47	14.20	13.33	13.00	13.10	12.81	13.04	12.61	13.16
8	13.02	13.28	13.14	12.83	12.71	13.00	12.62	**12.40**	**12.41**	**12.56**
12	13.00	13.40	13.40	12.94	12.70	13.11	12.88	12.64	13.29	**12.60**
16	13.09	13.18	13.00	**12.53**	$\underline{\textbf{12.33}}$	12.73	$\underline{\textbf{12.24}}$	$\underline{\textbf{12.34}}$	12.67	**12.60**
20	13.17	12.87	12.80	13.06	12.90	12.83	12.69	12.94	12.73	12.63
24	13.49	13.20	13.50	12.69	12.90	**12.59**	12.63	12.99	12.72	13.06

Table 3.9: Errors (%) for data set Sonar (resp. 50, 100 and 150 epochs).

	h										
n_h	1.0	1.3	1.6	1.9	2.2	2.5	2.8	3.1	3.4	3.7	4.0
2	49.47	48.49	24.90	24.78	23.17	24.37	23.82	24.04	24.11	24.50	24.26
4	48.63	45.77	23.65	23.27	23.68	23.05	23.70	24.06	23.82	23.82	23.15
6	49.18	45.67	23.61	22.96	23.53	**22.28**	23.61	22.98	24.18	$\underline{\textbf{21.73}}$	23.42
8	48.97	43.22	22.96	**22.40**	24.28	23.08	**22.72**	22.98	24.04	22.81	23.56
10	48.56	41.37	22.88	**22.65**	23.77	23.34	**22.76**	22.98	23.08	**22.21**	$\underline{\textbf{22.12}}$
12	50.65	43.73	23.17	23.03	23.66	24.18	**22.36**	**22.81**	23.13	22.81	22.96
2	48.36	36.71	24.64	24.02	24.16	23.99	23.51	24.64	24.06	23.56	23.58
4	49.90	35.36	23.44	24.52	24.33	**22.40**	24.47	24.04	24.35	23.58	23.77
6	49.30	37.74	23.41	23.51	23.49	24.52	23.00	**22.45**	23.15	23.37	23.13
8	49.04	35.53	24.06	22.86	22.84	24.35	22.81	23.82	$\underline{\textbf{22.12}}$	23.44	**22.40**
10	48.41	34.52	23.94	23.29	24.42	22.81	23.10	23.29	23.27	23.44	**22.67**
12	49.81	31.42	23.80	23.08	23.08	**22.74**	23.87	**22.74**	$\underline{\textbf{21.32}}$	$\underline{\textbf{21.61}}$	$\underline{\textbf{22.14}}$
2	50.82	31.63	25.53	24.23	25.31	24.06	24.33	24.98	24.42	25.41	23.73
4	50.48	28.10	24.52	24.52	24.45	24.06	23.85	23.17	23.56	23.58	23.89
6	50.00	27.81	23.92	23.08	23.27	**22.26**	23.99	23.29	$\underline{\textbf{22.14}}$	23.51	23.08
8	49.86	29.88	24.18	22.33	**22.67**	22.67	23.00	23.05	23.32	22.84	$\underline{\textbf{21.95}}$
10	50.82	26.90	25.00	24.13	22.74	23.41	22.86	**22.26**	23.17	23.27	$\underline{\textbf{22.07}}$
12	49.95	25.51	23.61	24.52	23.15	23.25	**22.67**	22.88	$\underline{\textbf{22.00}}$	$\underline{\textbf{22.07}}$	**22.60**

Table 3.10: Errors (%) for data set Wdbc (resp. 40, 60 and 80 epochs).

	h										
n_h	1.0	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0
2	50.34	8.83	**2.53**	2.77	2.83	2.84	2.91	2.67	3.28	2.89	**2.69**
4	52.31	10.37	**2.33**	**2.70**	2.75	2.87	2.77	2.76	2.88	2.85	2.91
6	51.22	19.04	**2.65**	**2.61**	2.90	**2.64**	2.91	2.92	3.31	2.79	3.09
8	49.44	19.74	**2.57**	**2.71**	2.84	**2.58**	3.07	2.84	2.97	2.94	3.19
10	46.88	21.80	3.01	2.77	2.78	2.97	2.90	2.59	3.28	2.95	2.98
2	51.54	10.71	**2.79**	2.97	3.05	2.97	**2.72**	2.94	3.07	3.06	2.98
4	51.16	11.46	**2.80**	**2.83**	3.06	3.01	3.06	3.04	**2.93**	3.02	3.14
6	49.32	11.48	3.07	**2.91**	3.06	3.06	**2.81**	3.04	3.02	3.25	3.16
8	47.58	9.74	2.95	**2.83**	2.97	3.07	2.97	**2.88**	3.06	3.07	3.17
10	45.97	19.47	**2.82**	2.96	3.05	3.08	3.23	3.24	3.49	3.28	3.37
2	48.80	4.41	**2.77**	**2.95**	**2.96**	3.26	3.13	3.06	3.03	3.15	3.30
4	46.52	**2.89**	**2.81**	3.23	3.04	**2.87**	3.11	3.26	3.08	3.05	3.49
6	49.48	5.49	3.08	3.75	3.11	3.12	3.17	3.22	3.06	3.28	3.49
8	48.60	5.83	3.00	3.06	3.08	2.99	**2.95**	**2.99**	3.21	3.20	3.35
10	48.58	6.11	**2.92**	**2.91**	3.08	3.06	3.05	3.36	3.34	3.40	3.35

Table 3.11: Errors (%) for data set Iris (resp. 40, 60 and 80 epochs).

	h										
n_h	2.0	2.4	2.8	3.2	3.6	4.0	4.4	4.8	5.2	5.6	6.0
2	7.97	7.17	6.77	7.37	4.70	5.50	4.80	5.50	7.40	5.54	5.00
4	7.20	6.63	4.43	4.87	6.67	4.30	**4.20**	**4.10**	4.67	**3.90**	4.37
6	6.33	4.60	4.77	**4.10**	4.67	4.83	4.37	4.33	4.63	**4.20**	**4.07**
8	9.20	4.63	**3.83**	4.40	**4.10**	5.50	4.73	4.57	4.97	4.47	**4.00**
10	7.27	4.24	4.70	4.40	**4.03**	4.53	4.23	4.37	4.50	4.27	4.53
2	5.07	4.80	5.87	8.40	6.37	6.60	6.60	6.37	4.60	5.40	5.70
4	5.23	**3.67**	4.67	4.10	5.00	**3.97**	4.47	**3.80**	4.23	4.53	4.03
6	5.07	4.50	4.27	**3.93**	**3.97**	**3.97**	**3.80**	4.40	4.07	4.20	4.43
8	4.40	**3.87**	4.07	4.27	**3.97**	**3.87**	4.20	4.50	4.13	4.40	4.13
10	4.90	4.30	4.13	**3.80**	4.13	4.53	4.37	4.07	4.00	4.10	4.10
2	5.13	7.83	4.60	5.27	4.57	5.57	7.30	5.17	6.10	4.70	5.00
4	4.57	**3.73**	4.50	4.40	4.07	5.73	4.23	4.53	4.53	4.30	5.00
6	4.10	4.80	**3.77**	**3.73**	4.63	**3.83**	**3.67**	4.03	4.53	4.57	4.30
8	4.23	4.23	**3.57**	3.83	4.00	**3.50**	4.30	4.20	3.90	4.10	3.83
10	4.13	4.23	4.10	3.93	4.10	**3.80**	4.57	4.00	3.87	**3.73**	**3.80**

Table 3.12: Errors (%) for data set Wine (resp. 40, 60 and 80 epochs).

	h									
n_h	1.4	1.8	2.2	2.6	3.0	3.4	3.8	4.2	4.6	5.0
4	46.41	8.54	2.28	2.30	2.47	2.33	2.64	2.14	2.56	2.05
6	34.92	2.61	2.44	2.84	2.42	2.59	2.59	2.28	2.28	$\underline{\mathbf{1.86}}$
8	29.30	2.28	2.19	2.25	2.36	2.19	2.14	2.22	2.16	2.42
10	22.89	2.28	2.56	$\underline{\mathbf{1.94}}$	**1.97**	2.05	2.11	2.11	2.39	2.16
12	17.89	2.28	2.50	2.36	2.39	2.78	2.30	2.64	**2.00**	2.22
14	18.60	2.53	2.67	2.42	2.31	2.08	2.30	$\underline{\mathbf{1.94}}$	$\underline{\mathbf{1.83}}$	$\underline{\mathbf{1.97}}$
16	17.75	2.45	2.36	2.33	2.22	$\underline{\mathbf{1.94}}$	2.33	2.42	$\underline{\mathbf{1.91}}$	**2.02**
4	6.77	**2.16**	2.95	2.28	2.42	2.50	**2.05**	2.45	2.47	2.89
6	2.25	2.64	2.28	2.39	2.39	2.22	2.39	2.36	2.31	2.53
8	3.79	2.33	2.42	2.19	2.44	2.89	2.64	$\underline{\mathbf{1.88}}$	2.78	**2.11**
10	2.30	2.61	2.56	**2.14**	2.08	2.22	$\underline{\mathbf{1.94}}$	2.75	2.39	2.36
12	2.81	2.39	2.61	2.36	2.56	2.30	2.42	2.45	**2.16**	**2.17**
14	2.61	2.50	2.59	2.33	2.28	**2.16**	2.64	**2.19**	2.28	2.39
16	2.56	2.36	2.30	2.28	2.22	2.45	2.22	2.44	2.61	2.28
4	**2.30**	2.47	2.53	2.47	2.81	**2.17**	2.81	2.78	3.01	3.06
6	2.92	2.64	2.64	2.81	2.56	**2.33**	2.59	$\underline{\mathbf{1.88}}$	**2.36**	2.84
8	2.67	2.42	2.61	2.64	2.42	2.44	2.84	2.67	2.50	2.61
10	2.56	2.75	2.73	2.78	2.84	2.42	2.70	2.36	2.95	2.44
12	2.39	2.64	2.59	2.47	2.61	**2.25**	2.78	2.87	2.39	**2.02**
14	2.67	2.95	2.28	2.64	2.87	2.47	**2.36**	**2.30**	2.89	2.42
16	2.39	2.59	2.64	2.89	**2.31**	2.42	2.75	2.53	2.87	2.70

Table 3.13: Errors (%) for data set 2VowelsPB (resp. 200, 250 and 300 epochs).

	h							
n_h	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0
2	24.09	21.25	25.49	28.72	28.64	28.94	28.77	29.35
4	16.17	11.53	9.11	8.79	11.65	9.73	11.99	11.51
6	13.26	**8.19**	**7.82**	**8.66**	8.77	9.19	11.28	11.37
8	14.46	9.29	**8.11**	**8.49**	9.18	9.34	9.36	9.05
10	14.71	8.88	**8.51**	8.85	**8.35**	9.70	9.61	**8.30**
12	14.62	**8.12**	8.92	9.31	9.88	**8.73**	9.36	9.89
2	22.29	22.89	24.94	28.46	30.83	30.30	28.75	28.29
4	15.07	**7.80**	$\underline{\mathbf{7.65}}$	10.35	12.38	13.22	10.97	10.81
6	12.76	8.50	**7.82**	**8.24**	9.61	8.77	8.90	9.60
8	10.21	8.29	8.40	**7.80**	8.53	10.58	**8.22**	10.43
10	11.65	$\underline{\mathbf{7.76}}$	$\underline{\mathbf{7.64}}$	$\underline{\mathbf{7.58}}$	**8.21**	9.08	9.89	10.15
12	11.23	8.36	9.08	8.92	8.81	9.26	9.38	8.63
2	18.73	25.80	25.71	27.02	29.65	29.03	28.63	28.31
4	12.10	8.00	8.36	9.40	10.77	10.78	10.21	11.55
6	10.63	$\underline{\mathbf{7.51}}$	8.43	8.17	8.41	9.43	8.77	$\underline{\mathbf{7.55}}$
8	10.78	**7.89**	$\underline{\mathbf{7.55}}$	**7.94**	8.70	$\underline{\mathbf{7.72}}$	9.83	8.26
10	10.68	8.39	8.08	**8.05**	9.76	8.31	9.37	8.51
12	9.03	8.25	8.17	$\underline{\mathbf{7.60}}$	$\underline{\mathbf{7.76}}$	**8.09**	8.82	8.89

Table 3.14: Errors (%) for data set Olive (resp. 140, 200 and 260 epochs).

	h										
n_h	3.0	3.4	3.8	4.2	4.6	5.0	5.4	5.8	6.2	6.6	7.0
10	6.65	6.53	6.34	6.30	6.03	6.15	6.05	6.27	5.92	6.10	5.91
15	6.07	5.69	6.37	6.00	6.14	5.68	5.72	5.84	**5.56**	5.98	**5.60**
20	5.92	6.11	5.94	5.99	**5.68**	5.70	5.75	5.74	5.76	5.74	5.70
25	6.06	6.02	6.00	5.86	$\underline{\textbf{5.29}}$	**5.64**	5.65	$\underline{\textbf{5.29}}$	5.83	**5.59**	5.78
30	5.81	5.86	5.73	5.74	**5.58**	5.93	5.70	5.85	**5.38**	5.88	**5.65**
10	6.37	5.90	5.94	6.08	6.16	5.65	5.92	5.62	6.41	6.08	5.74
15	5.89	5.46	5.48	5.70	5.77	5.37	5.71	5.40	5.57	5.39	5.58
20	5.39	5.38	5.52	5.42	5.49	$\underline{\textbf{5.26}}$	$\underline{\textbf{5.25}}$	5.49	5.38	5.58	5.74
25	5.60	5.84	5.50	**5.31**	**5.04**	5.47	5.45	5.36	$\underline{\textbf{5.19}}$	5.34	5.45
30	5.30	5.58	5.47	5.38	**5.31**	$\underline{\textbf{5.27}}$	$\underline{\textbf{5.25}}$	5.44	5.40	$\underline{\textbf{5.29}}$	$\underline{\textbf{5.28}}$
10	6.12	6.23	6.18	6.46	5.90	6.23	6.36	5.90	6.10	6.13	6.33
15	5.86	5.74	6.37	5.59	$\underline{\textbf{5.29}}$	5.88	5.48	5.76	5.85	**5.40**	5.60
20	5.70	5.69	$\underline{\textbf{5.29}}$	5.54	5.74	5.96	5.95	5.71	5.79	5.66	5.41
25	5.49	**5.35**	5.67	**5.35**	**5.37**	**5.31**	5.48	5.58	5.77	5.48	5.40
30	6.08	5.86	5.45	5.42	5.46	5.46	**5.31**	5.69	$\underline{\textbf{5.28}}$	5.80	**5.39**

In Table 3.15 we present the values of h for the minimum classification errors ("Best h" column) and the value of h representative [5] of the underlined smallest classification errors ("Suggested h" column). We present also the values proposed for each data set by formula 3.17 ("Formula 3.17" column) and formula 3.18 ("Formula 3.18" column). In both formulas the N values correspond to the number of elements of the training set (50% of the number of elements of the data set given in column "# samples").

The experiments clearly show that, for small values of h, the classification errors are significantly large and, therefore, the use of formulas like (3.17), used for density estimation, are not appropriate for the EEM algorithm. Figure 3.12 shows (crosses and open circles), the h values obtained in the experiments with all data sets (except for data set Olive). In this figure we can easily see the relation between the h values of the proposed formula 3.18 and the values obtained in the experiments. For each data set the crosses correspond to the values of the

[5]This "suggested h", h_s, was obtained by averaging the error results using the following formula: $h_s = \sum \frac{h_i}{e_i} / \sum \frac{1}{e_i}$, where e_i are the smallest classification errors ($\sim$ 10, underlined for each experiment) and h_i the respective h values.

Table 3.15: Values of h for each data set driven by the experiments and defined by formulas 3.17 and 3.18.

Data sets	*Classes*	*#samples*	*Best h*	*Suggested h*	*Formula 3.18*	*Formula 3.17*
Ionosphere	2	351	4.6	3.9	2.67	0.85
Sonar	2	208	3.4	3.7	3.47	0.92
Wdbc	2	569	1.4	1.7	2.10	0.78
XOR-n	2	200	4.0	4.7	3.54	0.93
Iris	3	150	4.0	3.6	5.00	1.05
Wine	3	178	4.6	4.2	4.59	1.02
2VowelsPB	4	608	1.8	2.2	2.87	0.93
XOR-n	4	200	5.2	5.0	5.00	1.07
Olive	9	572	4.6	5.4	4.43	1.20

"Suggested h" and the circles to the h values of the minimum classification error.

Based on the experimental evidence, we conclude that the proposed formula 3.18 for the value of the smoothing parameter, h, of the entropy estimation, yields good results when compared to existing formulas of h used for density estimation. Also, the proposed formula, contrary to the one proposed by Bowman, does not and cannot depend on s, due to the iterative nature of the EEM-VLR algorithm. We showed that it produces very good results using a set of experiments where the values proposed by our formula are much closer to the best ones found empirically.

3.3.3 The Batch Sequential Algorithm

The fact that, in the EEM algorithm, the entropy is estimated using the probability density function estimation with the Parzen window method implies the use of all available error samples to estimate its value. This fact forces the use of the batch mode in the back-propagation algorithm, limiting the use of the sequential or stochastic mode which would be, in some cases, more appropriate.

Apart from the higher complexity of the algorithm in batch mode, we know that this approach has some limitations over the sequential mode. This was the reason for us to try to combine both modes when using entropic criteria. To overcome the above mentioned limitation, we propose a new approach that combines these two modes (the batch and the sequential) trying to use their mutual advantages. The proposed Batch-Sequential algorithm combines the two methods applied in the back propagation learning algorithm: the *sequential* mode, also referenced as on-line or stochastic mode, where the update is made for each sample of the training set, and the *batch* mode, where the update is performed after the presentation of all samples of the training set. A brief reference to the possibility of combining both batch and sequential modes when training neural networks was made in [21].

We said, in Chapter 2, that the sequential mode of weight updating leads to a sample-by-sample *stochastic* search in the weight space implying that it becomes less likely for the back-propagation algorithm to be trapped in local minima [80]. However, we still need some samples to estimate the entropy, and this limits the use of the sequential mode. One of the advantages of the batch mode is that the gradient vector is estimated with more accuracy, guaranteeing the convergence to, at least, a local minima.

In order to make use of the advantages of both modes and also to speedup the algorithm, we developed a batch-sequential algorithm consisting on the splitting of the training set in several groups that are presented to the algorithm in a sequential way. In each group we apply the batch mode.

Let Ts be the training set of a given data set and Ts_j the subsets obtained by randomly dividing Ts in several groups with an equal number of samples, such as

$$|Ts| = n + \sum_{j=1}^{L} |Ts_j|, \tag{3.19}$$

where L is the number of subsets and n the remainder. This division is performed in each epoch of the learning phase. Leaving, in each epoch, some samples out of the learning process (when $n \neq 0$) is not significant because those samples will most likely be included in the next epoch. The partition of the training set in subsets reduces the probability of the algorithm getting trapped in local minima since it is performed in a random way. The subsets are sequentially presented to the learning algorithm, which applies to each one, in batch mode, the respective back-propagation and subsequent weight update. The pseudo code for the Error Entropy Minimization Batch-Sequential algorithm (EEM-BS) is presented in Table 3.16.

Table 3.16: Pseudo-code for the EEM-BS Algorithm.

```
For k:=1 to number of epochs
    Create $L$ subsets of $Ts$.
    For j:=1 to $L$
        Compute the error entropy gradient of $Ts_j$ applying formula 3.11.
        Perform weight update.
    End For
End For
```

One of the advantages in using the batch-sequential algorithm is the decreasing of the algorithm complexity. The complexity of the original EEM algorithm, due to formulas 3.10 and 3.11, is $O(|Ts|^2)$. We clearly see that, for large training sets, the algorithm is highly time consuming. With the EEM-BS algorithm the complexity is proportional to:

$$L\left(\frac{|Ts|}{L}\right)^2. \tag{3.20}$$

Therefore, the complexity ratio of both algorithms is:

$$\frac{|Ts|^2}{L(\frac{|Ts|}{L})^2} = L, \tag{3.21}$$

which means that, in terms of computational processing time, we achieve a reduction proportional to L. For a complete experiment, similar to the one presented in the next paragraph with the data set "Olive", we reduced the processing time from about 30 to 6 minutes in our machine.

The number of subsets, L, is determined by the size of the data set. If, in a given problem, the training set has a large number of data samples, we can use a higher number of subsets than if we have a small training set. We recommend the division of the training set in a number of subsets with a number of samples not less than 40, even though we sometimes got good results with less elements.

In order to perform the experiments with the batch-sequential algorithm, we tried to use the EEM algorithm with adaptive learning rate (EEM-VLR). Let us remind that EEM-VLR is based on the use of a global variable learning rate during the training phase, as a function of the error entropy value in consecutive iterations. Since in EEM-VLR we compare H_{R2} of a certain epoch with the same value in the previous one, we cannot combine it with the batch-sequential algorithm because, in each epoch, we use different sets of samples and, by this simple fact, we would have different values of H_{R2}. To overcome this limitation, and also with the goal of achieving a faster convergence, we implemented a similar process, also using variable learning rate, but this time, the variation of the learning rate is done for each neural network weight by comparing the respective gradient in consecutive iterations (EEM-BS(SA)). This approach was already used in back-propagation with MSE [177]. We also used, for the same purpose of speeding up the convergence, the combination of the batch-sequential algorithm with the resilient back-propagation [156], achieving very good results (EEM-BS(RBP)). Examples of the training phase for the three different methods, with the data set "Olive", are depicted in Fig.3.14.

Experiments In order to establish the validity of the proposed algorithm we performed several experiments, comparing the results obtained with the EEM-BS algorithm with those obtained with the simple EEM-VLR algorithm. We used the data sets Ionosphere, Olive, Wdbc and Wine.

In all experiments we used [I:n_h:O] MLP's, where I is the number of input neurons, n_h is the number of neurons in the hidden layer and O is the number of

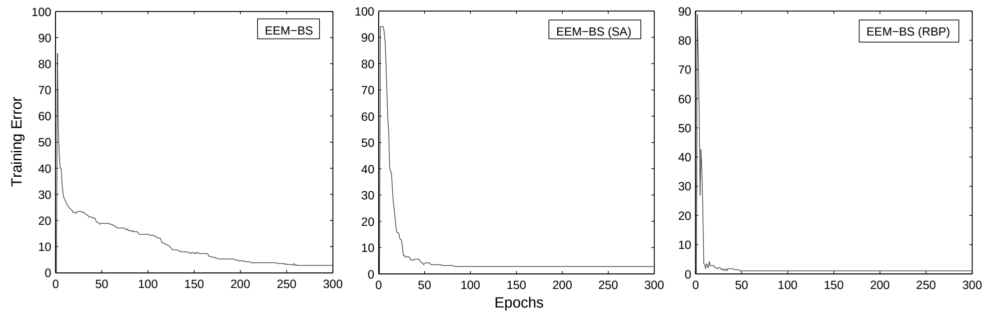


Figure 3.14: Training Error curves for the EEM-BS and the two combinations: EEM-BS(SA) and EEM-BS(RBP).

output neurons. We applied the 2-fold cross validation method. The experiments for each data set were performed varying the number of neurons in the hidden layer, the number of subsets used and the number of epochs. In Table 3.17 we present the results obtained with the EEM-BS algorithm with 4 and 8 subsets (5 and 8 for the Wdbc data set). Each result is the mean error of 20 repetitions. In Table 3.18 we only present the best results for each experiment with the EEM-BS algorithm and the comparison with the results obtained with the EEM-VLR algorithm.

The results presented in Table 3.17 show that the final errors, for each data set, do not present a significant variation according to the characteristics of the experiment (epochs, n_h). Also, the errors obtained, in each data set, with different number of subsets are very similar.

The comparison of the results obtained with both algorithms (EEM-BS and EEM-VLR), presented in Table 3.18, show that they are similar and, in some cases, the ones obtained by EEM-BS are better than those of EEM-VLR. Since the best results were obtained with different neural network complexities, we present in column *Tpe* the processing time per epoch for each algorithm. It is clear that the computational time can be reduced by a considerable factor when using the EEM-BS algorithm.

In this subsection we saw that, the combination of sequential and batch modes when using entropic criteria in the learning phase can profit from the advantages of both methods. We show, using experiments, that this is a valid approach that can be used to speed-up the training phase, maintaining a good performance, both in achieved error rate and computational time.

Table 3.17: Errors and standard deviations of the experiments performed with EEM-BS in data sets Ionosphere, Olive, Wdbc and Wine, for different values of n_h and different number of epochs and subsets (L).

Ionosphere								
	L=4				L=8			
	Epochs				Epochs			
n_h	60	80	100	120	60	80	100	120
4	12.81(1.08)	13.06(1.10)	12.49(1.52)	13.0(1.57)	12.40(0.82)	13.17(1.32)	12.29(1.16)	12.57(1.36)
6	12.64(1.13)	12.56(0.99)	12.39(0.96)	12.64(1.07)	12.81(1.01)	12.59(1.10)	12.63(1.46)	12.59(1.25)
8	13.14(1.30)	12.67(1.42)	**12.27**(1.23)	12.54(1.16)	12.52(1.03)	12.79(0.88)	**12.00**(1.09)	12.09(1.23)
10	12.53(1.40)	12.36(0.91)	12.77(1.49)	12.94(1.26)	12.64(1.18)	12.66(1.27)	12.46(1.18)	12.89(1.01)
12	12.40(1.24)	12.46(0.94)	12.67(1.46)	12.66(1.24)	13.00(1.12)	12.36(1.45)	13.09(1.42)	12.31(1.10)

Olive								
	L=4				L=8			
	Epochs				Epochs			
n_h	100	140	180	220	100	140	180	220
10	5.73(0.70)	5.65(0.73)	5.71(0.76)	5.79(0.63)	6.42(0.61)	5.67(0.68)	5.33(0.52)	5.66(0.47)
20	5.83(0.69)	5.24(0.49)	5.48(0.56)	5.42(0.67)	6.06(0.74)	5.81(0.88)	**5.24**(0.70)	5.55(0.70)
30	5.65(0.75)	**5.17**(0.51)	5.38(0.56)	5.24(0.52)	6.11(0.67)	5.62(0.67)	5.45(0.87)	5.35(0.63)
40	5.85(0.82)	5.48(0.49)	5.29(0.51)	5.57(0.57)	6.47(0.60)	5.85(0.73)	5.64(0.75)	5.54(0.69)

Wdbc								
	L=5				L=8			
	Epochs				Epochs			
n_h	40	60	80	100	40	60	80	100
4	2.63(0.47)	2.47(0.49)	2.58(0.56)	2.54(0.51)	2.58(0.40)	2.41(0.42)	2.54(0.54)	2.48(0.49)
6	2.66(0.70)	2.66(0.58)	2.60(0.45)	2.59(0.66)	2.54(0.53)	2.51(0.50)	2.47(0.55)	2.51(0.54)
8	2.51(0.52)	2.56(0.49)	2.46(0.43)	2.54(0.47)	2.44(0.58)	2.46(0.52)	2.43(0.48)	2.69(0.57)
10	2.52(0.41)	**2.31**(0.35)	2.52(0.42)	2.49(0.34)	**2.35**(0.48)	2.77(0.51)	2.46(0.37)	2.90(0.57)
12	2.39(0.48)	2.39(0.40)	2.50(0.53)	2.64(0.62)	2.47(0.49)	2.61(0.45)	2.63(0.60)	2.67(0.55)

Wine								
	L=4				L=8			
	Epochs				Epochs			
n_h	20	40	60	80	20	40	60	80
8	2.61(0.82)	2.22(0.81)	2.42(0.89)	2.33(0.88)	2.45(0.69)	2.0(0.85)	2.67(0.85)	2.67(1.25)
10	2.36(0.99)	2.61(1.17)	2.59(0.92)	2.58(1.32)	2.59(0.84)	2.16(1.27)	2.28(0.15)	2.50(0.88)
12	2.4(1.15)	2.36(1.15)	2.17(0.66)	2.19(0.73)	2.61(1.28)	2.05(0.86)	2.33(0.64)	1.94(0.76)
14	2.28(0.64)	2.08(1.01)	2.56(0.69)	2.50(0.88)	2.50(0.82)	2.16(1.11)	2.58(0.99)	2.36(1.15)
16	2.75(0.89)	2.03(0.53)	**1.88**(0.80)	2.25(0.91)	2.33(1.13)	2.42(1.15)	**1.88**(0.86)	2.16(1.05)

Table 3.18: Summary of the best results of EEM-BS and comparison with EEM-VLR (Tpe: Time per epoch $\times 10^{-3}$ sec.).

Ionosphere

Algorithm	*Error (Std)*	*L*	*n_h*	*Epochs*	*Tpe*
EEM-VLR	12.06 (1.11)	-	12	40	16.7
EEM-BS	12.27 (1.23)	4	8	100	6.4
EEM-BS	12.00 (1.09)	8	8	100	4.8

Olive

Algorithm	*Error (Std)*	*L*	*n_h*	*Epochs*	*Tpe*
EEM-VLR	5.04 (0.53)	-	25	200	77.7
EEM-BS	5.17 (0.51)	4	30	140	17.6
EEM-BS	5.24 (0.70)	8	20	180	12.8

Wdbc

Algorithm	*Error (Std)*	*L*	*n_h*	*Epochs*	*Tpe*
EEM-VLR	2.33 (0.37)	-	4	40	38.7
EEM-BS	2.31 (0.35)	5	10	60	13.6
EEM-BS	2.35 (0.48)	8	10	40	9.6

Wine

Algorithm	*Error (Std)*	*L*	*n_h*	*Epochs*	*Tpe*
EEM-VLR	1.83 (0.83)	-	14	40	5.8
EEM-BS	1.88 (0.80)	4	16	60	3.2
EEM-BS	1.88 (0.86)	8	16	60	2.5

Chapter 4

Clustering with Entropy

In the previous chapter we have applied entropic concepts to neural network supervised classification. In this chapter we will focus on unsupervised data classification, i.e. data clustering. The motivation for the new clustering algorithm presented here came from our goal to perform an entropic task decomposition for modular neural networks (this subject will be discussed in the following chapter).

Our new clustering algorithm [168] is a hierarchical algorithm, a stepwise clustering method, usually based on dissimilarity measures between objects or sets of objects from a given data set, but now based on a new entropic dissimilarity measure [1]. The most common dissimilarity measures are distance measures. The derived proximity matrices can be used to build graphs, which provide the basic structure for some clustering methods. We present in this chapter a new proximity matrix based on the new entropic dissimilarity measure and also a new clustering algorithm that builds layers of subgraphs based on this matrix, and uses them and a hierarchical agglomerative clustering technique to form the clusters. Our approach capitalizes on both a graph structure and a hierarchical construction. Moreover, by using entropy as a proximity measure we are able,

[1]In our proposed algorithm we use Rényi's quadratic entropy because of its simplicity; however, one could use other entropic measures as well.

with no assumption about the cluster shapes, to capture the local structure of the data, forcing the clustering method to reflect this structure. We present several experiments performed on artificial and real data sets that provide evidence of the superior performance of this new algorithm when compared with competing ones.

Some examples of the application of entropy and information-theoretic concepts in clustering are the minimum entropic clustering [121], entropic spanning graphs clustering [81] or entropic subspace clustering [29]. In some works the entropic concepts are usually related to measures similar to the Kullback-Leibler divergence. In some recent works several authors used entropy as a measure of proximity or interrelation between clusters. Examples of these algorithms are those proposed by Jenssen [97] or Gokcay [63], that use a so-called Between-Cluster Entropy, and the one proposed by Lee [119], [120] that uses the Within-Cluster Association. Despite the good results in several data sets, these algorithms are heavily time consuming and they start by selecting some random seeds as first clusters which may produce very different results in the final cluster solution depending on the number of seeds chosen and their "position". If none of the elements of a real cluster is selected as seed, those elements probably will be included in other built clusters. These algorithms usually give good results for compact and well separated clusters.

4.1 What is Clustering?

Clustering deals with the process of finding possible different groups in a given set, based on similarities or differences among their objects. This simple definition does not convey the richness of such a wide area of research. What are the similarities and what are the differences? How do the groups differ? How can we find them? These are examples of some basic questions, none with an unique answer. There is a wide variety of techniques to do clustering. Results are not unique and they always depend on the purpose of the clustering. The same data can be clustered with different acceptable solutions. Hierarchical clustering, for example, gives several solutions depending on the tree level chosen for the final solution.

There are algorithms based on similarity or dissimilarity measures between the objects of a set, like sequential and hierarchical algorithms; others, are based on the principle of function approximation, like fuzzy clustering or density based algorithms; yet others, are based on graph theory or competitive learning. In this paper we combine hierarchical and graph approaches and present a new clustering algorithm based on a new proximity matrix that is built with an entropic measure. With this measure, connections between objects are sensitive to the local structure of the data, achieving clusters that reflect that same structure.

In the following subsections we present some basic concepts and notation that serve as the basis to present our algorithm and we make an overview of some of the most popular clustering algorithms.

4.1.1 Proximity Measures

Let X be the data set, $X = \{\mathbf{x}_i\}$, $i = 1, 2, ..., N$, where N is the number of objects and $\mathbf{x}_i$ an l-dimensional vector representing each object. We define S, an s-clustering of X, as a partition of X into s subsets $C_1, C_2, ..., C_s$, obeying the conditions: $C_i \neq \oslash, i = 1, ..., s$; $\cup_{i=1}^{s} C_i = X$ and $C_i \cap C_j = \oslash, i \neq j, i, j = 1, ..., s$. Each vector (point), given these conditions, belongs to a single subset (cluster). Our proposed algorithm uses this so called *hard clustering.* (There are algorithms, like those based on fuzzy theory, in which a point has degrees of membership for each cluster.) Points belonging to the same cluster have a higher degree of similarity with each other than with any other point of the other clusters. This degree of similarity is usually defined using similarity (or dissimilarity) measures.

The most common dissimilarity measure between two real-valued vectors $\mathbf{x}$

and $\mathbf{y}$, is the weighted l_p metric,

$$d_p(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{l} w_i |x_i - y_i|^p \right)^{\frac{1}{p}}, \tag{4.1}$$

where x_i and y_i are the ith coordinates of $\mathbf{x}$ and $\mathbf{y}$, $i = 1, ..., l$, and $w_i \geq 0$ is the ith weight coefficient. The unweighted ($\mathbf{w} = 1$) l_p metric is also known as Minkowski distance of order p ($p \geq 1$). Examples of this distance are the well-known Euclidian distance, obtained by setting $p = 2$, the Manhattan distance, $p = 1$, and the l_∞ or Chebyshev distance.

4.1.2 Overview of Clustering Algorithms

Probably the most used clustering algorithms are the hierarchical, agglomerative algorithms. They create, by definition, a hierarchy of clusters from the data set. Hierarchical clustering is widely used in biology, medicine, and also in computer science and engineering. (For an overview on clustering techniques and applications see [20, 93–95]). Hierarchical agglomerative algorithms start by assigning each point to a single cluster and then, usually based on dissimilarity measures, proceed to merge small clusters into larger ones in a stepwise manner. The process ends when all the points in the data set are members of a single cluster. The resulting hierarchical tree defines the clustering levels. Examples of hierarchical clustering algorithms are CURE [68] and ROCK [69] developed by the same researchers, AGNES [110], BIRCH [206], [207] and Chameleon [108].

The merging phase of the agglomerative algorithms differs in the sense that, depending on the measures used to compute the similarity or dissimilarity between clusters, different merge results can be obtained. The most common methods to perform the merging phase are:

- Single Link Method: the dissimilarity between two clusters is measured by the distance between the two closest vectors.

- Complete Link Method: the dissimilarity between two clusters is measured by the distance between the two most distant vectors.

- Centroid Method: the dissimilarity between two clusters is measured by the distance between their centroids (usually the mean vectors).
- Ward's Method: two clusters are merged if the sum of all the distances between the resulting centroid and the joined vectors is the smallest one.

The Single Link method usually creates elongated clusters and the Complete Link usually results in more compact clusters. The Centroid method acts in a midway basis, yielding clusters somewhere between the two previous ones. Ward's method is considered very effective in producing balanced clusters; however, it has several problems dealing with outliers and elongated clusters. In [102] one can find a probabilistic interpretation of these classical agglomerative methods.

Another type of algorithms are the ones based on graphs and graph theory. A graph is defined as an ordered pair $G = (V, E)$, where $V = \{v_i\}, i = 1, ..., N$ is a set of vertices and E is a set of edges connecting pairs of vertices. An edge connecting v_i and v_j is denoted by e_{ij}. One can have directed or undirected graphs depending on whether or not the order of v_i and v_j is important. Using directed graphs originates double edges between a pair of vertices, one in each direction. Unweighted graphs are those where there is no cost associated with each edge. If there is a path in G where the first and last vertices coincide then this path is called a loop or circle. A subgraph $G' = (V', E')$ of G is a graph with $V' \subseteq V$ and $E' \subseteq E$, where edges of E' connects pairs of vertices from V'. A similarity graph is a graph based on the similarity matrix of a specific data set.

Clustering algorithms based on graph theory are usually divisive algorithms, meaning that they start with a single highly connected graph (that corresponds to a single cluster) that is then splited using consecutive cuts. A cut in a graph corresponds to the removal of a set of edges that disconnects the graph. A minimum cut (min-cut) is the removal of the smallest number of edges that produces a cut. The result of a cut in the graph causes the splitting of one cluster into, at least, two clusters. An example of a min-cut clustering algorithm can be found in [99]. Clustering algorithms based on graph theory have existed since the early 1970's. They use the high connectivity in similarity graphs to perform clustering ([130], [131]). More recent works such as [75], [76] and [196] also perform clustering using highly connected graphs and subsequent partitions by edge cutting to obtain subgraphs. Chameleon [108], mentioned earlier as a hierarchical agglomerative algorithm, also uses a graph-theoretic approach. It starts by constructing a graph, based on k-nearest neighbors, then it performs the partition of the graph into several clusters (using the hMetis [109] algorithm) such that it minimizes the edge cut. After finding initial clusters, it repeatedly merges these small clusters, using relative cluster interconnectivity and closeness measures.

Graph cutting is also used in spectral clustering, commonly applied in image segmentation and, more recently, in web and document clustering and bioinformatics. The rationale of spectral clustering is to use the special properties of the eigenvectors of a Laplacian matrix as the basis to perform clustering. Fidler [52] was one of the first to show the application of eigenvectors to graph partitioning. The Laplacian matrix is based on an affinity matrix, A, built with a similarity measure. The most common similarity measure used in spectral clustering is $A_{ij} = \exp\left(-d_{ij}^2/2\sigma^2\right)$, where d_{ij} is the Euclidian distance between vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ and σ is a scaling parameter. With matrix A, the Laplacian matrix L is computed as $L = D - A$, where D is the diagonal matrix whose elements are the sums of all row elements of A.

There are several spectral clustering algorithms that differ in the way they use the eigenvectors in order to perform clustering. Some researchers use the eigenvectors of the "normalized" Laplacian matrix [31] (or a similar one), in order to perform the cutting usually using the second smallest eigenvector [175], [103], [39]. Others, use the highest eigenvectors as input to other clustering algorithm [141], [133]. One of the advantages of this last approach is that, by using more than one eigenvector, enough information can be provided to obtain more than two clusters as opposed to cutting strategies where clustering must be performed recursively to obtain more than two clusters. A comparison of several spectral clustering algorithms can be found in [187].

The practical problems encountered with graph-cutting algorithms are basically related to the belief that the subgraphs produced by cutting are always related to real clusters. This assumption is frequently true with well separated compact clusters; however, in data sets with, for example, elongated clusters, this may not occur. Also, if we use weighted graphs, the choice of the threshold to perform graph partition can produce very different clustering solutions.

Other clustering algorithms use the existence of different density regions of the data to perform clustering. One of the density based clustering algorithms, apart from the well-known DBScan [48], is the Mean Shift algorithm. Mean Shift was introduced by Fukunaga [58], rediscovered in [30] and also studied in more detail by Comaniciu [32], [33], with applications to image segmentation. The original algorithm, with a flat kernel, works this way: in each iteration, for each point P, the cluster center is obtained by repeatedly centering the kernel (originally centered in P) by shifting it in the direction of the mean of the set of points inside the same kernel. The process is similar if we use a Gaussian kernel. The mean shift vector is aligned with the local gradient estimate and defines a path leading to a stationary point in the estimated density [33]. This algorithm seeks modes in the sample density estimation and so is considered to be a gradient mapping algorithm [30]. Mean Shift has some very good results in image segmentation and computer vision applications but, like other density based algorithms, it builds clusters with the assumption that each of them is related to a mode of the density estimation. For problems like the one depicted in Fig. 4.1a, with clusters of different densities very close to each other, this kind of algorithm usually has difficulties in performing the right partition because it finds only one mode in the density function. (If we use a smaller smoothing parameter it will find several local modes in the low density region). This behavior is also observable in data sets like the double spiral data set depicted in Fig. 4.10.

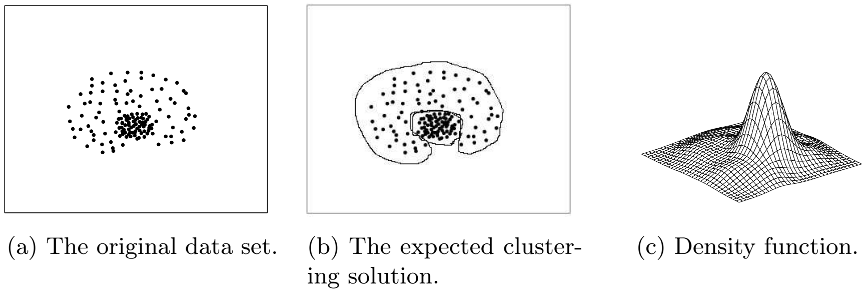

(a) The original data set. (b) The expected clustering solution. (c) Density function.

Figure 4.1: An example of a data set difficult to cluster using density based clustering algorithms like Mean Shift.

Another example of a clustering algorithm is the path-based pairwise clustering algorithm [53, 54]. This clustering method also groups objects according to their connectivity. It uses a pairwise clustering cost function with a dissimilarity measure that emphasizes connectedness in feature space to deal with cluster compactness. This simple approach gives good results with compact clusters. To deal with structured clusters a new objective function, containing the same properties of the pairwise cost function, is used. This new objective function is based on the *effective dissimilarity*, the length of the minimal connecting path between two objects, and is the basis for the path-based clustering. Some of the applications of this clustering algorithm are edge detection and texture image segmentation.

4.2 The Clustering Algorithm Components

One of the main concerns when we started searching for an efficient clustering algorithm was to find an extremely simple idea, based on very simple principles, that didn't need complex measures of intra- or inter-cluster association. Keeping this in mind, we performed clustering tests involving several types of individuals (including children) in order to grasp the mental process of data clustering. The results and analysis of these tests can be found in Appendix B. One of the most important conclusions from our tests is that human clustering exhibits some balance between the importance given to local (e.g., connectedness) and global (e.g., structuring direction) features of the data, a fact that we tried to reflect in our algorithm. The tests also provided majority choices of clustering solutions against which one can compare the clustering algorithms. Some of the experiments performed with these data sets are presented later.

In the following we introduce two new clustering algorithm components: a new proximity matrix and a new clustering process. We first present the new entropic dissimilarity measure and, based on that, the computing procedure of a layered entropic proximity matrix; afterwards we present the LEGClust (Layered Entropic subGraphs Clustering) algorithm.

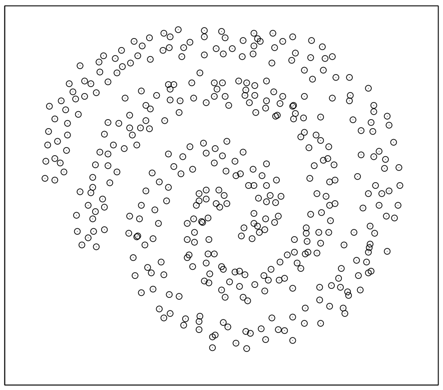
4.2.1 The Entropic Proximity Matrix

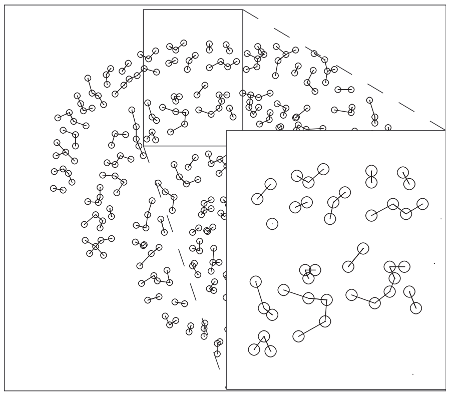
Given a set of vectors $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, .., \mathbf{x}_N\}$, $\mathbf{x_i} \in \mathbb{R}^m$, corresponding to a set of objects, each element of the dissimilarity matrix A, $A \in \mathbb{R}^{N\times N}$, is computed using a dissimilarity measure $A_{i,j} = d(\mathbf{x}_i, \mathbf{x}_j)$. Using this dissimilarity matrix one can build a proximity matrix, L, where each ith line represents the data set points, each jth column the proximity order (1st column=closest point ... last column=farthest point) and each element the point reference that, relative to row point i, is in the jth proximity position. An example of a proximity matrix, is shown in Table 4.5 (to be described in detail later on). The points referenced in the first column (L1) of the proximity matrix are those that have the smallest dissimilarity value relative to the corresponding row elements.

Each column of the proximity matrix is considered one layer of connections. We can use this proximity matrix to build subgraphs for each layer, where each edge is the connection between a point and the corresponding point of that layer.

If we use a proximity matrix based on a dissimilarity matrix built with the

Euclidian distance to connect each point with its corresponding L1 point (first layer) we get a subgraph similar to the one presented in Fig. 4.2b for the data set of Fig. 4.2a. We call the clusters formed with this first layer connections the elementary clusters. Each of these resulting elementary clusters (not considering directed edges) is a Minimum Spanning Tree.


(a) Spiral data set.


(b) Connections based on Euclidian distance.

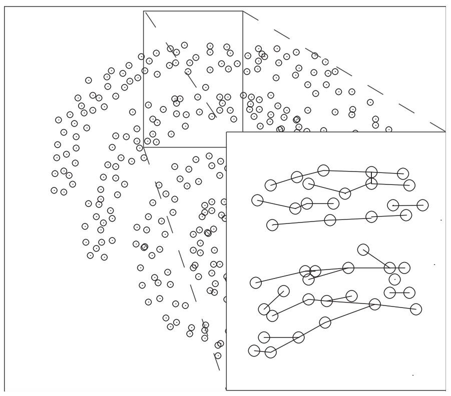

(c) "Ideal" connections.

Figure 4.2: Connections of the first layer using Euclidian distance and the "ideal" connections for the spiral data set.

As we can see from Fig. 4.2b, these connections have no relation with the structure of the given data set. In Fig. 4.2c we present what one should expect to be the "ideal" connections. These ideal connections should, in our judgement, reflect the local structuring direction of the data. However, using classical distance measures, we are not able to achieve this behavior. As we will see bellow, entropy will allow us to do it. The main idea behind the entropic dissimilarity measure is to make the connections follow the local structure of the data set, where the meaning of "local structure" will be clarified later. From now on we will take "local structure" or "local structuring" in an intuitive common-sense basis, as a designation of a prevailing direction in the data. This concept can be applied to data sets with any number of dimensions.

Let us consider the set of points depicted in Fig. 4.3. These points are in a square grid except for points P and U. For simplicity we use a two-dimensional data set, but the analysis is valid for higher dimensions. Let us denote:

- $K = \{k_i\}$, $i = 1, 2, .., M$, the set of the M nearest neighbors of P;
- d_{ij}, the difference vector between points k_i and k_j, $i, j = 1, 2, .., M$, $i \neq j$, that we will call the *connecting vector* between those points;
- p^i, the difference vector between point P and each of the M-nearest neighbors k_i.

We wish to find the connection between P and one of its neighbors that best reflects the local structure. Without making any computation and just by "looking" at the points we can say, despite the fact that the shortest connection is p_1, that the ideal candidates for "best connection" are those connecting P with Q or with R because they are the ones that best reflect the structuring direction of the data points.

Let us represent all d_{ij} connecting vectors translated to a common origin as shown in Fig. 4.4a. We will call this an *M-neighborhood vector field.* An M-neighborhood vector field can be interpreted as a probability density function in correspondence with the two-dimensional histogram shown in Fig. 4.4b, where in each bin we plot the number of occurrences of d_{ij} vector ends. This histogram estimates the probability density function of d_{ij} connections. It can be inter-

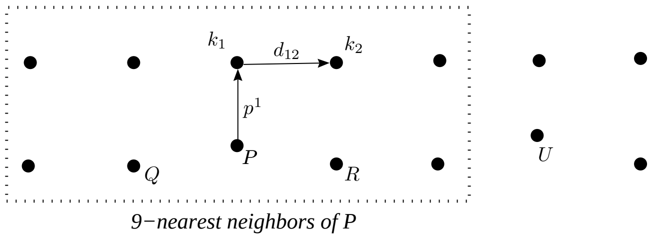


Figure 4.3: A simple example with the considered M-nearest neighbors of point P, $M = 9$; the M-neighborhood of P corresponds to the dotted region.

preted as a Parzen window estimate of the pdf using a rectangular kernel. The main elongation direction of the pdf defines our "structuring direction".

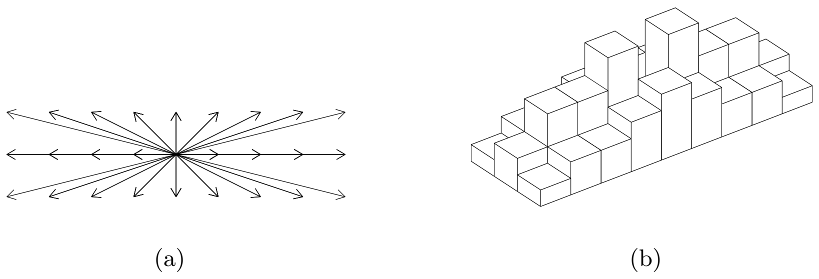


Figure 4.4: The M-neighborhood vector field of point P (a) and the histogram representation of the probability density function (b).

The probability density function associated with point P, reflects, in this case, an horizontal M-neighborhood structure and, therefore, we expect to choose an "ideal" connection for P that follows this horizontal direction. Although the direction is an important factor we should also consider the size of the connections and avoid the selection of connections between points far apart. Taking this into consideration, we can also see that in terms of the probability density function, the small connecting vectors are the most probable ones.

Now, since we want to choose a connection for point P based on ranking all possible connections, we have to compare all the probability density functions resulting from adding each connection p^i to the set of connection of the

M-neighborhood vector field. One of the measures that compares probability density functions is an entropic measure and we will use it to rank all the possible connections p^i. Basically, what we are going to do, is to rank connections p^i according to the variation introduced by each one in the probability density function. The connection that introduces less disorder into the system, that least increases the entropy of the system, will be top ranked as the *stronger* connection, followed by the other $M-1$ connections in decreasing order.

Let $D = \{d_{ij}\}$, $i, j = 1, 2, .., M$, $i \neq j$. Let $H(D, p^i)$ be the entropy associated with connection p^i, the entropy of the set of all connections d_{ij} plus connection p^i, such that

$$H(D, p^i) = H(\{d_{ij}\} \cup \{p^i\}), i = 1, 2, .., M. \tag{4.2}$$

This entropy is our dissimilarity measure. We compute, for each point, the M possible entropies and build an entropic dissimilarity matrix and the corresponding entropic proximity matrix (examples are shown in Tables 4.4 and 4.5). The elements of the first column of the proximity matrix are those corresponding to the points having the smallest entropic dissimilarity value (strongest entropic connection), followed by those in the subsequent layers in decreasing order.

We show, in Table 4.1, the dissimilarity and proximity values for point P and their neighbors depicted in Fig. 4.3. We use Rényi's quadratic entropy computed as explained in section 2.2.2. The points of Fig. 4.3 are referenced left to right and top to bottom as 1 to 14.

In Fig. 4.5 we show the first layer connections, where we can see the difference between using a dissimilarity matrix based on classic distance measures such as the Euclidian distance (Fig. 4.5a) and a dissimilarity matrix based on our entropic measure (Fig. 4.5b).

The connections derived by the first layer, when using the entropic measure, clearly follow an horizontal line and, despite the fact that point k_1 is the closest one to P in the Euclidian sense, the stronger connection for point P is the

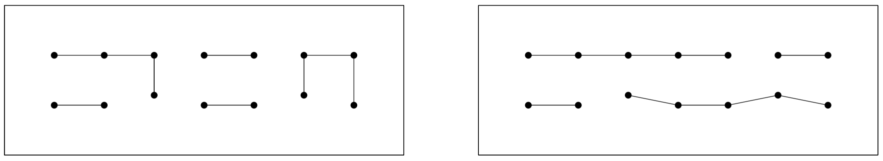
Table 4.1: Entropic dissimilarities and proximities relative to point P (10).

Point	1	2	3	4	5	6	7	8	9	10	11	12	13	14
10	-2.42	-2.67	-2.69	-2.67	-2.42			-2.87	-3.12		-3.12	-2.87		

(a) Entropic dissimilarities.

	Layers								
Point	L1	L2	L3	L4	L5	L6	L7	L8	L9
10	11	9	12	8	3	4	2	5	1

(b) Entropic proximities.



(a) (b)

Figure 4.5: Difference on Elementary Clusters using a dissimilarity matrix based on Euclidian distance (a) and on our Entropic measure (b).

connection between P and R, as desired. This different behavior can also be seen in the spiral data set depicted in Fig. 4.6. The connections that produce the elementary, first layer, clusters are clearly following the "structuring direction" of the data. We obtain the same behavior for the connections of all the layers favoring the union of those clusters that follow the structure of the data.

The pseudo-code to compute the entropic proximity matrix is presented in Table 4.2.

The process just described is different from the apparently similar process of ranking the connections p^i according to the value of the probability density

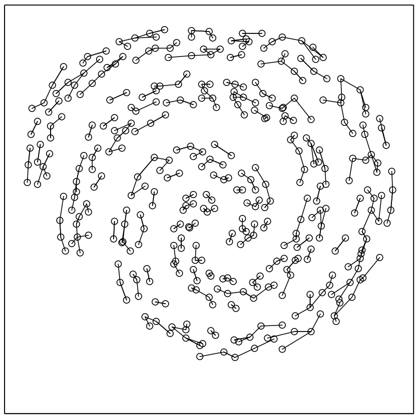


Figure 4.6: The first layer connections following the structure of the data set when using an entropic proximity matrix.

Table 4.2: Pseudo-code for computing the entropic proximity matrix.

```
For $i = 1$ to $N$ (number of objects)
    For $j = 1$ to $M$ (number of nearest neighbors)
        Compute $H(D, p^j) = H(\{D\} \cup \{p^j\})$.
    end $j$
end $i$
Build the $(N \times M)$ entropic proximity matrix.
```

function derived from the M-neighborhood vector field. In Fig. 4.7 we show the estimated probability density function and the points corresponding to the p^i connections. We can see that, even in this simple example, a difference exists in the ranking of the connections (fifth element).

4.2.2 The Clustering Process

We could use the new entropic proximity matrix with an existing clustering algorithm to cluster the data. However, the potentialities of the new proximity matrix can be exploited with a new hierarchical agglomerative algorithm that we propose and call LEGClust (Layered Entropic subGraph Clustering algorithm). The basic structure used in this new clustering algorithm is the unweighted

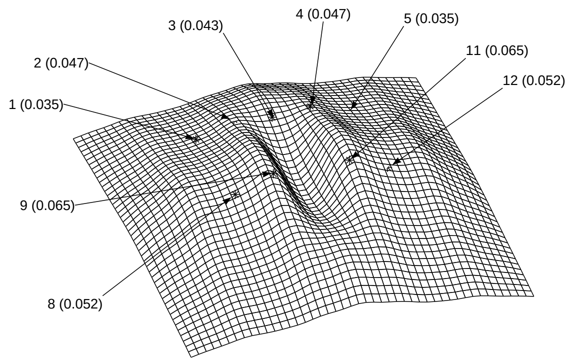


Figure 4.7: The probability density function of the M-neighborhood vector field and the points corresponding to the p^i connections. The labels indicate the element number and the pdf value.

subgraph. More specifically, we use directed, maximally connected, unweighted subgraphs, built with the information provided by the entropic proximity matrix (EPM). Each subgraph is built by connecting each point with the corresponding point of each layer (column) of the EPM. An example of such a subgraph was already shown in Fig. 4.6. The clusters are built hierarchically by joining together the clusters that correspond to the layer subgraphs.

We start by presenting, in Table 4.3, the pseudo-code of LEGClust Algorithm.

Table 4.3: Pseudo-code for the LEGClust Algorithm.

```
Compute the entropic proximity matrix. (Table 4.2)
Form the elementary clusters using the first layer.
Define $k$ - the minimum number of connections.
While number-of-clusters> 1 do
    Go to next layer (L).
    Join each cluster with the one having the highest number of connections
    with it ($\geq k$) in L.
End While
```

To illustrate the procedure applied in the clustering process we use a simple two dimensional data set example (Fig. 4.8a). This data set consists of 16 points apparently constituting 2 clusters with 10 and 6 points each. Since the number of clusters in a data set is highly subjective, the assumption that it has a specific number of clusters is always affected by the knowledge about the problem.

In Tables 4.4 and 4.5 we present the EPM built from the entropic dissimilarity matrix.

Table 4.4: The dissimilarity matrix for Fig. 4.8 data set.

Points	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
1	-	5.64	-	6.36	5.66	6.32	-	-	-	5.77	-	-	-	-	-	-
2	6.00	-	6.03	6.26	6.40	-	-	-	-	6.12	-	-	-	-	-	-
3	-	6.19	-	6.61	-	-	-	-	-	7.03	6.76	6.48	-	-	-	-
4	6.48	6.50	6.58	-	-	-	-	-	-	6.36	6.47	-	-	-	-	-
5	6.10	-	-	-	-	6.24	-	6.16	6.09	6.18	-	-	-	-	-	-
6	-	-	-	6.05	6.21	-	-	6.03	6.14	6.11	-	-	-	-	-	-
7	-	-	-	-	5.61	6.06	-	5.67	5.28	6.84	-	-	-	-	-	-
8	-	-	-	-	6.09	5.54	5.82	-	5.89	6.27	-	-	-	-	-	-
9	-	-	-	-	5.78	5.98	5.79	6.03	-	5.99	-	-	-	-	-	-
10	6.06	5.98	-	6.05	6.00	5.98	-	-	-	-	-	-	-	-	-	-
11	-	-	-	-	-	-	-	-	-	-	-	3.74	4.41	4.73	3.93	3.86
12	-	-	-	-	-	-	-	-	-	-	3.86	-	3.88	4.36	4.60	3.95
13	-	-	-	-	-	-	-	-	-	-	4.39	3.78	-	3.79	4.75	3.87
14	-	-	-	-	-	-	-	-	-	-	4.71	4.36	3.80	-	3.93	3.86
15	-	-	-	-	-	-	-	-	-	-	3.85	4.81	5.01	3.82	-	3.71
16	-	-	-	-	-	-	-	-	-	-	4.19	4.18	4.18	4.18	4.18	-

The EPM defines the connections between each point and those points in each layer: point 1 is connected with point 2 in the first layer, with point 5 in the second layer and with point 10 in the third layer and so on (see Table 4.5).

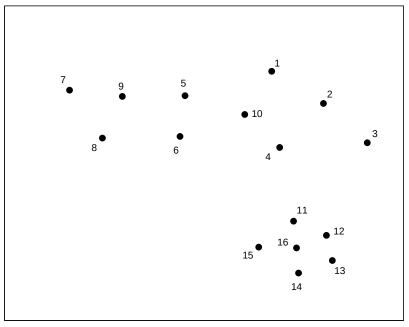
Table 4.5: The proximity matrix for Fig. 4.8 data set.

	Layers				
Points	**L1**	**L2**	**L3**	**L4**	**L5**
1	2	5	10	6	4
2	1	3	10	4	5
3	2	4	11	1	10
4	10	3	6	2	11
5	9	1	8	10	6
6	8	4	10	9	5
7	9	5	8	6	10
8	6	7	9	5	10
9	5	7	10	6	8
10	6	2	5	4	1
11	12	16	15	13	14
12	11	13	16	14	15
13	12	14	16	11	15
14	13	16	15	12	11
15	16	14	11	12	13
16	14	11	13	12	15

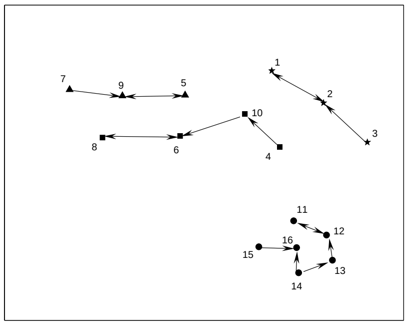
We start the process by defining the elementary clusters. These clusters are built by connecting, with an oriented edge, each point with the corresponding point of the first layer (Fig. 4.8b). There are 4 elementary clusters in our simple example.

In the second step of the algorithm we connect, with an oriented edge, each point with the corresponding point of the second layer (Fig. 4.8c).

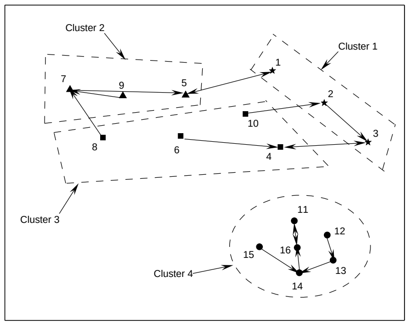
In order to build the second step clusters we apply a rule based on the number of connections to join each pair of clusters. We can use the simple rules of: a) joining each cluster with the ones having at least k connections with it; b) joining



(a) The data points.



(b) First layer connections and resulting elementary clusters.



(c) The 4 elementary clusters and the second layer connections.

Figure 4.8: The clustering process in a simple two dimensional data set.

each cluster with the one having the highest number of connections with it, not less than a predefined k. In the performed experiments this second rule proved to be more reliable, and the resulting clusters were usually "better" than using the first rule. The parameter k must be greater than 1 in order to avoid outliers and noise in the clusters. In our simple example we chose to join the clusters with the maximum number of connections larger than 2 ($k > 2$). In the second step we form 3 clusters by joining cluster 1 and 3 with 3 edges connecting them (note that the edge connecting points 3 and 4 is a double connection). The process is repeated and the algorithm stops when only one cluster is present or when we get the same number of clusters in consecutive steps. The resulting number of clusters for this simple example was 4-3-2-2-2. As we can see, the number of clusters in steps 3 and 4 is the same (2); therefore, we consider it to be the acceptable number of clusters.

4.2.3 Parameters involved in the clustering process

4.2.3.1 Number of nearest neighbors

The first parameter that one must choose in LEGClust is the number of nearest neighbors (M). We do not have a specific rule for this. However, one should not choose a very small value because a minimum number of steps in the algorithm is needed in order to guarantee reaching a solution. Choosing a relatively high value for M is also not a good alternative because one loses information about the local structure, which is the main focus of the algorithm.

Based on the large amount of experiments performed with the LEGClust algorithm on several data sets, we came to a rule of thumb of using M values not higher than 10% of the data set size. Note that, since the entropy computation for the whole data set has complexity $O\left(N\left(\binom{M}{2}+1\right)^2\right)$, the value of M has a large influence on the computational time. Hence, for large data sets a smaller M is recommended, down to 2% of the data size. For image segmentation M can be reduced to less than 1% due to the nature of image characteristics (elements are much closer to each other than in other classification problems).

4.2.3.2 The smoothing parameter

As we said earlier, the h parameter is very important when computing the entropy. In other works that also use the Rényi's quadratic entropy to perform clustering it is assumed that the smoothing parameter is experimentally selected and that it must be fine tuned to achieve acceptable results [63,97]. A comparison between formulas 2.60 and 2.61 was also performed in [96]. In this work they use formula 2.60 to estimate the optimal one-dimensional kernel size for each dimension of the data, and use the smallest value as the smoothing parameter.

In Section 3.3.2.1 we have proposed the formula $h_{op} = 25\sqrt{m/N}$ and have shown experimentally that higher values of h than those given by formula 2.61 produce better results in neural network classification using error entropy minimization as a cost function. Following the same approach, we propose a formula similar to formula 2.61, but with the introduction of the mean standard deviation:

$$h_{op} = 2\, s^* \left(\frac{4}{(m+2)N} \right)^{\frac{1}{m+4}} , \tag{4.3}$$

where s^* is the mean value of the sample standard deviations for each dimension. All experiments of LEGClust were performed using formula 4.3.

Although the value of the smoothing parameter is important, it is not crucial in order to obtain good results. As we increase the h value, the kernel becomes smoother and the entropic proximity matrix becomes similar to the Euclidian distance proximity matrix. Extremely small values of h will produce undesirable behaviors because the entropy will have high variability. Using h values in a small interval, near the optimal value, does not affect the final clustering results (e.g., we used in the spiral data set – Fig. 4.2 – values between 0.05 and 0.5 without changing the final result).

4.2.3.3 Minimum number of connections

The third parameter that must be chosen in the LEGClust algorithm, is the value of k, the minimum number of connections to join clusters in consecutive steps of the algorithm. As mentioned earlier, one should not use $k = 1$ to avoid outliers and noise, especially if they are located between clusters. In our experiments we obtained good results using either $k = 2$ or $k = 3$. If the elementary clusters have a small number of points, we do not recommend higher values for k because it can cause the impossibility of joining clusters due to lack of a sufficient number of connections among them.

4.2.4 Algorithm Optimization

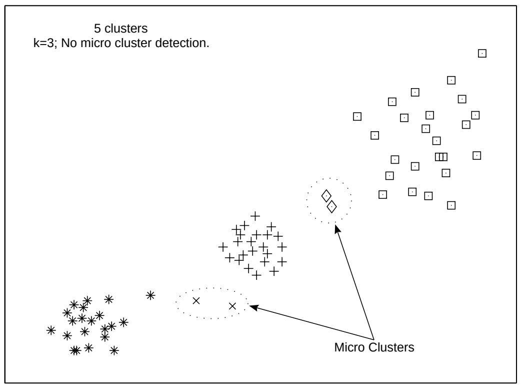
In order to optimize the LEGClust algorithm, we have introduced some extra refinements that can produce better final clustering solutions. One of them, related with the problem of outliers or noise, is the presence of "micro clusters". Micro clusters are clusters with a very small number of elements: in a small data set these could consist of 2 or 3 elements. These micro clusters can appear if, for example, there is a very small number of elements in a isolated region (Fig. 4.9a).

What should we do with these micro clusters? There are several solutions:

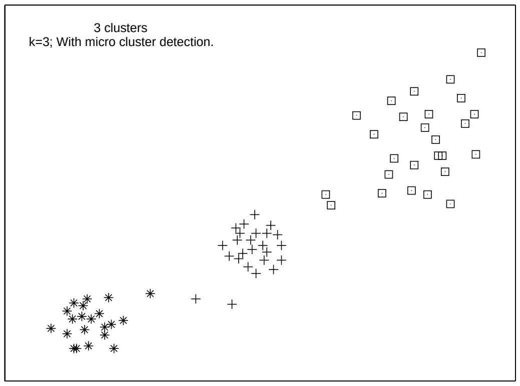
- At each step of the algorithm we can join each micro cluster to the cluster having the highest number of connections with it. This must be done with caution because it can, eventually, lead to the union of two normal clusters into one single cluster. However, in most cases, this approach produces good results. An example of it is shown in Fig. 4.9, where we can see the difference between a clustering with no micro clusters consideration and one where the clusters with less than 3 elements, before the second step of the algorithm, are joined with other clusters. The number of clusters in each step of the clustering process was, in the first case (Fig. 4.9a): 18, 16, 12, 6, 5, 5 and in the second case (Fig. 4.9b): 18, 13, 8, 3, 3.

- We can repeat the previous process periodically along the clustering process, increasing the size of the acceptable micro clusters. This means that one can start by forcing the clustering of micro clusters with, for instance, less than 3 elements at a first stage and then repeat the process in posterior stages increasing, at each step, the minimum number of elements in each micro cluster. The only problem involved with this approach is the fact that one may wrongly exclude real small clusters from the final clustering.

- The clustering process is performed normally and, in the end, we can take



(a) Clustering with out micro cluster detection.



(b) Clustering with micro cluster detection.

Figure 4.9: Different solutions considering or not micro cluster detection.

actions in order to include the micro clusters in the formed clusters. This option, however, has the disadvantage of the possible inclusion of outliers and noise in real clusters.

- We just leave the final micro clusters as they are and consider them as noise or outliers.

Other aspects that can be considered for inclusion in the algorithm in order to try to achieve better results are:

- Regarding the possibility of joining two clusters, we have presented in section 4.2.2 two possible solutions based on a fixed number k of connections between elements of two different clusters. One of the options that can be included in the algorithm is to increase the number k as we evolve in the clustering process. This option is based on the fact that, as we evolve in the clustering process, larger clusters will be formed and, therefore, the probability of having a higher number of connections between clusters will be much higher.

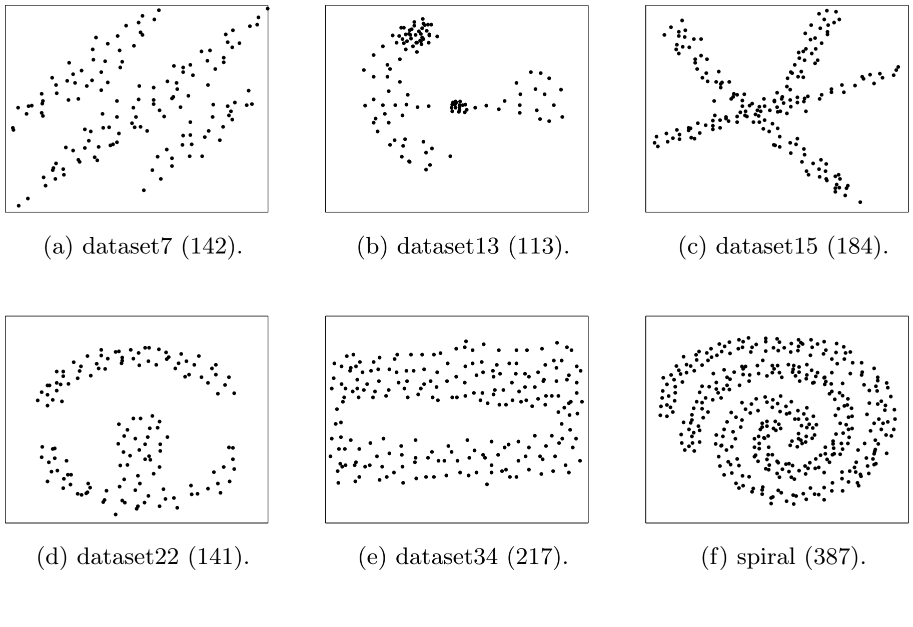
- At each step of the algorithm we only use the connections of the corresponding layer to evaluate the possibility of joining two clusters. It may happen that at a certain step, the number of connections between two clusters may not be sufficient to perform their union but, at a further step, when considering the connections of all the previous steps, we may reach the number k of necessary connections to joint the two clusters. One could then use as an optional feature the evaluation of the clustering process at each step based on all previous connections.

We have made some experiments with these two options but we didn't reach a conclusion about the type of data sets that benefit from their use.

Despite the fact that we do not use any kind of intra-cluster or inter-cluster association measure, we can easily introduce this concept in our algorithm but this would highly increase its complexity, turning against our objective of a extremely simple clustering algorithm. We also suspect that the introduction of our new entropic dissimilarity measure in other hierarchical and graph-based algorithms may improve their results.

4.3 Experiments

We have experimented the LEGClust algorithm in a large variety of applications. We have performed experiments with the real data sets, UBIRIS, NCI Microarray, 20NewsGroups, Dutch Handwritten Numerals (DHN), Iris, Wdbc, Wine, and Olive, some of them with a large number of features, and also with several artificial two-dimensional data sets. The artificial data sets were created in order to better visualize and control the clustering process and some examples are depicted in Fig. 4.10. For the artificial data set problems the clustering solutions yielded by different algorithms were compared with the majority choice solutions obtained in the human clustering experiment mentioned in Section 4.2 and described in Appendix B. For real data sets the comparison was made with the supervised classification of these data sets with the exception of the UBIRIS data set where the objective of the clustering tasks was the correct segmentation of the eye iris. In both cases – majority choice or supervised classes – we will designate these solutions as reference solutions or reference clusters.



(a) dataset7 (142). (b) dataset13 (113). (c) dataset15 (184).

(d) dataset22 (141). (e) dataset34 (217). (f) spiral (387).

Figure 4.10: Some of the artificial data sets used in the experiments (in brackets the number of elements).

We have compared our algorithm with several well known clustering algorithms: Chameleon algorithm, two Spectral clustering algorithms, DBScan and Mean Shift algorithms.

The Chameleon clustering algorithm, included in Cluto [107], is a software package for clustering low and high-dimensional data sets. The parameters used in the experiments, among the innumerous used by Chameleon are mentioned in the results. In fact, the number of parameters needed to tune this algorithm was one of the main problems we encountered when we tried to use it in our experiments. To perform the experiments with Chameleon we followed the advice in [108] and in the manual for the Cluto software [106].

For the experiments with the spectral clustering approaches we implemented the algorithms (Spectral-Ng) and (Spectral-Shi) presented in [141] and [175] respectively. One of the difficulties with both Spectral-(Ng/Shi) algorithms, is the choice of the scaling parameter. Extremely small changes in the scaling parameter produced very different clustering solutions. In these algorithm the number of clusters is the number of eigenvectors used to perform clustering. The number of clusters is a parameter that is chosen by the user in both algorithms. We tried to make this choice, in Spectral-Ng, an automatic procedure by implementing the algorithm presented in [162]; this, however, produced poor results. When making the choice of the cluster centroids in the k-means clustering used in Spectral-Ng we performed a random initialization and 10 restarts (deemed acceptable by the authors).

We tested the adaptive Mean Shift algorithm [33] in our artificial data sets and the results were very poor. In most of the cases the proposed clustering solution has a high number of modes and, consequently, a high number of clusters. For problems having a small number of points, the estimated density function will present, depending on the window size, either a unique mode if we use a large window size, or several modes not corresponding to really existing clusters, if we use a small window size. An example of a clustering solution given by this algorithm is presented in Fig. 4.11. As we can see, the number of modes is extremely high and, even if we optimize the result by joining modes that are very close to each other, we still have a considerable number of modes. We think that this algorithm probably works better with large data sets. An advantage of this algorithm is the fact that one does not have to specify the number of clusters as these will be driven by the data according to the number of modes.

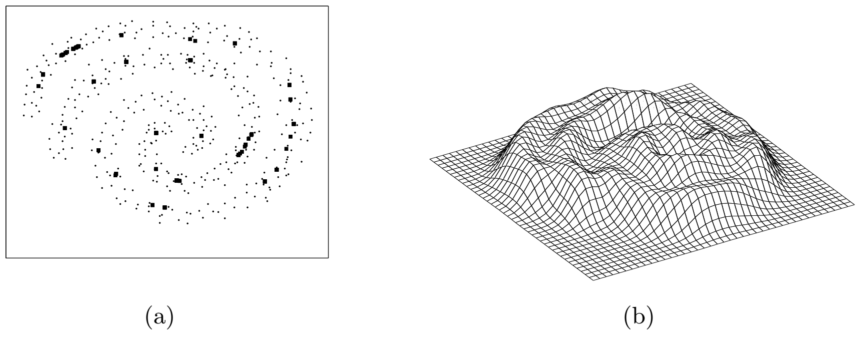
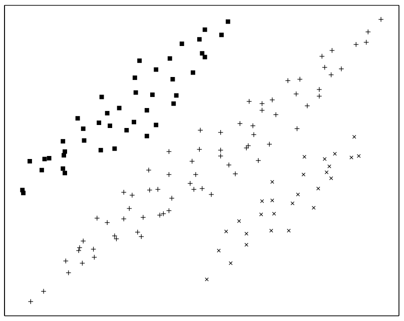


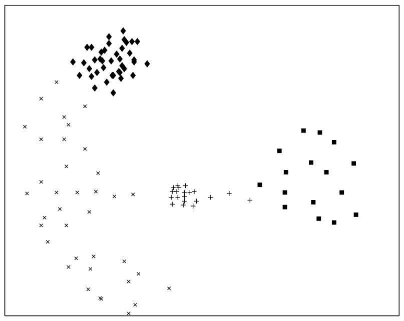
Figure 4.11: An example of the number of modes (square points) obtained with the adaptive Mean Shift algorithm (considering 30 nearest neighbors) when applied to the spiral data set (a) and an estimated probability density functions of the same data set (b).

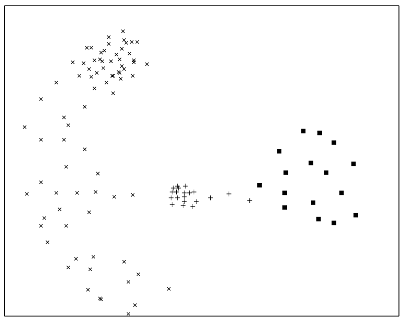
The DBScan algorithm is a density based algorithm that claims to find clusters of arbitrary shapes, but presents, basically, the same problems as the Mean Shift algorithm. It is based in several density definitions between a point and its neighbors. This algorithm only requires two input parameters, Eps and MinPts, but small changes in their values, specially in Eps, produce very different clustering solutions. For our experiments we used the implementation of DBScan available in [200].

In the LEGClust algorithm the parameters involved are: the smoothing parameter (h), related to the Parzen pdf estimation; the number of neighbors to consider (M); and the number of connections to join clusters (k). For the parameter h we used in all experiments the proposed formula 4.3. For the other two parameters we indicate in each experiment the chosen values.

Regarding the experiments with artificial data sets, depicted in Fig. 4.10, we present in Figures 4.12 and 4.12 the results obtained with LEGClust.


(a) dataset7; M=14; k=3;
43 33 17 10 5 $\underline{3}$ 3 3 2 2 1.


(b) dataset13; M=10; k=3;
11 7 5 $\underline{4}$ 3 3 3.


(c) dataset13; M=10; k=3;
11 7 5 4 $\underline{3}$ 3 3.

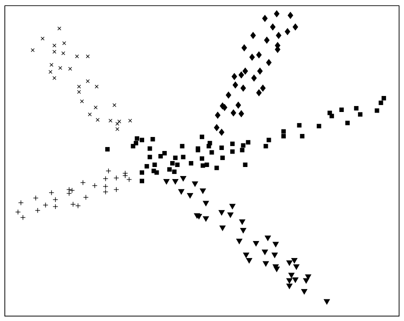
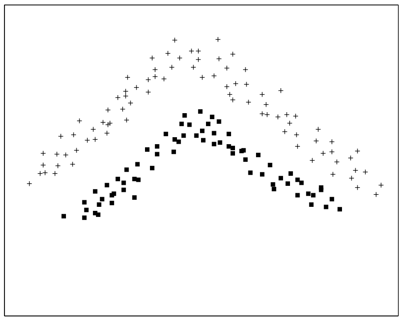

(d) dataset15; M=18; k=2;
55 37 16 7 $\underline{5}$ 3 2 1.

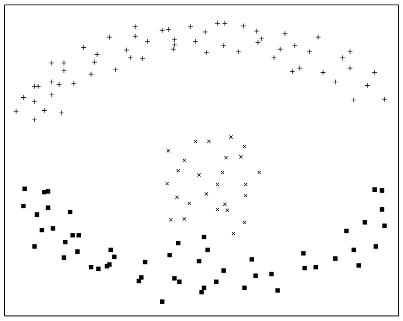
Figure 4.12: The clustering solutions suggested by LEGClust (part 1). Each label shows: the data set name; the number of neighbors (M); the number of connections to join clusters (k); the number of clusters found at each step of the algorithm (underlined is the selected step).

In Fig. 4.14 we present the solutions obtained with Chameleon algorithm that differ from those suggested by LEGClust.

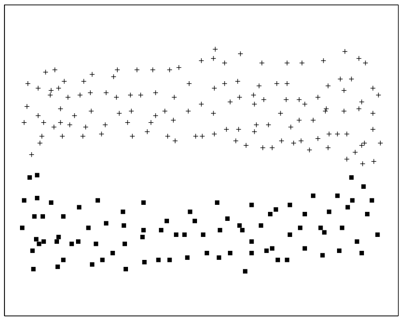
From the performed experiments, an important aspect noticed when using the Chameleon algorithm was the different solutions obtained for slightly different parameter values. Data set 4.14c was the one where we had more difficulties in tuning the parameters involved in Chameleon algorithm. A particular difference between the Chameleon and LEGClust results corresponds to the curious



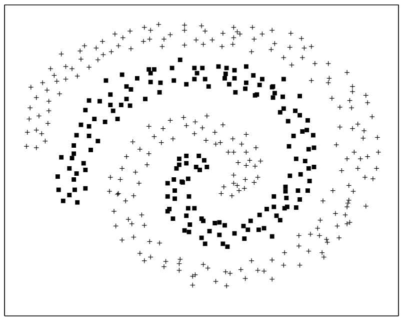
(a) dataset20; M=15; k=3;
41 28 18 10 4 $\underline{2}$ 2 1.



(b) dataset22; M=14; k=2;
45 26 12 8 5 $\underline{3}$ 2 2.



(c) dataset34; M=20; k=3;
68 59 36 25 15 8 5 3 $\underline{2}$ 2.



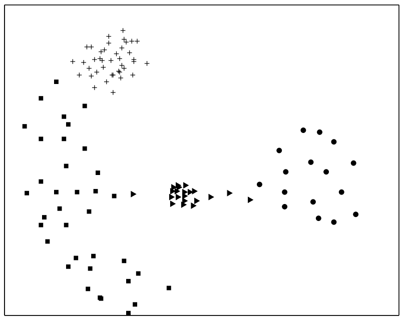
(d) spiral; M=30; k=2;
116 72 28 14 5 3 $\underline{2}$ 1.

Figure 4.13: The clustering solutions suggested by LEGClust (part 2). Each label shows: the data set name; the number of neighbors (M); the number of connections to join clusters (k); the number of clusters found at each step of the algorithm (underlined is the selected step).

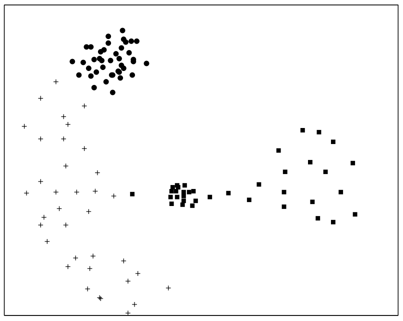
solution given by Chameleon, and depicted in Fig. 4.14b. When choosing 3 clusters as input parameter (nc=3) this solution is the only solution that is not suggested by the individuals that performed the tests referred in Section 4.2. The solutions for this same problem, given by LEGClust, are shown in figures 4.12b and 4.12c.

The spectral clustering algorithms gave some good results for some data sets, but they were unable to resolve some non-convex data sets like the double spiral problem (Figures 4.15 and 4.16).

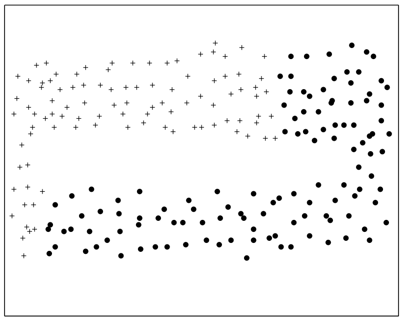
The DBScan algorithm clearly fails in finding the reference clusters in all



(a) dataset13; nc=4; a=20; n=20.



(b) dataset13; nc=3; a=20; n=20.

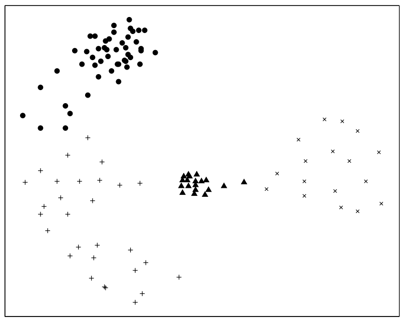


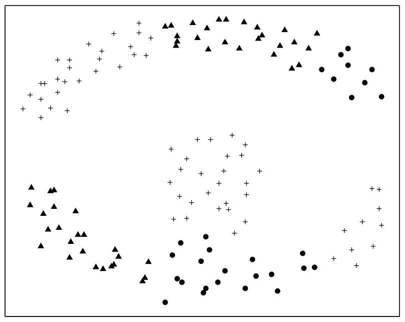
(c) dataset34; nc=2; a=50; n=6.

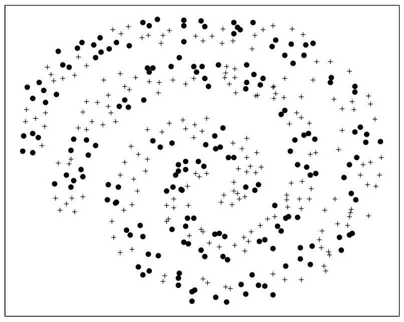
Figure 4.14: Some clustering solutions suggested by Chameleon. The considered values nc, a and n are shown in each label.

data sets (except the one of Fig. 4.10a).

Comparing the results given by all the algorithms applied to the artificial data sets we clearly see, as expected, that the solutions obtained with the density based algorithms are worse than those obtained with any of the other algorithms. The best results were achieved with the LEGClust and Chameleon algorithms.


(a) dataset13; nc=4; $\sigma = 0.065$.


(b) dataset22; nc=3; $\sigma = 0.071$.


(c) spiral; nc=2; $\sigma = 0.0272$.

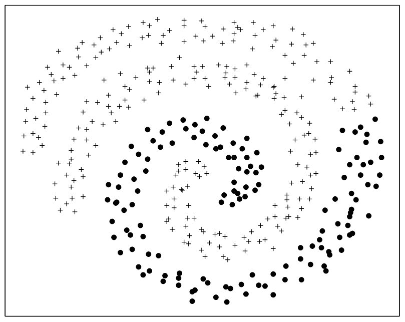
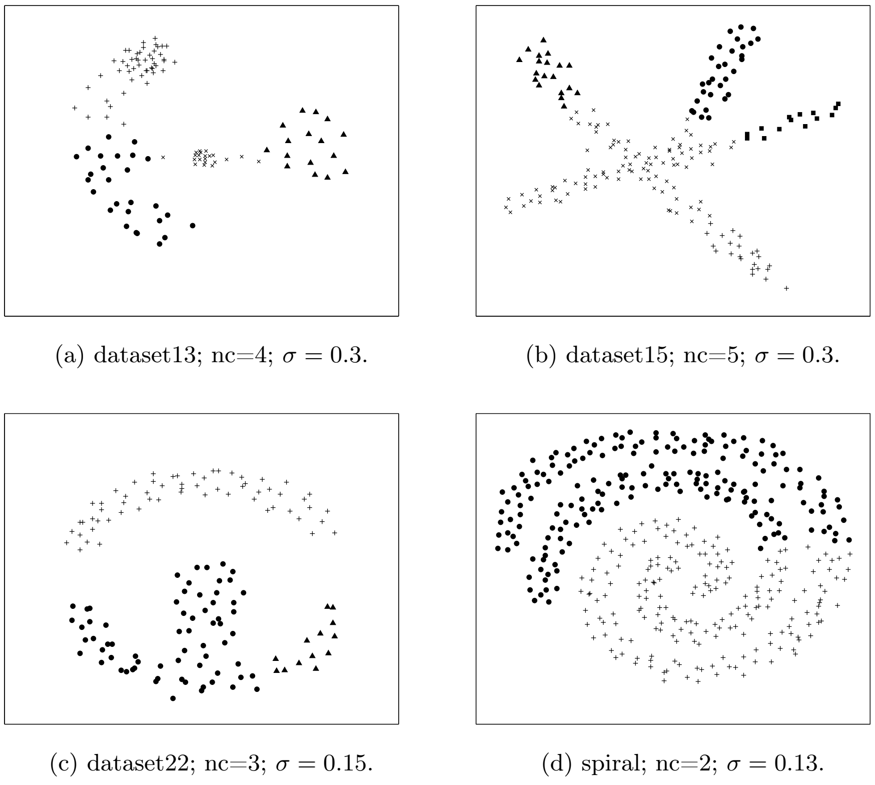

(d) spiral; nc=2; $\sigma = 0.0275$.

Figure 4.15: Some clustering solutions suggested by Spectral-Ng. Each label shows: the data set name; the pre-established number of clusters; the σ value.


(a) dataset13; nc=4; $\sigma = 0.3$.

(b) dataset15; nc=5; $\sigma = 0.3$.

(c) dataset22; nc=3; $\sigma = 0.15$.

(d) spiral; nc=2; $\sigma = 0.13$.

Figure 4.16: Some clustering solutions suggested by Spectral-Shi. Each label shows: the data set name; the pre-established number of clusters; the σ value.

We now present the performed experiments with LEGClust in real data sets and the comparative results obtained with the different clustering algorithms.

We start by presenting the results of the segmentation performed by LEG-Clust and Spectral-Ng in the images of UBIRIS data set sample. The results for this image segmentation problem are depicted in Fig. 4.17 (second and third columns). In all experiments with LEGClust we used the values $M = 30$ and $k = 3$. For the experiments with Spectral-Ng we chose 5 as the number of final clusters. We can see by the segmentations produced that both algorithms gave acceptable results. However, one of the striking differences is the way Spectral clustering splits each eyelid in two by its center region (the third image is a good example of this behavior) that is also observable if we choose different numbers of clusters.

To test the sensitivity of our clustering algorithm to different values of the parameters we have made some experiments with different values of M and k, in the UBIRIS data set sample. An example is shown in Fig. 4.18. We can see that, different values of M and k do not affect substantially the final result of the segmentation process; the eye iris in all solutions is distinctly obtained.

Experiments with the DHN data set were performed with LEGClust and Spectral clustering and their results compared. These results are presented in Table 4.7. ARI stands for Adjusted Rand Index, a measure for comparing results of different clustering solutions when the labels are known [86]. This index is an improvement of the Rand Index; it lies between 0 and 1 and the higher the ARI index is the better is the clustering solution. The parameters for both Spectral clustering and LEGClust were tuned to give the best possible solutions. We can see that, in this problem, LEGClust performs much better than Spectral-Shi and with similar (but slightly better) results than Spectral-Ng. We also show in Table 4.7 some different results for LEGClust obtained with different choices of the minimum number of connections (k) to join clusters. In these results we can see that different values of k produce results with small differences in the ARI

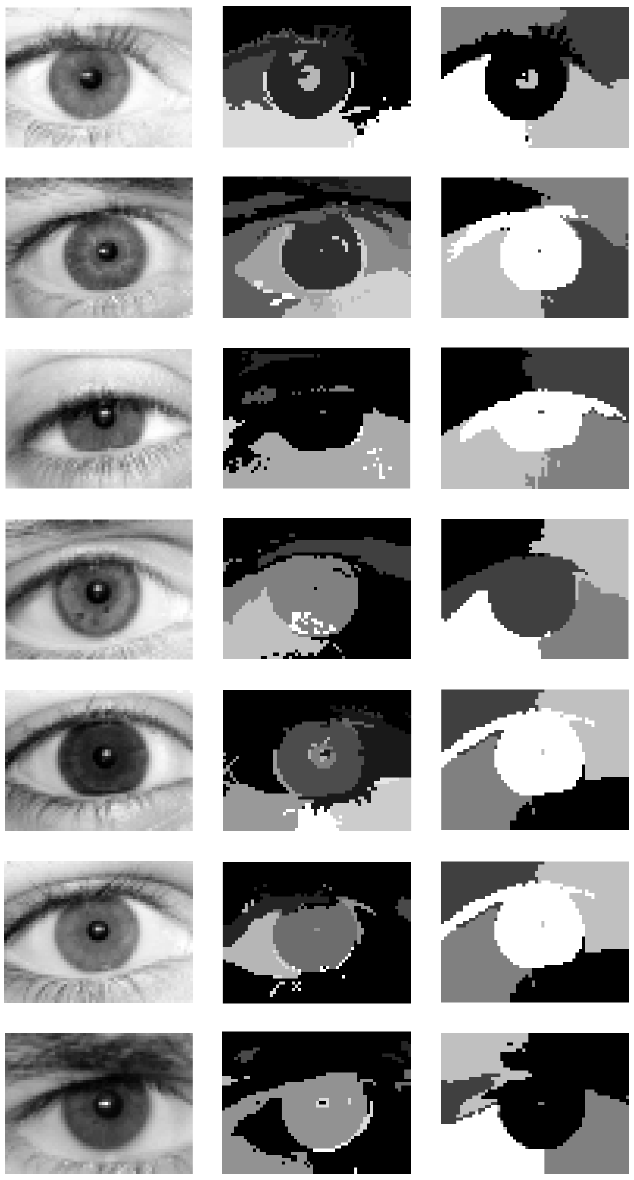
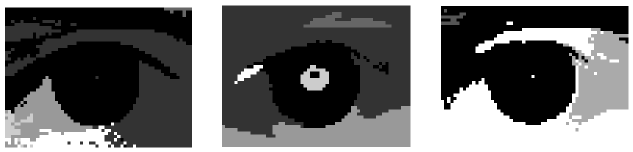


Figure 4.17: Sample from UBIRIS data set (first column) and the results of the LEGClust (second column) and Spectral (third column) clustering algorithms. The number of clusters for LEGClust was 8, 9, 12, 7, 5 and 8 respectively, with $M = 30$ and $k = 3$.



(a) $k = 2$ (b) $M = 10$ (c) $M = 20$

Figure 4.18: Segmentation results for the fourth image (line 4) of Fig. 4.17 using different values of k or M.

index.

In Table 4.6 we show an example of the confusion matrix obtained with LEGClust for an experiment with the DHN data set.

In the experiments with the 20NewsGroups data set we have compared the LEGClust algorithm with both Spectral clustering algorithms. The results of the experiments are shown in Table 4.7. We can see that the best results are obtained with the Spectral-Ng algorithm. However the results of LEGClust are much better when compared with Spectral-Shi.

Table 4.6: A confusion matrix of a clustering solution given by LEGClust for the DHN data set.

	Classes									
	176	1	0	1	0	3	9	0	169	3
	1	0	0	0	0	0	0	0	19	0
	7	2	1	3	14	178	6	16	1	188
	0	2	195	5	0	0	0	83	3	3
Clusters	1	10	4	7	1	4	0	101	0	3
	0	167	0	1	3	0	1	0	2	2
	0	2	0	0	86	0	1	0	1	0
	0	8	0	3	95	0	1	0	2	1
	0	8	0	180	0	14	1	0	0	0
	15	0	0	0	1	1	181	0	3	0

In the experiments performed with the NCI Microarray data set we also have compared the LEGClust and Spectral clustering algorithms. To perform these experiments we have chosen 3 clusters, following the example in [78], as the final number of clusters for both algorithms. The results are also shown in Table 4.7. Again the results produced by LEGClust were quite insensitive to the choice of parameter values.

Table 4.7: The results and parameters used in the comparison of LEGClust and Spectral clustering in experiments with DHN, 20NewsGroups and NCI Microarray data sets.

	LEGClustt			Spectral-Ng			Spectral-Shi		
DHN	M	k	**ARI**	nc	σ	**ARI**	nc	σ	**ARI**
	30	10	**0.628**	10	12	**0.573**	10	10	**0.287**
	30	8	**0.608**						
	30	12	**0.574**						
20NewsGroups									
	20	3	**0.289**	20	12	**0.479**	20	20	**0.006**
	20	2	**0.287**						
NCI Microarray									
	4	2	**0.148**	3	80	**0.177**	3	10	**0.138**
	6	3	**0.148**						
	10	3	**0.148**						

The results presented in Table 4.7 show that the LEGClust algorithm performs better than the Spectral-Shi algorithm in the three data sets and, compared with Spectral-Ng, it gives better results in the DHN data set and similar ones in the NCI Microarray.

In the experiments with the data sets Iris, Olive, Wdbc and Wine, we compared the clustering solutions given by LEGClust and Chameleon. The parameters used for each experiment and the results obtained with both algorithms are shown in Table 4.8. Each experiment with the Chameleon algorithm, followed the command: **vcluster** dataset_name number_of_clusters=**nc** -clmethod=graph -sim=dist -agglofrom=**a** -agglocr=wslink -nnbrs=**n** given in [106]. The final number of clusters is the same as the number of classes. We can see that the results with LEGClust are better than the ones obtained with Chameleon, except for the data set Olive.

Table 4.8: The results and parameters used in the comparison of LEGClust and Chameleon in experiments with 4 real data sets.

	Chameleon			LEGClust		
Data set	a	n	ARI	M	k	ARI
Iris	9	50	0.658	15	3	0.750
Olive	40	40	0.733	25	3	0.616
Wdbc	40	25	0.410	20	3	0.574
Wine	30	21	0.400	15	3	0.802

Finally we also experimented our algorithm in two images from [54], used to test textured image segmentation. We show in Fig. 4.19 the results obtained and the comparison with those obtained by Fischer [54] with their Path-based algorithm. We are aware that our algorithm was not designed having in mind the specific requirements of texture segmentation; as expected, the results were not as good as those obtained in [54], but nevertheless LEGClust was still capable of detecting some of the structured texture information.

In this chapter we have presented a new proximity matrix, built with a new entropic dissimilarity measure, as input for clustering algorithms. We also presented a simple clustering process that uses this new proximity matrix and performs clustering by combining a hierarchical approach with a graph technique.

The new proximity matrix and the methodology implemented in the LEG-

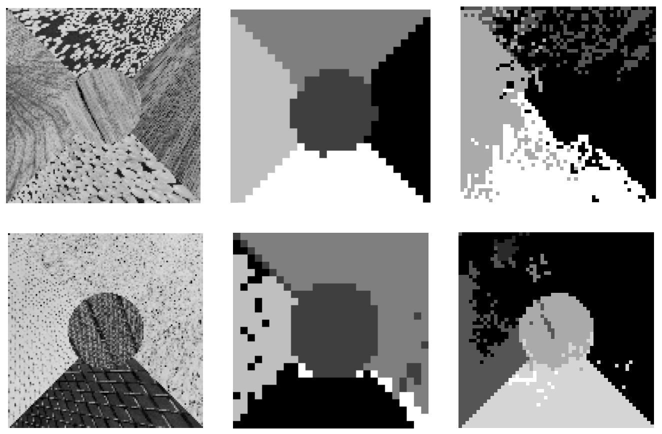


Figure 4.19: Segmentation results for textured images (left hand side) with Fischer's path-based clustering (midle) and LEGClust (right). The parameters used in LEGCLust were $M = 30$ and $k = 3$ and the final number of clusters was 3 (top) and 6 (bottom).

Clust algorithm allows taking into account the local structure of the data, represented by the statistical distribution of the connections in a neighborhood of a reference point achieving a good balance between structuring direction and local connectedness. In this way, LEGClust is able, for instance, to correctly follow a structuring direction presented on the data, with sacrifice of local connectedness (minimum distance), as human clustering often does.

In the next chapter we will use LEGClust to perform modular neural network task decomposition.

Chapter 5

MNN with Entropic Task Decomposition

As we said in the beginning of the previous chapter, the proposed new clustering algorithm was developed because we intended to perform modular neural network task decomposition based on entropic criteria. The purpose of this chapter is to present the experiments with modular neural networks with task decomposition performed with LEGClust (a small part of this chapter was presented in [165]). We will start the chapter with a brief introduction to modular neural networks and the task decomposition process followed by the presentation of the results obtained in classification problems.

The use of simple neural networks such as multi-layer perceptrons has some drawbacks: e.g. slow learning, weight coupling or the black box effect. These can be alleviated by using a modular neural network (MNN). A modular neural network is an ensemble of learning machines. The idea behind this kind of learning structure is the divide-and-conquer paradigm: the problem should be divided into smaller subproblems that are solved by experts (modules) and their partial solutions should be integrated to produce a final solution. (Figure 5.1).

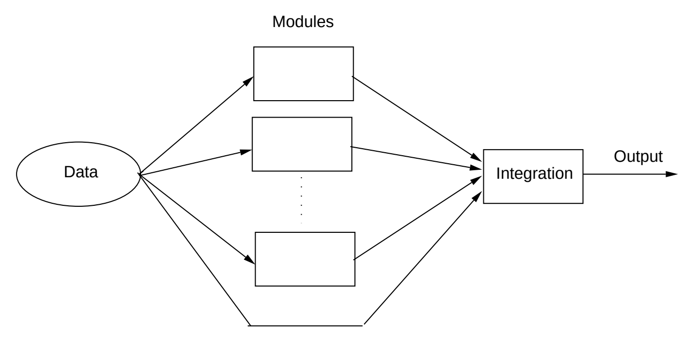


Figure 5.1: A modular neural network.

Ensembles of learning machines proved to give, in some cases, better results than the single learners that produce them. The proofs are mainly empirical [15, 38] but there are some theoretical results [5, 6, 111] that also support this assumption.

The purpose here is to address the issue of using our entropic clustering algorithm (LEGClust) to perform the task decomposition in a modular neural network (MNN) approach, as opposite to the traditional methods. Task decomposition is one of the strategies used to simplify the learning process of any learning system. In neural networks it basically consists on the partition of the input space in several regions, this way decomposing the initial problem in different subproblems. This is done based on the assumption that these regions possess different characteristics and so they should be learned by specialized neural networks. By posterior integration of the learning results, we are able to hopefully achieve better solutions for the initial problem. Generally, task decomposition can be obtained in three different ways: explicit decomposition (the task is decomposed by the designer before training), class decomposition (the decomposition is made based on the classes of the problem) and automatic decomposition. When automatic decomposition is used, it can either be made during the learning stage or it can be made before training the modules, using an unsupervised or clustering algorithm. We use this last approach performing the task decomposition with LEGClust.

So, to use a MNN, three stages have to be considered:

- The task decomposition stage, where the problem is divided into smaller problems, each one to be delivered to one of the modules or expert networks.
- The training phase, where each individual expert (module) is trained until it learns to solve its particular subproblem.
- The decision integration. This strategy is used to combine the work of the experts and to produce a final network output. This can be done in several ways: using a gating network [90], making the modules vote [11] or through hierarchical integration (which can also use voting and/or gating networks) [91, 100]. In this work we consider the use of a gating network. This network can be considered as an additional expert that is trained to recognize the region of the input space where each of the experts have their region of expertise, defined in the task decomposition phase.

After finishing the learning process, when a new pattern to be classified is presented to the network, the individual experts compute the class it might belong, but the gate network selects only a particular output that is given by the expert it considers to be 'competent' to solve the problem, taking into account the region of the input space to which the pattern belongs.

5.1 Task decomposition and MNN Structure

As we mentioned before, the task decomposition is done before training the modules, using a clustering algorithm. Previous works like [4, 44, 189] and [5, 50] already used this approach. There are several well known algorithms to perform clustering, being the most common ones those based on matrix theory and graph theory. However, it is also known, and we have mentioned it on the previous chapter, that this kind of algorithms often have serious difficulties in identifying real clusters. The algorithms based on matrix theory build clusters according to some distance measure producing usually globular clusters and the algorithms based on graph theory, usually divisive algorithms, present some difficulties in the process of graph partitioning to obtain plausible clusters. In the following we propose the use of LEGClust to do the automatic task decomposition.

To better understand the need for a task decomposition process, let us take a look at the artificial data set depicted in Fig.5.2.

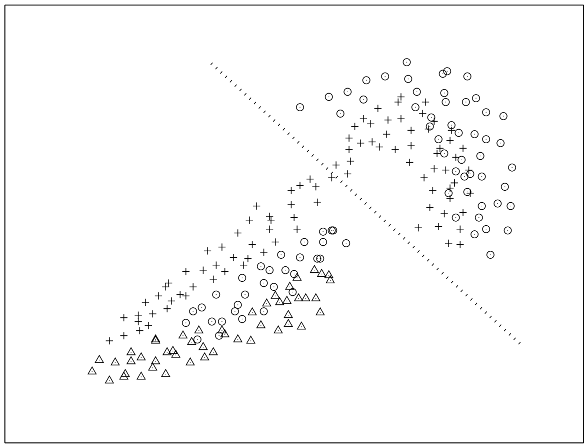


Figure 5.2: The partition of the input space for a 3 class problem.

This is a 3 class problem where the input space is clearly divided into 2 regions: one of the regions (upper right) contains samples from 2 classes (crosses and circles) and the other contains samples from all 3 classes. Note that there are two classes with samples belonging to the 2 different regions. This classification problem, apparently trivial, will require a complex MLP to obtain a satisfactory solution due to the fact that there are non convex and disjoint classes. By having a classifier dedicated to each region we are able to transform this particular problem into two simpler ones with each classifier responsible to learn its subproblem defined by its region of interest.

A schematic view of the structure of our modular neural network and all the steps involved in the training phase of the MNN is depicted in Fig. 5.3.

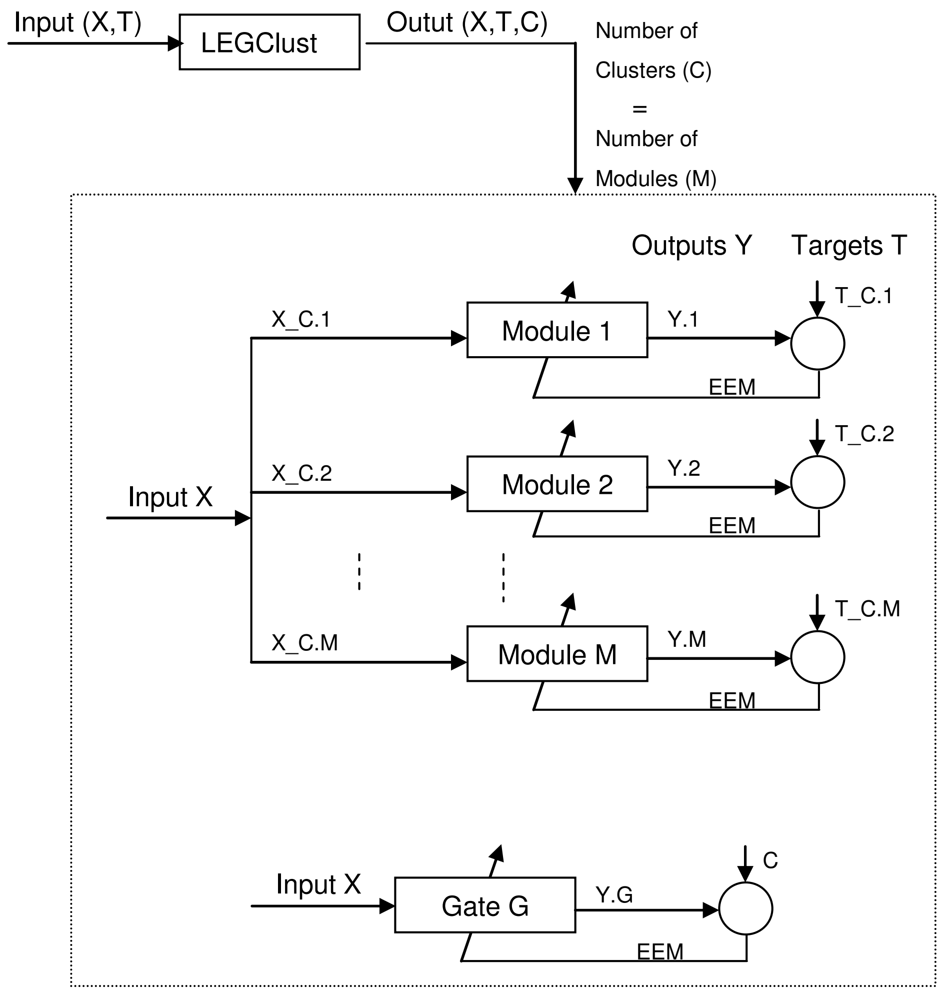


Figure 5.3: Schematic view of the structure and the training phase of the Modular Neural Network.

Each module, including the gate, is an MLP.

The process starts with the partition of the input space in C clusters. This partition is performed by the LEGCLust algorithm that receives as input the features matrix X and the target vector T and produces as output the same X and T and an additional vector C representing the clusters labels for the elements of the data set.

We have already mentioned, in the previous chapter, that the final number of clusters in a clustering process is very subjective and is also influenced by the problem domain. In classification problems to be solved by a MNN approach, one must not work with a high number of final clusters because the problem would be extremely partitioned and, consequently, the errors induced by the gate module, responsible to learn the division of the input space, would probably be very high. In fact, this, and the absence of natural clusters in a data set, are the main obstacles to use a MNN in a classification problem. If the data set does not possess natural clusters, by forcing the division of the input space in, at least, two clusters, the errors induced by the gate would be, probably, higher than those produced by a single MLP. To better understand this problem let us look at Fig.5.4.

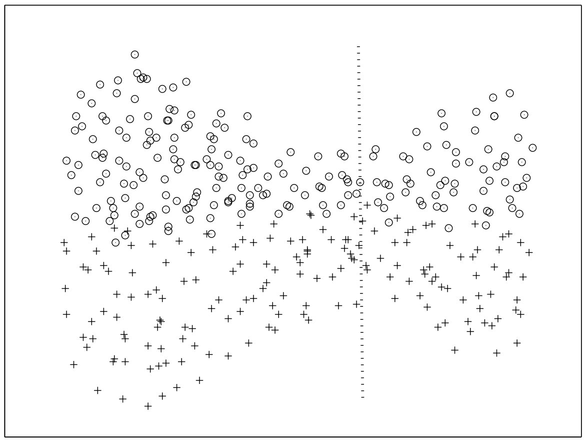


Figure 5.4: The possible partition of the input space in 2 clusters for this 2 class problem would produce worse results with a MNN than with a simple MLP.

This is a very simple problem for a simple MLP (the classes are almost linear separable). By dividing the input space in two clusters, like the ones depicted in

Fig. 5.5, obtained with the clustering process, the errors produced by the gate module would be responsible for a worse solution than the possible one obtained with a single MLP.

To illustrate the training phase, let us return to the example of Fig.5.2 and let us suppose that the clustering process divides the data set in two clusters, C_1 and C_2, depicted in Fig. 5.5.

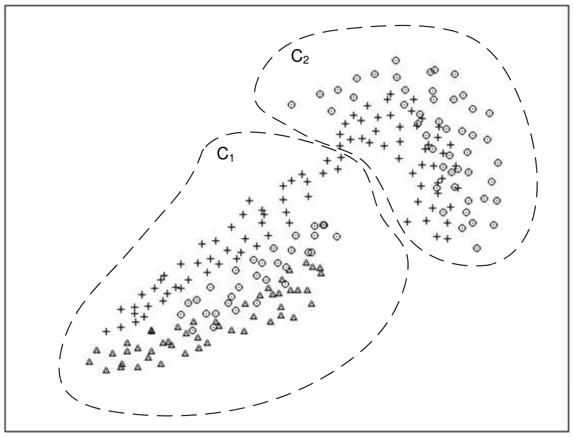


Figure 5.5: The possible 2 clusters originated by the clustering process applied to the data set of Fig.5.2.

The MNN for this problem will have two modules and the gate. The two modules are responsible for learning each of the subsets of the data defined by each cluster and the gate is responsible for learning the problem as if it was a 2-class problem like the one depicted in Fig. 5.6.

Each module, M_1 and M_2, is trained with the subset of the training data corresponding to each cluster, C_1 and C_2, respectively. The targets of these subsets are the original class labels. Module M_1 is trained with 3 classes and module M_2 with 2 classes. So, we have a $[2, n_h, 3]$ MLP for module M_1 and a $[2, n_h, 2]$ MLP for module M_2. The gate is trained with all the training set data but now the initial targets are substituted by the clusters labels (Fig. 5.6). By doing this, the gate network will learn to identify the data corresponding to each input region (cluster). The training process for each module/gate is performed

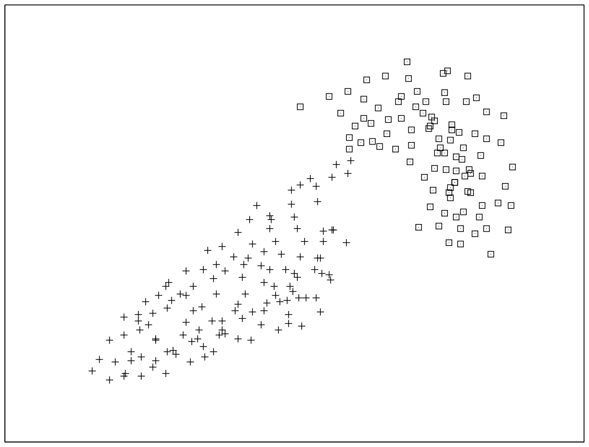


Figure 5.6: The gate is trained as if the initial problem is transformed in a 2-class problem like the one depicted here.

exactly the same way as in the MLP training. In addition we also use in each module/gate our EEM algorithm as referenced in Fig. 5.3.

In the test phase, for each new pattern that is presented to the MNN, the gate will determine which module is going to classify this new pattern, depending on the output label. If the gate classifies the pattern as belonging to C_1, the output of module M_1 is the one that is chosen, and the similar for cluster C_2 and module M_2. Other strategies can be used, such as the combination of both outputs, to obtain a classification for each pattern, but we didn't consider this hypothesis in our work.

5.2 Experiments

We have performed a considerable number of experiments with several data sets using modular neural networks with task decomposition performed by different clustering algorithms: one using k-means (K-MNN), another using Spectral Clustering [141] and the last one using our entropic clustering (EC-MNN). All the neural networks used in the experiments, both the modules and the gates of the MNN, were MLP's with one hidden-layer. The topologies of the MLPs were $[a : n_h : c]$, where a is the number of features, n_h is the number of neurons in the hidden layer and c is the number of classes treated by each expert and the number of clusters/experts for the gates. The neural networks were trained with the back-propagation algorithm and early stopping. The experiments were made using the 2-fold cross validation method. Each module is trained with the input data defined by the clustering algorithm (each module learns with the data from each cluster). The gate network is trained with all the data labeled by the LEGClust algorithm.

Regarding the clustering processes, since none of them produce automatically the number of clusters, these must be defined by the user. Taking into account the data sets used in the experiments, we only considered the possibilities of 2 and 3 clusters. Otherwise, the training set for each module could have an insufficient number of samples.

We used in our experiments several real data sets: Breast Tissue, CTG, Diabetes, Olive, 2VowelsPB and Sonar, and also the artificial one depicted in Fig. 5.2 further designated as *ArtificialF5.2* containing 222 elements, 2 features and 3 classes.

In Table 5.1 we present the parameters of each modular neural network for the results presented in Table 5.2. For each type of MNN, we show the number of experts and, for each of them, we present the number of hidden neurons and the number of output neurons. The number of output neurons is defined by the number of classes in each cluster. The presented structures correspond to the best results in a large number of experiments with different combinations in the number of neurons in each module and in the gate.

In Table 5.3 we present the errors for the performed experiments with single neural networks, SNN, (MLP's with one hidden layer). The number of neurons in the hidden layer is shown in column n_h. The results in Tables 5.2 and 5.3 are the average and standard deviations for 20 repetitions of each experiment.

Table 5.1: Structure of the modular neural networks used in the experiments corresponding to the results in Table 5.2. The last topology corresponds to the gate structure.

Data set	Algorithm	# Modules	MNN Structure
ArtificialF5.2			
	EC-MNN	2	[2:20:3][2:12:2]-[2:14:2]
	K-MNN	2	[2:18:3][2:18:2]-[2:18:2]
	S-MNN	2	[2:18:3][2:18:2]-[2:16:2]
Breast Tissue			
	EC-MNN	2	[9:12:2][9:12:2]-[9:3:2]
	K-MNN	2	[9:10:2][9:12:2]-[9:12:2]
	S-MNN	2	[9:4:2][9:14:2]-[9:6:2]
CTG			
	EC-MNN	3	[22:20:6][22:18:2][22:26:2]-[22:26:3]
	K-MNN	3	[22:22:10][22:18:9][22:18:9]-[22:26:3]
	S-MNN	3	[22:18:10][22:22:10][22:22:10]-[22:22:3]
Diabetes			
	EC-MNN	3	[8:14:2][8:18:2][8:12:2]-[8:16:3]
	K-MNN	3	[8:10:2][8:12:2][8:10:2]-[8:12:3]
	S-MNN	2	[8:18:2][8:12:2]-[8:10:3]
Olive			
	EC-MNN	3	[8:10:4][8:4:3][8:12:2]-[8:8:3]
	K-MNN	3	[8:6:6][8:12:4][8:12:8]-[8:12:3]
	S-MNN	3	[8:12:4][8:12:4][8:4:3]-[8:6:3]
2VowelsPB			
	EC-MNN	2	[2:4:2][2:5:2]-[2:2:2]
	K-MNN	2	[2:6:2][2:6:2]-[2:2:2]
	S-MNN	2	[2:5:2][2:5:2]-[2:2:2]
Sonar			
	EC-MNN	2	[60:12:2][60:12:2]-[60:12:2]
	K-MNN	2	[60:12:2][60:12:2]-[60:14:2]
	S-MNN	2	[60:10:2][60:16:2]-[60:16:2]

Table 5.2: Errors and standard deviations for the performed experiments with MNN's with task decomposition made by K-means clustering (K-MNN), spectral clustering (S-MNN) and entropic (LEGClust) clustering (EC-MNN).

Data set	K-MNN	S-MNN	EC-MNN
ArtificialF5.2	16.40 (2.40)	15.32 (3.55)	14.70 (3.22)
Breast Tissue	58.95 (7.54)	33.53 (4.47)	32.79 (3.72)
CTG	22.90 (0.86)	23.91 (2.91)	20.67 (2.38)
Diabetes	24.45 (1.45)	23.96 (1.76)	23.89 (1.64)
Olive	49.11 (2.89)	5.20 (1.11)	4.74 (0.89)
2VowelsPB	7.23 (1.17)	7.28 (0.95)	7.25 (0.80)
Sonar	16.14 (3.43)	23.69 (4.57)	18.57 (3.40)

Table 5.3: Errors and standard deviations for the performed experiments with single neural networks (SNN).

Data set	SNN	n_h
ArtificialF5.2	19.56 (3.95)	20
Breast Tissue	32.75 (3.26)	22
CTG	15.70 (0.60)	20
Diabetes	23.90 (1.69)	15
Olive	5.45 (0.62)	15
2VowelsPB	7.51 (0.37)	6
Sonar	21.90 (2.80)	14

5.3 Results Discussion

We must start by reminding that the MNN approach, with its associated task decomposition, will only be effective, giving better results than single neural networks, if the input space possesses some divisive properties, i.e., if different regions of the input space exhibit clear data clusters as exemplified in Fig. 5.2.

This is the main reason why we first focused our experiments on the comparison between three different modular neural network task decomposition approaches, and secondly, we have compared the results obtained by these MNN's and the ones obtained with SNN's. So, what we have compared in these first experiments is the suitability of the entropic clustering to perform task decomposition when compared with other methods, namely the k-means and spectral clusterings. To achieve that comparison we have two choices: to use, for each data set, the same topology for the different modules of the MNN's for each method or, for each data set, to try to find the best topology for each module of the MNN for each method. We think that the second hypothesis is the most reasonable because one should expect that different clustering processes would produce different clusters/regions by differently dividing the input space. Therefore, the modules must possess different structures to learn each of the subproblems, depending on the cluster complexity.

We can see that the average errors, in almost every performed experiment, were smaller for the EC-MNN than for the K-MNN and S-MNN. In some cases the differences in the performances are very substantial, specially when comparing EC-MNN and K-MNN, in the Breast Tissue and Olive data sets. The 2VowelsPB is a data set with 2 well separated globular clusters, one containing class 1 and 2 and the other classes 3 and 4 (see Appendix A); that is the reason for the almost equal results for the three different MNN's.

We used the Wilcoxon or ranksum non-parametric test to check the statistic significance of the differences between similar results. The significance level used was $\alpha = 0.05$. The null hypothesis (medians are equal) can not be rejected for data set Diabetes between EC-MNN and K-MNN and for data sets Breast Tissue, Diabetes, ArtificialF5.2, and Olive between EC-MNN and S-MNN. Despite the results of the statistic tests, the means and standard deviations of the performed experiments are smaller for the EC-MNN.

When comparing the results of the performed experiments for both MNN and single MLP's (Tables 5.2 and 5.3), we can see that, for the data set ArtificialF5.2 and 2VowelsPB we have a reduction in the final error when using MNN's of any kind (in the 2VowelsPB data set the difference is very small, due to the nature of the problem). Comparing only the results of the SNN and the EC-MNN for the remaining data sets, we can see that there are some data sets where the final errors are considerable better for the EC-MNN. This is the case of data sets Olive and Sonar. On the other hand, the results for data set CTG are better when using a SNN. A possible explanation for this fact is that, as we mentioned before, the data set must possess some divisive properties to obtain some gain in using MNN's. We know that data set Olive probably possesses some divisive characteristics because olive oil samples came from three different regions of Italy (see Fig. A.3). For the remaining data sets, the results for EC-MNN and SNN are very similar.

Chapter 6

Conclusions

Our work in data classification with entropic criteria is a contribution both in supervised classification with multi-layer perceptrons and in unsupervised classification. In the following we will present a review of the contributions made in this thesis.

6.1 Contributions

Following previous works on the application of entropic criteria to regression and prediction, we have introduced and applied the error entropy minimization algorithm (EEM) for classification with MLP's. We have applied it in classification problems with data sets with different number of elements, features and classes (balanced and unbalanced). The EEM algorithm proved to give better results than the usual MSE in almost every performed experiment [1].

In order to improve the learning process with EEM we have implemented several optimization procedures. We first tried to use a variable smoothing parameter in the entropy gradient computation with the purpose of a better gradient estimation. This attempt didn't produce good results and we explained the reasons for it. For this reason, we used a fixed smoothing parameter and, in

[1]Despite the fact that we do not presented here a comparison of the EEM algorithm with other well known classification methods such as SVMs or even other cost functions for MLPs like the cross-entropy, we have made several experiments comparing our algorithm with these last two and the results also show that EEM is a valid alternative.

a following phase, we performed experiments trying to obtain a formula for it because the known formulas for pdf estimation proved to be inadequate for our algorithm. After performing a large variety of experiments, we proposed a new formula for the EEM smoothing parameter. In another phase of our work, with the purpose of achieving a faster convergence, we presented several algorithms for adaptive learning rate. With this optimization strategy, we were able to speed-up the learning process with good final results. The last optimization procedure was the batch-sequential algorithm. It was implemented in order to gain from the advantages of both methods and also to reduce the complexity of the EEM algorithm. Since the computational complexity of the entropy and its gradient is proportional to the number of elements of the training set, the algorithm became very time demanding for large data sets. With the batch-sequential algorithm we were able to achieve, in some cases, a reduction on the computational time of 5:1.

In the second part of our research, we have applied the entropy to clustering problems by developing a new clustering algorithm: LEGClust. This clustering algorithm is based on layered entropic subgraphs build based on a proximity matrix obtained from a new entropic dissimilarity matrix. This dissimilarity matrix is built with an entropic measure obtained by computing, for all the data set elements, the entropy of the difference vectors in a neighborhood of every element. With this entropic measure we were able to capture the local structure of the data allowing us to apply LEGClust to data sets with clusters of different "shapes" without knowing them *a priori*. The experiments with the LEGClust algorithm in both artificial and real data sets have shown that: it achieves good results compared with other classical and sophisticated clustering algorithms; it is simple to use, since it only needs to adjust 3 parameters and simple guidelines for these adjustments were presented; its sensitivity to small changes of the parameter values is low; it often yields solutions that are majority voted by humans; and, it is a valid proposal for data sets with any number of features.

We have finished this work by implementing a modular neural network (MNN) for classification using the LEGClust algorithm in its task decomposition phase. The use of a clustering algorithm is one of the possible strategies to perform task decomposition. The input space of the problem is divided into several regions, with each one being learned by a different module (NN). We have compared our clustering algorithm with the k-means and with the spectral clustering algorithms. The performed experiments have shown that the use of LEGClust to perform task decomposition is a valid approach, that there are several data sets that can benefit from it, and that (generally) this approach will yield better results (smaller classification error) than those using k-means and spectral clustering algorithms. We have also compared the results obtained with our MNN with entropic task decomposition with the results obtained with a single neural network. They have shown that better results with such a modular approach are only expected when the classification problem possesses some divisive characteristics. Otherwise, by dividing the input space in several regions, the error introduced by the decision integration of the different modules may lead to a worse final classification error.

6.2 Future Work

First of all, we must say that we think to have achieved the proposed objectives for this work and that we have fulfilled our initial expectations. We are aware that several other related research topics are opened to research and deserve a future study. Among other possible subjects we mention the following ones:

The use of entropic cost functions in classification problems with MLPs must be the subject of a further theoretical study. The nature of the errors, being the combination of two random variables of different kinds – the output of the NN is a continuous variable and the defined targets are discrete variables – make this issue a very complex one in theoretical terms.

We think that a supplemental effort should be made in order to try to use a variable smoothing parameter during the training phase.

We have made several preliminary experiments comparing the results obtained by NNs with the EEM algorithm with other well known classifiers such as SVMs or other cost functions such as cross-entropy. The results show that EEM is comparable, with slightly better results, to cross-entropy and, when compared to SVMs, it gives better results in some data sets and worse results in others. Further experiments should be performed in order to try to establish the kind of problems where one may expect better results when using the EEM algorithm. This, is in fact, a metalearning research task.

We will try to include our entropic measure in other existing hierarchical and graph based algorithms in order to compare them with LEGClust algorithm. By doing this, we can also establish the importance of the entropic measure in the clustering process.

The implementation of a clustering process using as input our entropic dissimilarity matrix with a different approach than the one presented here, independent of the user choice of parameters, and with a fixed number of clusters if so desired, is one of the open research perspectives.

Further more, the possibility of combining our entropic measure with measures of intra- and inter-cluster association in order to try to obtain an even better final result, in more specific clustering problems like image segmentation, is also an open research project.

Appendix A

Data Sets

In this appendix we present information about the real data sets used in this work. Table A.1 contains a summary of the characteristics of these data sets.

In the following sections we present detailed information about each one of the data sets.

Table A.1: The real data sets used in this work.

Data set	*# samples*	*# features*	*# classes*
20NewsGroups	1000	565	20
2VowelsPB	608	2	4
Breast Tissue	106	9	6
CTG	2126	22	10
DHN	2000	3	10
Diabetes	768	8	2
Ionosphere	351	33	2
Iris	150	4	3
NCI Microarray	64	6830	12
Olive	572	8	9
Sonar	208	60	2
UBIRIS		image data set	
Wdbc	569	30	2
Wine	178	13	3

20NewsGroups

The 20 Newsgroups data set can be found in the UCI repository of machine learning databases [22]. It is a collection of approximately 20000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups. The 20 newsgroups collection has become a popular data set for experiments in text applications of machine learning techniques, such as text classification and text clustering.

The data is organized into 20 different newsgroups, each corresponding to a different topic. Some of the newsgroups are very closely related to each other, while others are highly unrelated. Here is a list of the 20 newsgroups: alt.atheism; comp.graphics; comp.os.ms-windows.misc; comp.sys.ibm.pc.hardware; comp.sys.mac.hardware; comp.windows.x; misc.forsale; rec.autos; rec.motorcycles; rec.sport.baseball; rec.sport.hockey; sci.crypt; sci.electronics; sci.med; sci.space; soc.religion.christian; talk.politics.misc; talk.politics.guns; talk.politics.mideast; talk.religion.misc.

The data set that we use in this work (20NewsGroups) is a random sub-sample of 1000 elements from the original data set (50 elements from each group). This data set is a 20 class text classification. We have prepared this data set by stemming words according to the Porter Stemming Algorithm [148]. The size of the corpus (the number of different words presented in all the stemmed data set) defines the number of features. In this sub-sample we consider only the words that occur at least 40 times, thus obtaining a corpus of 565 words.

2VowelsPB

The data set 2VowelsPB, represented in Fig. A.1, can be found in [90]. It is a speaker independent, four-class, vowel discrimination problem. The data consists on the first and second formants of the vowels [i], [I], [a] and [A] from 75 speakers (males, females and children). The data forms two pairs of overlapping classes.

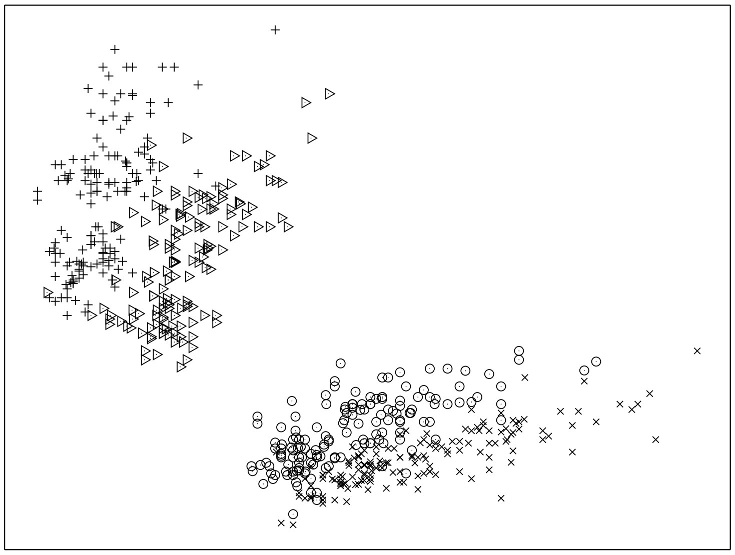


Figure A.1: The 2VowelsPB data set.

Vowels [i] and [I] form one overlapping pair of classes and vowels [a] and [A] form the other pair. The data set has 608 elements.

Features: (all numeric-valued)

1 First formant value.

2 Second formant value.

Class Distribution: 152 elements per class.

Breast Tissue

The Breast Tissue data set can be found in [128]. It contains 106 elements, 9 features and 6 classes.

This data set consists on electrical impedance measurements performed on samples of freshly excised tissue from the breast. The features, computed from the impedance spectrum obtained by measurements taken at seven different frequencies, are:

1 Impedivity (ohm) at zero frequency.
2 Phase angle at 500 KHz.
3 High-frequency slope of phase angle.
4 Impedance distance between spectral ends.
5 Area under spectrum.
6 Area normalized by feature 4.
7 Maximum of the spectrum.
8 Distance between feature 1 and the real part of the maximum frequency point.
9 Length of the spectral curve.

Class Distribution:

Class	# samples
Carcinoma	21
Fibro-adenoma	15
Mastopathy	18
Glandular	16
Connective	14
Adipose	22

CTG

The CTG data set can be found in [128]. It contains 2126 elements, 16 features and 10 classes.

This data set consists on measurements of cardiotocographic (CTG) examinations. Cardiotocography is a popular diagnostic method in Obstetrics, consisting on the analysis and interpretation of the foetal heart rate, the uterine contractions and the foetal movements. In this data set only the measures corresponding to the foetal heart rate signals and computed by an automatic system are included. The classification of the signal patterns was performed by expert obstetricians (following a clinical protocol). From the original data set (22 features) we have discarded 6 features as suggested in [128].

The used features are:

1 Baseline value in b.p.m.
2 Number of accelerations.
3 Number of uterine contractions.
4 Percentage of time with abnormal short term variability.
5 Mean value of short term variability.
6 Percentage of time with abnormal long term variability.
7 Mean value of long term variability.
8 Number of light decelerations.
9 Histogram width (histogram of heart rate in b.p.m.).
10 Low freq. of the histogram.
11 High freq. of the histogram.
12 Number of histogram peaks.
13 Histogram mean.
14 Histogram median.
15 Histogram variance.
16 Histogram tendency.

We used in our experiments the features 2, 3 and 8 has is. However, since they represent a number of occurrences in a certain period of time (different in each pattern) one should transform these features and represent them as the number of occurrences per unit of time (e.g. 10 min).

Class Distribution:

Class	# samples
Calm sleep	384
REM sleep	579
Calm vigilance	53
Active vigilance	81
Shift pattern	72
Accelerative/decelerative pattern	332
Decelerative pattern	252
Largely decelerative pattern	107
Flat-sinusoidal pattern	69
Suspect pattern	197

DHN

The Dutch Handwritten Numerals (DHN) data set can be found in [22]. It consists on 2000 images of handwritten numerals ('0'–'9') extracted from a collection of Dutch utility maps [43]. Each image has 15×16 pixels. A sample of this data set is depicted in Fig. A.2.

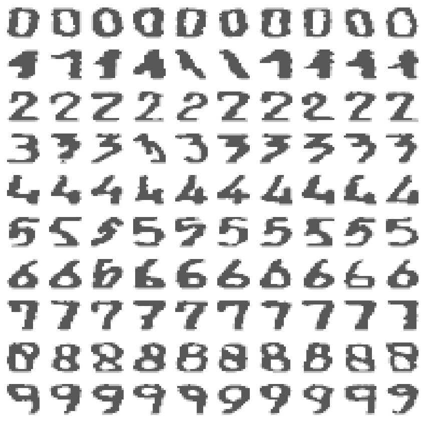


Figure A.2: A sample of the DHN data set.

Each one of the 240 features (15 ×16) corresponds to the pixel gray intensity level.

Class Distribution: 200 elements (handwritten numerals) in each of the 10 classes.

Diabetes

The data set Diabetes (Pima Indians Diabetes) can be found in [22]. It contains 768 elements, 8 features and 2 classes.

Features (all numeric-valued):

1 Number of pregnancies.
2 Plasma glucose concentration at 2 hours in an oral glucose tolerance test.
3 Diastolic blood pressure (mm Hg).
4 Triceps skin fold thickness (mm).
5 2-Hour serum insulin (mu U/ml).
6 Body mass index (weight in kg/(height in m)^2).
7 Diabetes pedigree function.
8 Age (years).

Class Distribution: (class value 1 is interpreted as "tested positive for diabetes")

Class	# samples
0	500
1	268

Ionosphere

The data set Ionosphere can be found in [22]. It contains 351 elements, 34 features and 2 classes.

This is a radar data collected by a system in Goose Bay, Labrador. This system consists on a phased array of 16 high-frequency antennas. The targets were free electrons in the ionosphere. "Good" radar returns are those showing evidence of some type of structure in the ionosphere. "Bad" returns are those that do not. Received signals were processed using an autocorrelation function whose arguments are the time of a pulse and the pulse number. Since there were 17 pulse numbers for the Goose Bay system, with 2 attributes per pulse number, the number of features is 34. We have removed one of the features from the original data set since it has the same value (zero) for all elements.

Class Distribution: (class value 1 is interpreted as "good")

Class	# samples
0	126
1	225

Iris

The data set Iris can be found in [22]. It contains 150 elements, 4 features and 3 classes.

This is the well known Fisher's Iris plants data set, perhaps the best known database to be found in the pattern recognition literature. Fisher's paper [56] is a classic in the field and is referenced frequently to this day. The data set contains 3 classes, where each class refers to a type of Iris plant(Iris Setosa, Iris Versicolour and Iris Virginica). One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.

Features: (all numeric-valued)

1 Sepal length in cm
2 Sepal width in cm
3 Petal length in cm
4 Petal width in cm

Class Distribution: 50 elements in each of 3 the classes.

NCI Microarray

The NCI Microarray data set can be found in [140]. It contains 64 elements, each described by 6830 features, and 12 classes.

Each pattern element corresponds to a human tumor microarray data. NCI is an example of a high-dimensional data set. The data are a 64×6830 matrix of real numbers, each representing an expression measurement for a gene (column) and a sample (row). There are 12 different tumor types, one with just 1 representative and three with 2 representatives. It is also, therefore, a quite unbalanced data set.

Class Distribution:

Class	# samples
Breast	7
CNS	5
Colon	7
K562	2
Leukemia	6
MCF7	2
Melanoma	8
NSCLC	9
Ovarian	6
Prostata	2
Renal	9
Unknown	1

Olive

The data set Olive can be found in [57]. It contains 572 elements, each described by 8 features, and 9 classes.

This data set contains data from eight fatty acid contents of different olive oils from several regions of Italy.

Figure A.3 represents the region of origin of the 9 different kinds of olive oils.

The class distribution is as follows:

Class	# samples
1-North Apulia Calabria	25
2-Calabria	56
3-South Apulia	206
4-Sicily	36
5-Inner Sardinia	65
6-Coastal Sardinia	33
7-East Liguria	50
8-West Liguria	50
9-Umbria	51

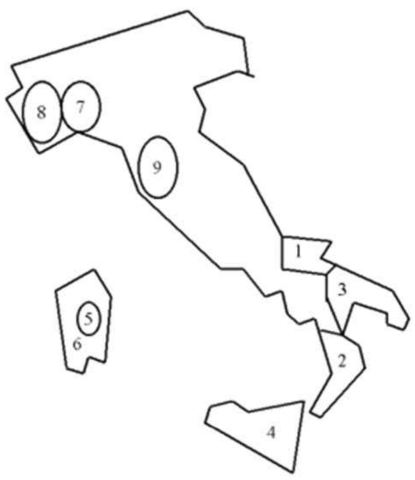


Figure A.3: Italian olive oil samples by nine regions of origin.

Sonar

The data set Sonar can be found in [22]. It contains 208 patterns, each described by 60 features, and 2 classes.

This data set, used by Gorman and Sejnowski in their study of the classification of sonar signals, contains 208 patterns obtained by bouncing sonar signals off a metal cylinder (111) and rocks (97) at various angles and under various conditions. Each pattern corresponds to a vector of 60 real numbers in the range 0.0

to 1.0. Each number represents the energy within a particular frequency band, integrated over a certain period of time.

UBIRIS

The data set UBIRIS is described in [153] and can be downloaded from `http://iris.di.ubi.pt/`. UBIRIS is a data set of eye images used for biometric recognition. In our experiments we used a sample of 12 graytone images with 256 gray levels from this data set, some of which are shown in Fig. A.4. Each image has 60×45 pixels. Each one of the 2700 features (60×45 pixels) corresponds to the pixel gray intensity level. The biometric identification process starts by detecting and isolating the iris with a segmentation algorithm.

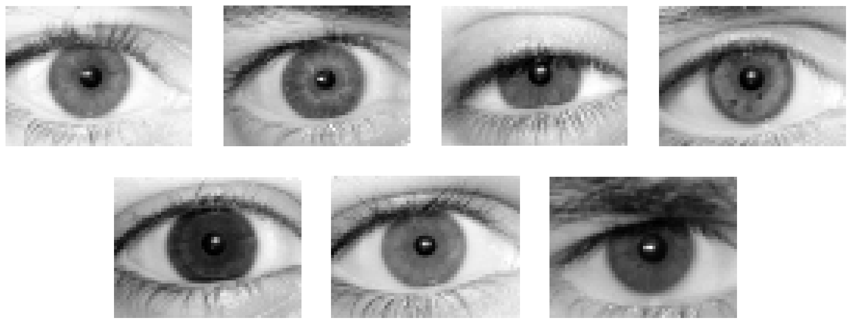


Figure A.4: Sample from UBIRIS data set.

Wdbc

The Wdbc data set is the Wisconsin Breast Cancer data set and can be found in [22]. It contains 569 elements, each described by 30 features, and 2 classes. The two classes, benign and malignant, are linearly separable using all 30 input features.

These 30 features are obtained from 10 original features, computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. These 10 features, that describe characteristics of the cell nuclei present in the image, are computed for each cell nucleus as follows:

1 Radius (mean of distances from center to points on the perimeter).
2 Texture (standard deviation of gray-scale values).
3 Perimeter.
4 Area.
5 Smoothness (local variation in radius lengths).
6 Compactness (perimeter^2 / area - 1.0).
7 Concavity (severity of concave portions of the contour).
8 Concave points (number of concave portions of the contour).
9 Symmetry.
10 Fractal dimension ("coastline approximation" - 1).

The 30 features are obtained by computing the mean, standard error, and "worst" or largest (mean of the three largest values) of the original 10 features, computed for each image.

Class distribution: 357 benign, 212 malignant.

Wine

This data set can be found in [22]. It contains 178 elements, each described by 13 features, and 3 classes.

The data is the result of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 chemical constituents found in each of the three types of wines. The attributes are:

1 Alcohol.
2 Malic acid.
3 Ash.
4 Alkalinity of ash.
5 Magnesium.
6 Total phenols.
7 Flavanoids.
8 Nonflavanoid phenols..
9 Proanthocyanins.
10 Color intensity.
11 Hue.
12 OD280/OD315 of diluted wines.
13 Proline.

The class distribution is as follows:

Class	# samples
Cultivar 1	59
Cultivar 2	71
Cultivar 3	48

Appendix B

An Assessment of Human Clustering on Bi-Dimensional Data

B.1 Introduction

Data clustering performed by humans is characterized by a high variability of solutions for non-trivial data sets. The complexity and subjectivity involved in the clustering process are highly related to the personal experience and sometimes to knowledge about the problem domain. Clustering solutions may depend on a variety of features perceived in the data set. Figure B.1 illustrates some of the features that seem to have a main role in guiding human solutions to clustering. They are as follows:

- **Connectedness** – This is probably the most basic feature leading us to join points into clusters whenever connecting paths are perceived. This feature is valued in the data set of Fig. B.1a when a human "sees" one cluster instead of two.

- **Structuring direction** – This feature leads us to "see" the two arms of the cross in Fig. B.1b instead of only one cluster. Humans are good at perceiving structuring directions in data set graphs, independently of those directions being straight or curved lines.

- **Structuring density** – This feature leads us to "see" two clusters in Fig. B.1c instead of only one.
- **Structuring morphology** – This feature leads us to "see" two clusters in Fig. B.1d instead of only one, deciding differently of the similar Fig. B.1a. The reason is that, contrary to Fig. B.1a, we now identify the bulging out wart of Fig. B.1d with a known form.

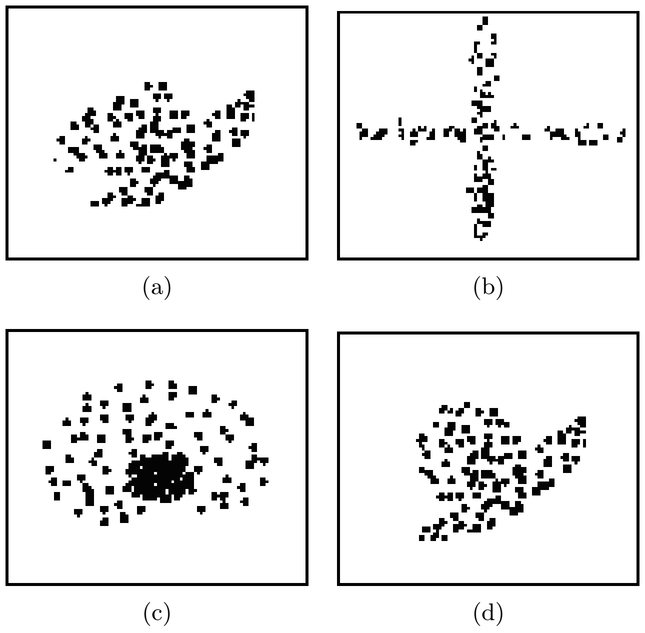


Figure B.1: Clustering features: a) connectedness; b) structuring direction; c) structuring density; d) structuring morphology.

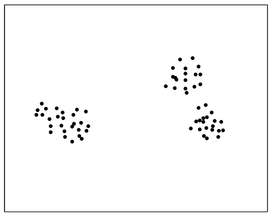
How much influence have these features in the clustering process? How do they interplay? In order to obtain some knowledge about these issues we performed a variety of 2D data clustering experiments involving children and adults. The reason to involve children in clustering experiments is related to the fact that we expected in this way to discriminate (and characterize) basic clustering skills present in children from more advanced skills present in adults. Based on the experimental results we were able to extract a few guidelines on the human approach to data clustering.

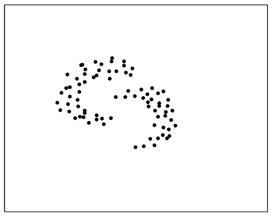
B.2 Clusters Experiments

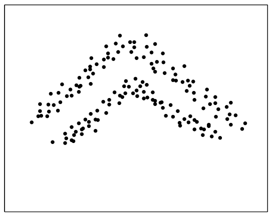
We performed tests involving several individuals (including children) in order to grasp, based on the results, the mental process of data clustering. We made the experiment with 37 individuals, 17 of them children (6-7 years old), 15 adults with no knowledge about clustering and 5 adults with some knowledge of clustering problems. The experiments were performed with the bi-dimensional data sets shown in Figure 2 and Figure 3. All data sets were manually drawn and we tried to create different situations using examples similar to those usually seen in clustering-related works and others created by us.

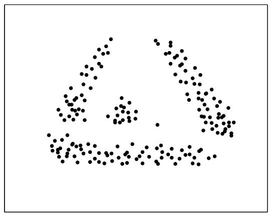
We have presented to the individuals all the data sets in the same order as in Figures B.2 and B.3, and they were asked to circle the possible groups of points in each data set. We haven't given any other explanation or made any comment on the way they should perform the experiment. We just said that in each figure some groups of points could exist, or not, and if they thought they existed they should circle them with a line.

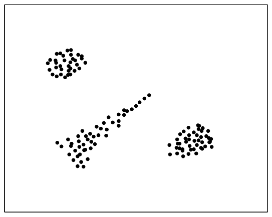
A few similar data sets with small differences among them were deliberately included in order to appreciate how small differences influence the clustering solutions. Examples of such data sets are the pairs (b-f) and (p-aa).

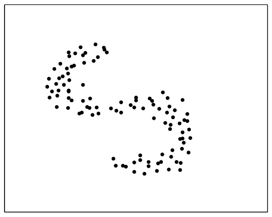

(a) Data set "a".

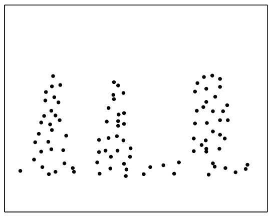

(b) Data set "b".

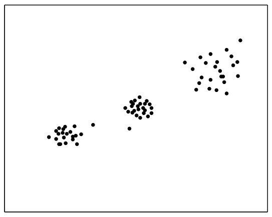

(c) Data set "c".

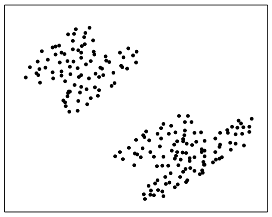

(d) Data set "d".

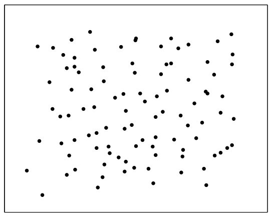

(e) Data set "e".

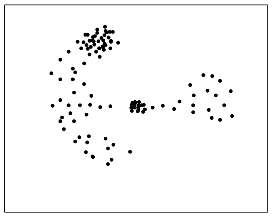

(f) Data set "f".

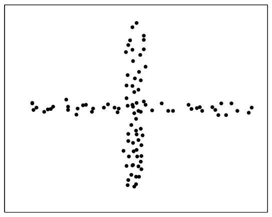

(g) Data set "g".

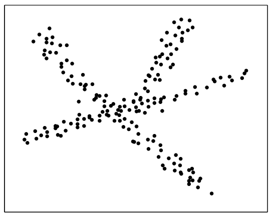

(h) Data set "h".

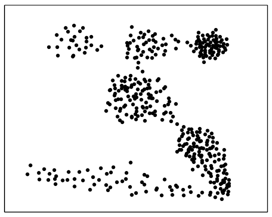

(i) Data set "i".


(j) Data set "j".


(k) Data set "k".


(l) Data set "l".


(m) Data set "m".


(n) Data set "n".

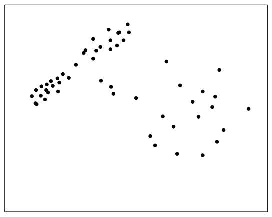
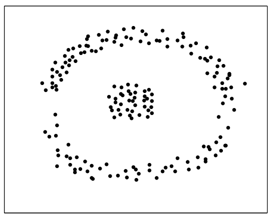
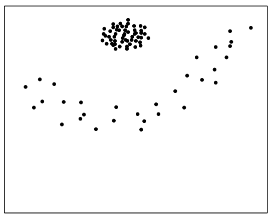
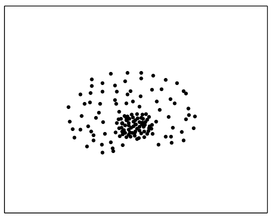
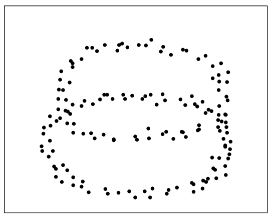

(o) Data set "o".

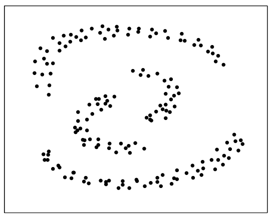
Figure B.2: Data sets I.

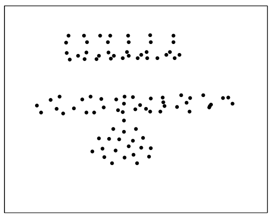

(a) Data set "p".

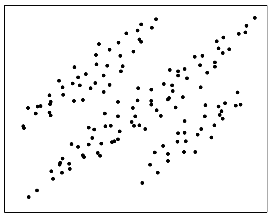

(b) Data set "q".

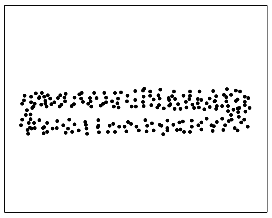

(c) Data set "r".

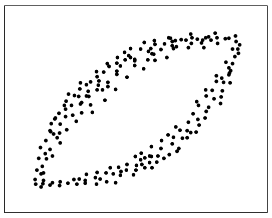

(d) Data set "s".

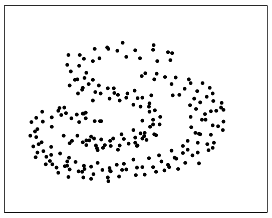

(e) Data set "t".

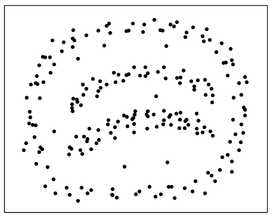

(f) Data set "u".

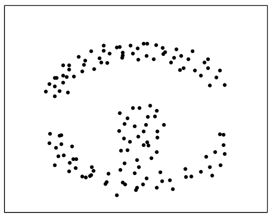

(g) Data set "v".

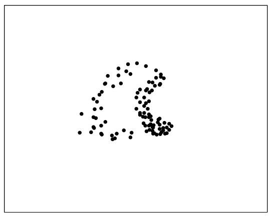

(h) Data set "w".

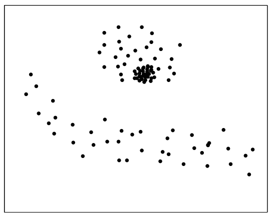

(i) Data set "x".


(j) Data set "y".


(k) Data set "z".


(l) Data set "aa".


(m) Data set "bb".


(n) Data set "cc".

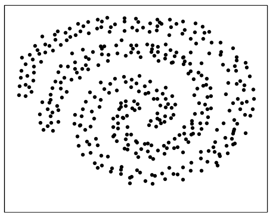

(o) Data set "dd".

Figure B.3: Data sets II.

B.3 Results

B.3.1 Global View

In this section, we present the results of the experiments in a global perspective.

The clustering solutions proposed by the adults and children are summarized in Table B.1. In the labelling of the solutions, we used the label "Others" to designate a group of various solutions different from the most occurring ones, labelled with numerals.

A glance at Table B.1 immediately shows that the solutions proposed by the adults are more consistent, exhibiting fewer solutions for each data set than the ones proposed by the children (6-7 years). Detailed observation of the children solutions revealed that a large percentage of children build clusters based on a small number of points. It seems that they focus on more local regions giving particular attention to small groups. An example of such behavior is shown in Fig. B.4.

During the labelling process we only considered "well-grown" clusters proposed in the solution, disregarding very small clusters (up to 2 points). This often happened with solutions proposed by children. An example of this situation is the one depicted in Fig. B.4a. In this case, we considered the proposed 3-cluster solution like the one shown in Fig. B.20b.

B.3.2 Detailed View

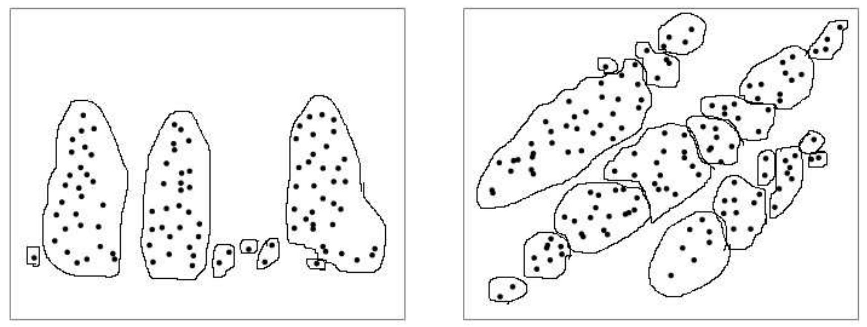
In this section, we present a detailed view of the results together with statistical assessment and some comments.

In order to understand in detail the clustering process, we have divided the data sets into several types. Type A: data sets with well-separated clusters; Type B: data sets with different point densities; Type C: data sets with crossing clusters; Type D: data sets with nested clusters; Type F: data sets with spiral-shaped clusters; Type E: other data sets.

In the next subsections, we take a closer view to each group of data sets and

Table B.1: Experimental results with adults and children. Column "others" refer to several isolated solutions.

	Adults					Children				
	Options					Options				
Data set	1	2	3	4	Others	1	2	3	4	Others
a	19	1				14	1			2
b	20					12				5
c	17	2			1	13	1			3
d	9	9			1	4	10			3
e	19				1	14				3
f	15	5				2	13			2
g	13	6			1	6	7			3
h	20					14				2
i	16	3			1	14				2
j	19				1	9				7
k	2	4	4	5	5	1	4		1	10
l	10	5			5	2	6			9
m	6	4	6		4	9	1	1		6
n	14				6	5				10
o	5	11			4	5	3			8
p	12	7			1	10	3			3
q	20					13				3
r	14	6				9				7
s	4	14			2	4				12
t	19				1	12				4
u	10	7			3	9	2			5
v	19				1	12				4
w	9	5	5		1	2	2	9		1
x	8	6	6			3	3	9		1
y	16	3			1	8	3			5
z	16				4	10				6
aa	11	8			1	8	2			5
bb	16	4				5	6			5
cc	10	8			2	1	11			4
dd	17	3				9	3			4



(a) Example of a clustering solution proposed for data set "g". (b) Example of a clustering solution proposed for data set "v".

Figure B.4: Children usually consider the existence of small clusters.

make some comments about the proposed solutions. We also present analysis of the clustering results with the following statistical tests: χ^2 test for goodness of fit to a postulated distribution; χ^2 test for independence between the Age variable (two categories: adults and children) and Solution variable (categories to be presented in the subsections). The independence test is complemented with Cramer's V measure of association for nominal variables. The level of significance of the tests was set at 5%. The usual conditions of validity of the χ^2 tests were taken into consideration: for one degree of freedom no expected value below 5; for more than one degree of freedom no expected value below 1 and no more than 20% of the expected values below 5. When these conditions were not met the tests were not applied.

B.3.2.1 Type A: Data sets with well-separated clusters

In this subsection, we analyze the group of data sets with well-separated clusters. This group is constituted by the set of data sets a, b, c, d, e, h, i, q, t, v.

For these data sets there is basically a unique solution shown in Fig. B.5 proposed by a large majority of adults and children. Connectedness and sometimes structuring direction (data sets b, c and d) are the main features valued in this unique clustering solution. The results for these data sets are shown in

Table B.2.

Table B.2: Experimental results with adults and children for well-separated clusters.

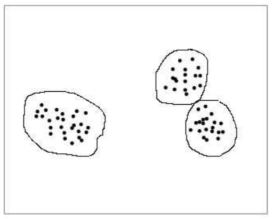
	Adults			Children		
Solutions	1	2	Others	1	2	Others
a	19	1		14	1	2
b	20			12		5
c	17	2	1	13	1	3
d	18		1	14		3
e	19		1	14		3
h	20			14		2
i	16	3	1	14		2
q	20			13		3
t	19		1	12		4
v	19		1	12		4

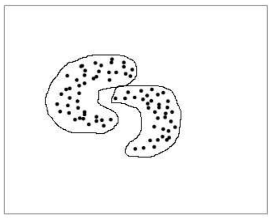
The χ^2 test for independence was performed for a Solution variable with two categories: major solutions; minor solutions. Thus, the 2×2 Table B.3 was used.

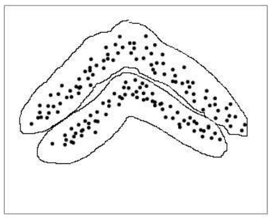
Table B.3: Major and minor solutions for independence test.

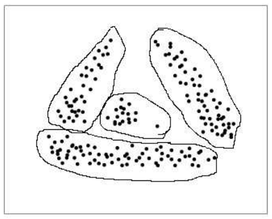
	Adults	Children
Major solutions	187	132
Minor solutions	13	33

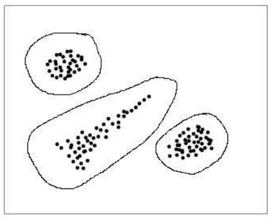
As expected, the independence hypothesis was rejected with $p \approx 0$. The Cramer V of the association is low (V=0.2).

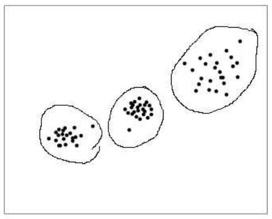

(a) a1.

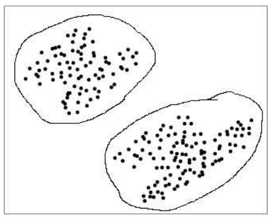

(b) b1.

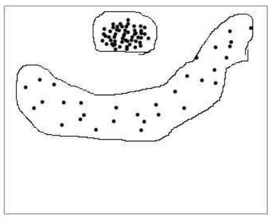

(c) c1.

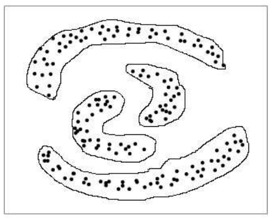

(d) d1.


(e) e1.


(f) h1.


(g) i1.


(h) q1.


(i) t1.

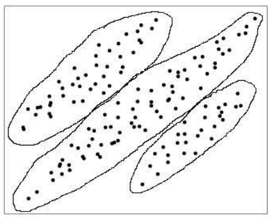

(j) v1.

Figure B.5: The solutions proposed for data sets a, b, c, e, h, i, q, t and v.

B.3.2.2 Type B: Data sets with different point densities

In this subsection we analyze the data sets exhibiting clusters with different point densities. This group is constituted by the set of data sets k, n, o, r, bb, cc. For data set "n" there is basically a unique proposed solution, shown in Fig. B.6.

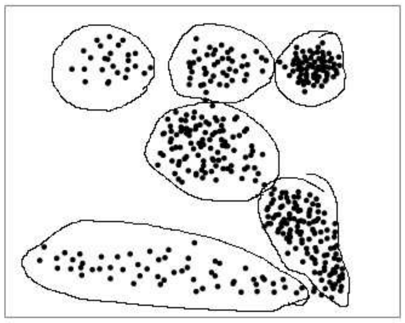


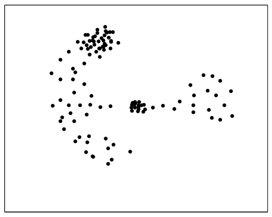
Figure B.6: The solution proposed for data set "n".

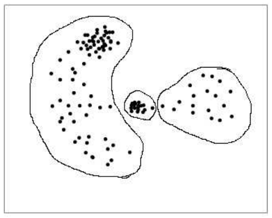
The other Type B data sets are discussed in the following paragraphs.

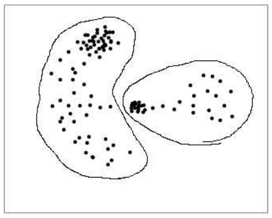
Data set "k" This data set was probably the one with the largest number of different proposed solutions (see Fig.B.7). Apart from the 4 considered solutions (k1 to k4) the adults proposed 5 more different solutions. The reasons for this variability can be attributed to the existence of different density regions and the peculiar structure of the data.

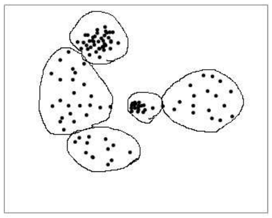
Solution "k4" is the most significant for adults and solution "k2" for children and adults. We think that solution "k3" was suggested by adults based on the symmetry of the data set. We can see that solution "k2" gives more importance to the global structure and that solution "k4" gives relevance to the local structure of the data. Therefore, this data set suggests that children do not value the density feature to the point of sacrificing local connectedness.

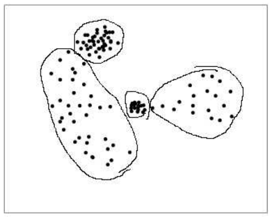
The χ^2 test for goodness of fit led us to accept the uniformity hypothesis (equiprobability of the solutions) for the adults (p=0.66). The χ^2 test for independence, for a Solution variable with two categories ("regular clusters", "other clusters"≡"non-regular clusters"), led us to reject the independence hypothesis (p=0.05). The Cramer V is moderate (V=0.38). The rejection of the independence hypothesis is related to the fact that there is a regular vs. non-regular


(a) k.


(b) k1.


(c) k2.


(d) k3.


(e) k4.

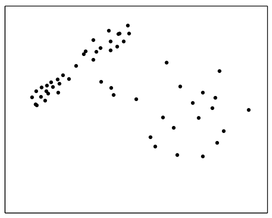
	k1	k2	k3	k4	Oth.
Adults	2	4	4	5	5
Children	1	4		1	10

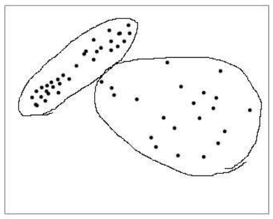
Figure B.7: The solutions proposed for data set "k".

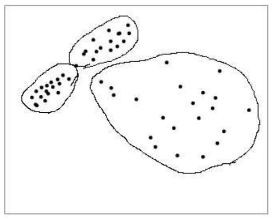
balance for children which is the opposite for adults.

Data set "o" For the data set "o" the solution "o2" was proposed by the majority of the adults (see Fig.B.8).

However, the χ^2 test for goodness of fit led us to accept the uniformity hypothesis for the adults (p=0.13) and for the children (p=0.26). Therefore, the behaviour of adults and children was quite similar in this case. The χ^2 test for independence, for a Solution variable with the three categories as above, led us to reject the independence hypothesis (p=0.056). The Cramer V is moderate (V=0.39). These findings further support the idea of identical behaviour of adults and children when connectedness prevails over slight differences of point


(a) o.


(b) o1.


(c) o2.

	o1	o2	Oth.
Adults	5	11	4
Children	5	3	8

Figure B.8: The solutions proposed for data set "o".

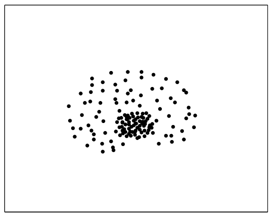
density.

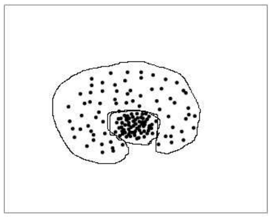
Data set "r" This data set was produced in order to try to perceive the influence of a high density region situated inside a low density region. The performed tests indicate that this high density region is considered by the majority of the individuals, both adults (70%) and children (56%), as a separate cluster.

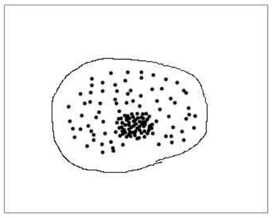
The χ^2 test for goodness of fit led us to reject the uniformity hypothesis for the adults (p=0.05) and accept it for the children (p=0.6) for the "regular"-"non-regular" categories.

Data set "bb" The results show that solution "bb1" was overwhelmingly chosen by the adults. For the children the two solutions "bb1" and "bb2" are almost equally suggested, confirming what we noted previously: children are less inclined to sacrifice connectedness to point density differences.

The χ^2 test for goodness of fit rejects the uniformity hypothesis for the adults and accepts it for the children (p=0.94), confirming the different behaviour of children and adults. The χ^2 test for independence, for a Solution variable

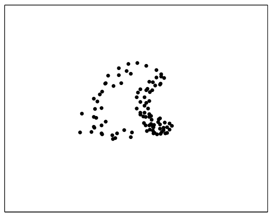

(a) r.

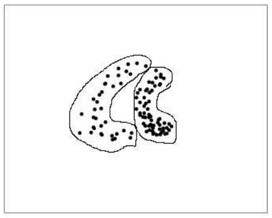

(b) r1.

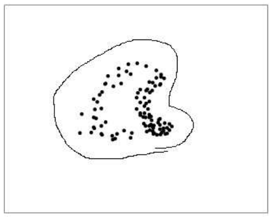

(c) r2.

	r1	r2	Oth.
Adults	14	6	
Children	9		7

Figure B.9: The solutions proposed for data set "r".


(a) bb.


(b) bb1.

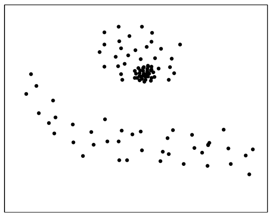

(c) bb2.

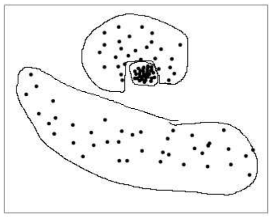
	bb1	bb2	Oth.
Adults	16	4	
Children	5	6	5

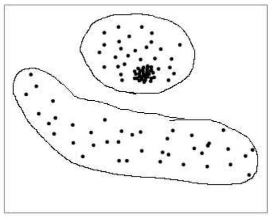
Figure B.10: The solutions proposed for data set "bb".

with the three categories as above, led us to reject the independence hypothesis (p=0.004). The Cramer V is high (V=0.594). This can be attributed to the lack of "Others" in the adult solutions.

Data set "cc" We prepared this data set with the aim of comparing it with data set "r". We have included here a similar region to the one appearing in the data set "r". We were expecting similar solutions in the similar regions. We indeed obtained adult results for this data set very similar to the results for data set "r". For the similar regions, the proposed solutions were also similar. In the children results, this did not happen. Children only considered the existence of the two most evident clusters (solution "cc2"). It seems that many adults were able to decompose the data set on several levels of clusters (something like hierarchical clustering). First, by mentally constructing 2 clusters and secondly, by separating one of them in 2 clusters. Children, on the contrary, tend to value the most prominent feature: connectedness. We think that only a hierarchical mental process is able to justify the differences between adults and children in this data set.


(a) cc.


(b) cc1.


(c) cc2.

	cc1	cc2	Oth.
Adults	10	8	2
Children	1	11	4

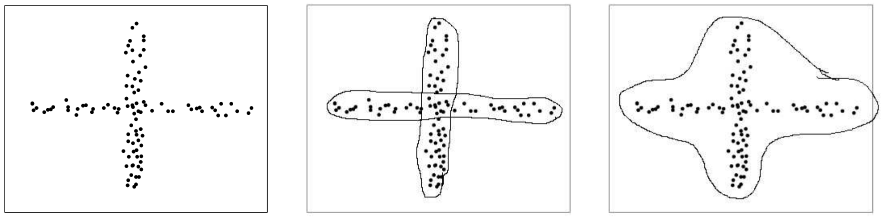
Figure B.11: The solutions proposed for data set "cc".

Disregarding solution "Others", the χ^2 test for goodness of fit accepts the uniformity hypothesis for the adults (p=0.64). The χ^2 test for independence, for a Solution variable with the three categories as above, led us to reject the independence hypothesis (p=0.017). The Cramer V is high (V=0.476).

B.3.2.3 Type C: Data sets with crossing clusters

In this subsection, we analyze the group of data sets with crossing clusters. This group is constituted by the set of data sets l, m, s. In the following subsections, we present and comment the different proposed solutions for these data sets.

Data set "l" In this data set the tests made on adults show that the preferred solution is the one that considers the 2 arms of the cross.



(a) l. (b) l1. (c) l2.

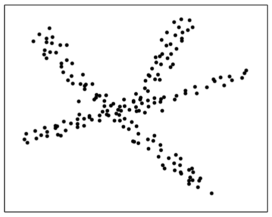
	l1	l2	Oth.
Adults	16	5	5
Children	2	6	9

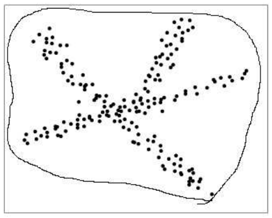
Figure B.12: The solutions proposed for data set "l".

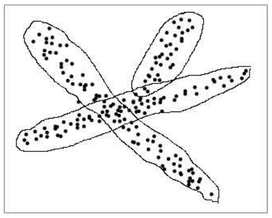
Children prefer to consider the cross as a single cluster. Among the other solutions proposed by children, there were a couple of them considering the division of the cross in 4 clusters, one for each branch. These results suggest that adults are able to trade connectedness by structuring direction, a feature not taken into account by children.

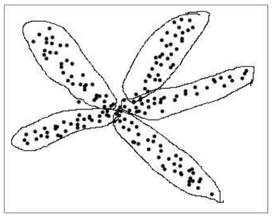
The χ^2 test for goodness of fit accepts the uniformity hypothesis for the adults (p=0.33) and rejects it for the children (p=0.018). The χ^2 test for independence, for a Solution variable with the three categories as above, lead us to reject the independence hypothesis (p=0.04). The Cramer V is high (V=0.415). These results support the different and almost opposite behaviour of adults and children.

Data set "m" This data set was the one where there was more reluctance in clustering the data in more than one cluster. Adults were divided between the existence of only one cluster and the existence of several clusters.


(a) m.


(b) m1.


(c) m2.


(d) m3.

	m1	m2	m3	Oth.
Adults	6	4	6	4
Children	9	1	1	6

Figure B.13: The solutions proposed for data set "m".

Among all the solutions, proposed by adults, the most significative was the one that considered the existence of 5 clusters. This solution for data set "m" is very curious when comparing with the solutions proposed for data set "l". In the latter, adults have not considered the hypothesis of dividing the data set in 4 clusters, one for each branch of the cross; however, in data set "m", maybe influenced by the existence of a branch with no correspondence in the other side of the star, adults have decided to consider each branch as a single cluster. Children consider this to be a single cluster problem, as they do with data set "l". The same comment made previously on connectedness and structuring direction applies here.

The χ^2 test for goodness of fit accepts the uniformity hypothesis for the adults (0.64) and rejects it for the children ($p \approx 0$). χ^2 test for independence, for a Solution variable with two categories - "regular clusters" and "non-regular clusters" -, led us to accept the independence hypothesis ($p=0.3$). The Cramer V is low ($V=0.17$).

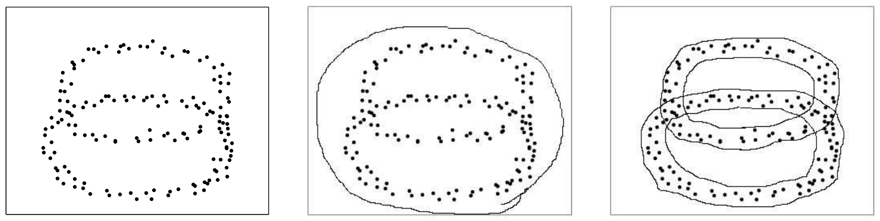
Data set "s" In this data set, almost all adults considered the existence of 2 annular clusters as shown in Figure 14c, however children were unable to do the same. We could see on the solutions proposed by the children that, in some cases, they have tried to represent the two clusters without success due to lack of representation skills. This is a data set where the notion of a structuring direction is of primordial importance, explaining the failure of children in "seeing" solution s2.

The χ^2 test for goodness of fit rejects the uniformity hypothesis for the adults ($p<0.014$).

B.3.2.4 Type D: Data sets with nested clusters

In this subsection we analyze the group of data sets with nested clusters (clusters inside clusters) not considered in previous types. This group is constituted by the set of data sets p, z, aa. For data set "z" there was basically a unique proposed solution, shown in Fig.B.15.

The other Type D data sets are discussed in the following subsections.



(a) s. (b) s1. (c) s2.

	s1	s2	Oth.
Adults	4	14	2
Children	4		12

Figure B.14: The solutions proposed for data set "s".

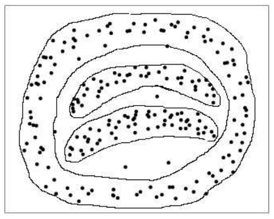
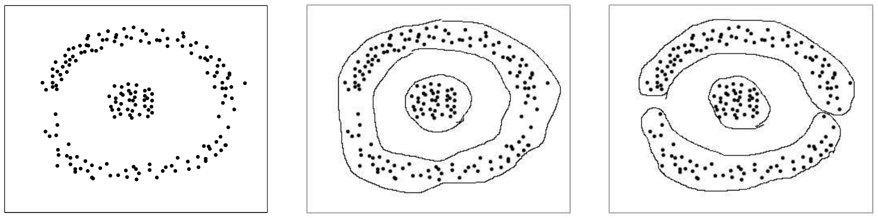


Figure B.15: The solution proposed for data set "z".

Data set "p" Data set "p" has two different proposed solutions. The majority of both children and adults proposed solution "p1".

Disregarding the solution "Others" the χ^2 test for goodness of fit accepts the uniformity hypothesis for the adults (p=0.23) and rejects it for the children (p=0.02. Disregarding the solution "Others" the χ^2 test for independence led us to accept the independence hypothesis (p=0.31). The Cramer V is moderate (V=0.256). Thus, although the majority chose "p1", the behaviour of adults and children is different and, in fact, there is a more than chance-explained (at 5% significance level) majority choice of "p1" for the children. This is a strange finding that at first sight could lead us to think that children valued



(a) p. (b) p1. (c) p2.

	p1	p2	Oth.
Adults	12	7	1
Children	10	3	3

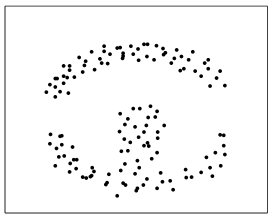
Figure B.16: The solutions proposed for data set "p".

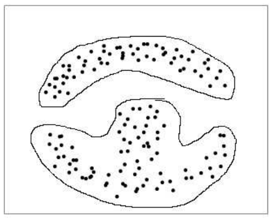
more than adults structuring direction and/or morphology. However, part of the explanation why so many adults chose "p2" may be due to the different point densities of the upper and lower part of the annular cluster; a feature which most of the children didn't see.

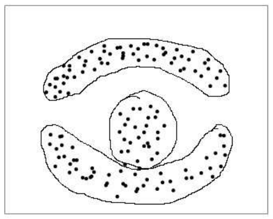
Data set "aa" As we previously mentioned in section 2, we made this data set similar to data set "p" for comparison purposes. We have separated the annular cluster and we have shifted down the circular cluster so that it touches the lower section of the annular cluster. By doing that, we tried to perceive if the individuals consider the circular cluster as a separate cluster.

The results show that this solution ("aa2") was not the preferred solution, especially in the children results, but it almost equals solution "aa1" (only two clusters) in the adult results.

Disregarding the solution "Others", the χ^2 test for goodness of fit accepts the uniformity hypothesis for the adults (p=0.49). Uniformity of the three categories is marginally acceptable for the children (p=0.06). The χ^2 test for independence, for a Solution variable with three categories as above, led us to reject the in-


(a) aa.


(b) aa1.


(c) aa2.

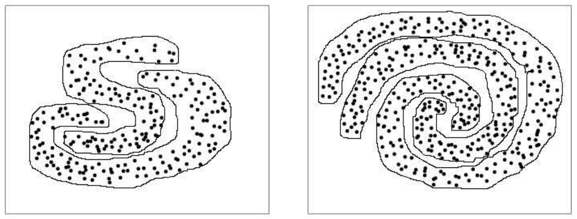
	aa1	aa2	Oth.
Adults	11	8	1
Children	8	2	5

Figure B.17: The solutions proposed for data set "aa".

dependence hypothesis (p=0.038). The Cramer V is high (V=0.42). Therefore, although the "aa2" solution was not the most preferred one by the adults, there is a clear different behaviour of children and adults. Adults were significantly (at 5% significance level) more capable of taking into account the structuring morphological feature present in solution "aa2".

B.3.2.5 Type F: Data sets with spiral-shaped clusters

In this subsection, we analyze the group of data sets with spiral-shaped clusters. This group is constituted by the set of data sets y, dd. For both data sets, the individuals basically considered them as 2-cluster data sets (Fig.B.18), despite the fact that the clusters present a very complex structure compared with the other data sets. We were even surprised by the fact that children also recognized the two spiral clusters presented in data set "dd"; a good illustration of prevalence of a structuring direction over connectedness.



(a) y1. (b) dd1.

Figure B.18: The most significative solutions proposed for data sets with spiral shaped clusters.

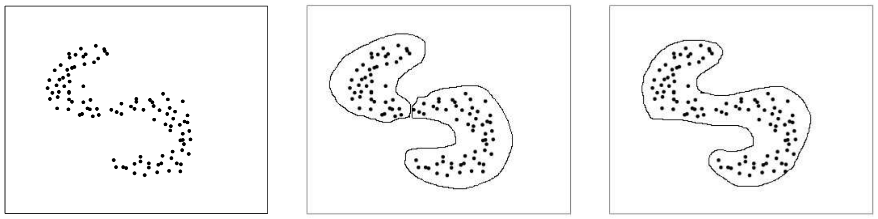
B.3.2.6 Type E: Other data sets

In this subsection we analyze the group of data sets not considered in any of the previous groups. This group is constituted by the set of data sets f, g, j, u, w, x. For data set "j" there is basically a unique proposed solution that considers it as a single cluster. The other data sets are discussed in the following subsections.

Data set "f" The results suggested by adults for data set "f" were, in our opinion, influenced by the previous solutions given to data set "b". We have already mentioned that these two data sets were intentionally produced with a small difference. In this case, the two clusters of data set "b" were shifted to be almost connected (apparently). We think that this fact, and also the low density in the "connecting" region, was responsible for the predominant 2 clusters solution.

However, in the children tests, this fact does not happen. It seems that the solutions that they proposed to data set "b" did not affect the proposed solutions for data set "f", confirming the overwhelming value that children attribute to connectedness.

Disregarding the solution "Others", the χ^2 test for goodness of fit rejects (at 5% significance level) the uniformity hypothesis for both adults and children ($p<0.01$). The χ^2 test for independence, for a Solution variable with the two "regular" categories, led us to reject the independence hypothesis ($p\approx 0$). The



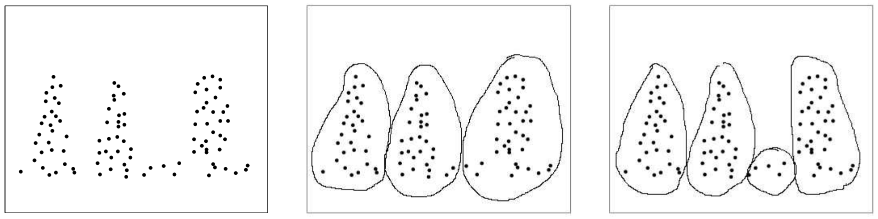
(a) f. (b) f1. (c) f2.

	s1	s2	Oth.
Adults	15	5	
Children	2	13	2

Figure B.19: The solutions proposed for data set "f".

Cramer V is quite high (V=0.61). Thus statistical analysis confirms that adult and children behaviours are different and opposite of each other.

Data set "g" The solutions for this data set are shown in Figure 20.



(a) g. (b) g1. (c) g2.

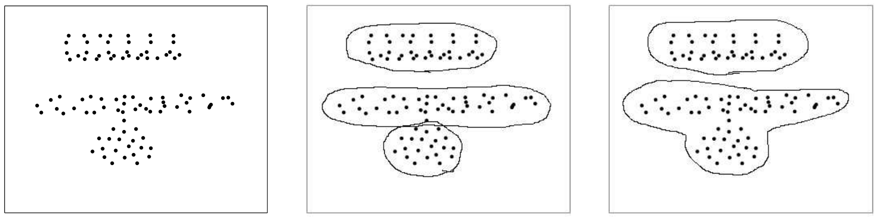
	g1	g2	Oth.
Adults	13	6	1
Children	6	7	3

Figure B.20: The solutions proposed for data set "g".

The majority of the adults have considered this a problem with 3 clusters. The children results are conditioned by the previous mentioned fact that they pay a particular attention to small clusters.

Disregarding solution "Others" the X^2 test for goodness of fit lead us to accept the uniformity hypothesis for both children and adults (here, with p=0.08). The X^2 test for independence, for a Solution variable with two categories (g1, g2), lead us to accept the independence hypothesis (p=0.21). The Cramer V is low (V=0.22).

Data set "u" Data set "u" was designed with the objective of assessing whether differently shaped groups, placed close to each other were considered as one or two clusters. Briefly, to assess the influence of the "structuring morphology" feature.


(a) u. (b) u1. (c) u2.

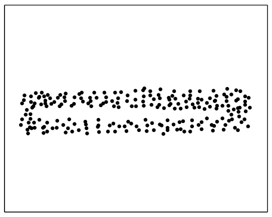
	u1	u2	Oth.
Adults	10	7	3
Children	9	2	5

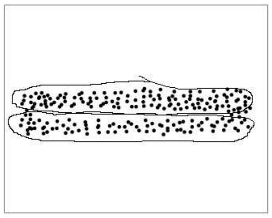
Figure B.21: The solutions proposed for data set "u".

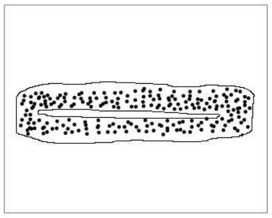
Regarding the solutions proposed by the adults, we see that surprisingly many adults failed to recognize the existence of three clusters, corresponding to separating the circular cluster from the elongated one. Children, on the contrary, seem to exhibit a definite preference by "u1", valuing the "structuring morphology" feature. They overwhelmingly separate the circular cluster from the elongated one.

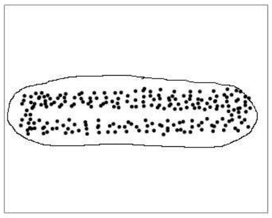
The χ^2 test for goodness of fit marginally accepts the uniformity hypothesis for the adults (p=0.06) and rejects it for the children (p=0.03). The χ^2 test for independence, for a Solution variable with the three categories as above, led us to accept the independence hypothesis (p=0.23). The Cramer V is quite moderate (V=0.29).

Data sets "w" and "x" The solutions proposed for data sets "w" and "x" were very similar. In both cases, there are connections at the ends of the point clouds that influence the different results. Although many two-cluster solutions were proposed, more than 50% of the individuals considered these as one-cluster cases.


(a) w.


(b) w1.

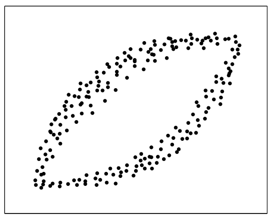

(c) w2.

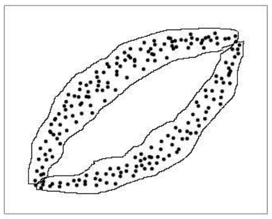

(d) w3.

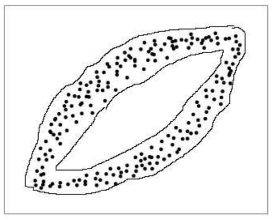
	w1	w2	w3	Oth.
Adults	9	5	5	1
Children	2	2	9	1

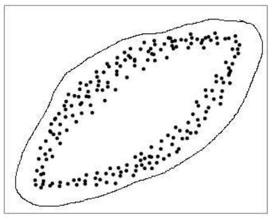
Figure B.22: The solutions proposed for data set "w".

Disregarding the solution "Others", the χ^2 test for goodness of fit accepts the uniformity hypothesis for the adults and for both data sets (p=0.48 and p=0.83 for "w" and "x", respectively). The uniformity hypothesis was only accepted for the children for data set "x". Also, the χ^2 test for independence, for a Solution


(a) x.


(b) x1.


(c) x2.


(d) x3.

	x1	x2	x3	Oth.
Adults	8	6	6	
Children	3	3	9	1

Figure B.23: The solutions proposed for data set "x".

variable with the three regular categories, yielded different results for the data sets: rejection for "w" (p=0.046) and acceptance for "x" (p=0.2).

These findings support the different behaviour of adults and children, with the adults valuing more than children the "structuring morphology" feature.

B.4 Conclusions

Clustering solutions proposed by children are quite different from those proposed by adults. We found for several data sets that the X^2 test for independence (at 5% significance level) either accepted the independence hypothesis (data sets d, g, m, p, u) or rejected it because of adult and children choices in opposite directions (data sets f, k, l, w, x, aa, bb, cc). Thus, we found statistical evidence supporting the thesis of different cluster behaviour of children and adults in those data sets. Children and adults showed a strong agreement of their clustering preferences for the data sets where clustering is mainly based on the connectedness or structuring direction features (well-separated data sets, nested clusters, spiral-shaped clusters).

Children usually "see" small clusters focusing their attention in small regions, leading to solutions with a large number of clusters. They praise overwhelmingly the connectedness feature to the point of sacrificing other ones.

From the analysis of the different types of data sets we draw the following conclusions:

- Connectedness or structuring-direction features are the easiest features to handle, by both adults and children.
- Children are often unable to sacrifice connectedness for other features. This is especially true when data sets exhibit cross-type clusters.
- Point density and morphological structuring are the most difficult clustering features to handle.
- Adults seem capable of performing some sort of hierarchical clustering, using clustering features at different decision levels. This was mainly apparent in the solutions produced for data sets "p", "k", "aa" and "cc".
- A small difference in the data sets, like in the pairs "b"-"f" and "p"-"aa", can lead to very different clustering solutions. This is especially to be expected when the point density and morphological structuring features come into play.

Bibliography

[1] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski. A learning algorithm for boltzmann machines. *Cognitive Science*, 9:147–169, 1985.

[2] I. Ahmad and P.-E. Lin. A nonparametric estimation of the entropy for absolutely continuous distributions. *IEEE Trans. on Information Theory*, 22(3):372–375, 1976.

[3] N. A. Ahmed and D. V. Gokhale. Entropy expressions and their estimators for multivariate distributions. *IEEE Trans. on Information Theory*, 35(3):688–692, 1989.

[4] L. A. Alexandre, A. C. Campilho, and M. Kamel. A probabilistic model for the cooperative modular neural network. In *Iberian Conference on Pattern Recognition and Image Analysis*, LNCS 2652, pages 11–18. Springer-Verlag, 2003.

[5] L. A. Alexandre, A. C. Campilho, and M. Kamel. Bounds for the average generalization error of the mixture of experts neural network. In *5th Int. Workshop on Statistical Techniques in Pattern Recognition*, LNCS 3138, pages 618–625. Springer-Verlag, 2004.

[6] E. L. Allwein, R. E. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. In *Int. Conference on Machine Learning*, pages 9–16. Morgan Kaufmann, 2000.

[7] L. B. Almeida. A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. In *IEEE First Conference on Neural Networks*, volume 1, pages 609–618, 1987.

[8] S. Amari. Characteristics of random nets of analog neuron-like elements. *IEEE Trans. on Systems, Man and Cibernetics*, 2(5):643–657, 1972.

[9] S. Amari, A. Cichocki, and H. Yang. A new learning algorithm for blind signal separation. In *Advances in Neural Information Processing System*, volume 8. MIT Press, 1996.

[10] A. Antos and I. Kontoyiannis. Estimating the entropy of discrete distributions. In *IEEE Int. Symposium on Information Theory*, 2001.

[11] G. Auda and M. Kamel. Modular neural network classifiers: A comparative study. *Intel. Robotic Systems*, 21:117–129, 1998.

[12] G. P. Basharin. On statistical estimate for the entropy of a sequence of independent random variables. *Theory of Probability and its Applications*, 4:333–336, 1956.

[13] R. Battiti. Accelerated backpropagation learning: Two optimization methods. *Complex Systems*, 3:331–342, 1989.

[14] T. Batu, S. Dasgupta, R. Kumar, and R. Rubinfeld. The complexity of approximating the entropy. *SIAM Journal on Computing*, 35(1):132–150, 2005.

[15] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36(1-2):105–139, 1999.

[16] R. T. Bayes. An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London*, 53:370–418, 1763.

[17] V. Beiu and T. D. Pauw. Tight bounds on the size of neural networks for classification problems. In *Int. Work-Conference on Artificial and Natural Neural Networks*, LNCS 1240, pages 743–752. Springer Verlag, 1997.

[18] A. Bell and T. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159, 1995.

[19] J.-F. Bercher and C. Vignat. Estimating the entropy of a signal with applications. *IEEE Trans. on Signal Processing*, 48(6):1687–1694, 2000.

[20] P. Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA, 2002.

[21] C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, N.Y., 1996.

[22] C. Blake, E. Keogh, and C. Merz. UCI repository of machine learning databases. http://www.ics.uci.edu/˜mlearn/MLRepository.html, 1998.

[23] R. Boscolo, H. Pan, and V. P. Roychowdhury. Non-parametric ica. In *Proc. of the Int. Conference on Independent Component Analysis and Blind Signal Separation*, pages 13–18, 2001.

[24] A. W. Bowman and A. Azzalini. *Applied Smooting Techniques for Data Analysis.* Oxford University Press, 1997.

[25] C. Cachin. Smooth entropy and renyi entropy. In *Advances in Cryptology: Eurocrypt'97*, LNCS 1233, pages 193–208. Springer-Verlag, 1997.

[26] L. S. Camargo and T. Yoneyama. Specification of training sets and the number of hidden neurons for multilayer perceptrons. *Neural Computation*, 13(12):2673–2680, 2001.

[27] L. W. Chan and F. Fallside. An adaptive training algorithm for backpropagation networks. *Computer Speech and Language*, 2:205–218, 1987.

[28] T. N. Chatterjee. On the application of information theory to the optimum state-space reconstruction of the short-term solar radio flux (10.7cm), and its prediction via a neural network. *Monthly Notices of the Royal Astronomical Society*, 323:101–108, 2001.

[29] C. H. Cheng, A. W. Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *Int. Conference on Knowledge Discovery and Data Mining*, 1999.

[30] Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(8):790–799, 1995.

[31] F. R. K. Chung. *Spectral Graph Theory*, volume 92. American Mathematical Society, Providence, RI, 1997.

[32] D. Comaniciu and P. Meer. Mean shift analysis and applications. In *IEEE Int. Conference on Computer Vision*, pages 1197–1203, 1999.

[33] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analisys. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[34] J. C. Correa. A new estimator of entropy. *Communications in Statistics Theory and Methods*, 24(10):2439–2449, 1995.

[35] T. M. Cover and J. A. Thomas. *Elements of Information Theory.* John Wiley & Sons, 1991.

[36] E. M. F. Curado and C. Tsallis. Generalized statistical mechanics: connection with thermodynamics. *Journal of Physics A: Mathematical and General*, 24(2):L69–L72, 1991.

[37] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, 1989.

[38] T. G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, 2000.

[39] C. Ding, X. He, H. Zha, M. Gu, and H. Simon. A min-max cut algorithm for graph partitioning and data clustering. In *Int. Conference on Data Mining*, pages 107–114, 2001.

[40] Y. G. Dmitriev and F. P. Tarasenko. On the estimation of functionals of the probability density and its derivatives. *Theory of Probability and its Applications*, 18(3):628–633, 1974.

[41] S. Draghici and V. Beiu. Entropy based comparison of neural networks for classification. In *Italian Conference on Neural Networks*, 1997.

[42] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification.* John Wiley, 2001.

[43] R. P. Duin. Dutch handwritten numerals. http://www.ph.tn.tudelft.nl /˜duin.

[44] A. Ennaji, A. Ribert, and Y. Lecourtier. From data topology to a modular classifier. *Int. Journal on Document Analysis and Recognition*, 6(1):1–9, 2003.

[45] D. Erdogmus. *Information Theoretic Learning: Renyi's Entropy and its Applications to Adaptive System Training.* PhD thesis, University of Florida, 2002.

[46] D. Erdogmus and J. Príncipe. An error-entropy minimization algorithm for supervised training of nonlinear adaptive systems. *IEEE Trans. On Signal Processing*, 50(7):1780–1786, 2002.

[47] D. Erdogmus and J. C. Príncipe. Entropy minimization algorithm for multilayer perceptrons. In *Int. Joint Conference on Neural Networks*, volume 4, pages 3003–3008, 2001.

[48] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In A. Press, editor, *2nd Int. Conference on Knowledge Discovery and Data Mining*, pages 226–231, 1996.

[49] R. M. Fano. Class notes for transmission of information. MIT, Cambridge, MA, 1952. Course 6.574.

[50] B. Ferguson, R. Ghosh, and J. Yearwood. An experiment in task decomposition and ensembling for a modular artificial neural network. In *LNCS*, volume 3029, pages 97–106. Springer, 2004.

[51] S. E. Fhalman. Faster-learning variations on back-propagation: An empirical study. In D. S. Touretzky, G. E. Hinton, and T. J. Sejnowski, editors, *Connectionist Models Summer School*, pages 38–51. Morgan Kaufmann, 1988.

[52] M. Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal*, 25(100):619–633, 1975.

[53] B. Fischer and J. M. Buhmann. Path-based clustering for grouping of smooth curves and texture segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 25(4):513–518, 2003.

[54] B. Fischer, T. Zöller, and J. M. Buhmann. Path based pairwise data clustering with application to texture segmentation. In *Int. Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, LNCS 2134, pages 235–250. Springer-Verlag, 2001.

[55] J. Fisher. *Nonlinear Extensions to the Minimum Average Correlation Energy Filter*. PhD thesis, University of Florida, 1997.

[56] R. Fisher. The use of multiple measurements in taxonomic problems. *Annual Eugenics*, 7(II):179–188, 1936.

[57] M. Forina and C. Armanino. Eigenvector projection and simplified non-linear mapping of fatty acid content of italian olive oils. *Ann. Chim. (Rome)*, 72:127–155, 1981.

[58] K. Fukunaga and L. D. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Trans. in Information Theory*, 21:32–40, 1975.

[59] P. L. Galindo, J. Pizarro-Junquera, and E. Guerrero. Multiple comparison procedures for determining the optimal complexity of a model. In *Advances in Pattern Recognition: Joint IAPR Int. Workshops*, LNCS 1876, pages 796–805. Springer-Verlag, 2000.

[60] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58, 1992.

[61] Y. S. Goh and E. C. Tan. Pruning neural networks during training by backpropagation. In *IEEE Region 10's Int. Conference*, volume 2, pages 805–808, 1994.

[62] E. Gokcay. *A New Clustering Algorithm for Segmentation of Magnetic Resonance Images*. PhD thesis, University of Florida, 2000.

[63] E. Gokcay and J. C. Príncipe. Information theoretic clustering. *IEEE Trans. on Pattern Analysis and Machine Learning*, 24(2):158–171, 2002.

[64] D. V. Gokhale. Maximum entropy characterization of some distributions. In K. Patil and O. Eds., editors, *Statistical Distributions in Scientific Work*, volume 3, pages 299–304, 1975.

[65] P. Grassberger. Entropy estimates from insufficient samplings. *ArXiv Physics e-prints, physics/0307138*, 2003.

[66] S. Grossberg. *Studies of Mind and Brain.* Boston, MA: Reidel, 1982.

[67] S. Grossberg. *The Adaptive Brain I: Cognition, Learning, Reinforcement, and Rythm, and The Adaptive Brain II: Vision, Speech, Language, and Motor Control.* Elsevier, 1986.

[68] S. Guha, R. Rastogi, and K. Shim. CURE: an efficient clustering algorithm for large databases. In *Int. Conference on Management of Data*, pages 73–84, 1998.

[69] S. Guha, R. Rastogi, and K. Shim. ROCK: A robust clustering algorithm for categorical attributes. *Information Systems*, 25(5):345–366, 2000.

[70] L. Györfi and E. C. van der Meulen. Density-free convergence properties of various estimators of entropy. *Computational Statistics & Data Analysis*, 5(4):425–436, 1987.

[71] L. Györfi and E. C. van der Meulen. An entropy estimate based on a kernel density estimation. *Limit Theorems in Probability and Statistics*, 57:229–240, 1990.

[72] L. Györfi and E. C. van der Meulen. On the nonparametric estimation of entropy functional. In G. Roussas, editor, *Nonparametric Functional Estimation and Related Topics*, pages 81–95. Kluwer Academic Publisher, 1991.

[73] P. Hall and S. Morton. On the estimation of entropy. *Annals of the Institute of Statistical Mathematics*, 45:69–88, 1993.

[74] R. V. Hartley. Transmission of information. *Bell System Technical Journal*, 7:535–563, 1928.

[75] E. Hartuv, A. Schmitt, J. Lange, S. Meier-Ewert, H. Lehrachs, and R. Shamir. An algorithm for clustering cDNAs for gene expression analysis. In *Annual Conference on Research in Computational Molecular Biology*, pages 188–197, 1999.

[76] E. Hartuv and R. Shamir. A clustering algorithm based on graph connectivity. *Information Processing Letters*, 76(4-6):175–181, 2000.

[77] H. Haselsteiner and J. Príncipe. Supervised learning without numerical targets - an information theoretic approach. In *European Signal Processing Conference*, volume n/a, page n/a, 2000.

[78] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning.* Springer Verlag, 2001.

[79] J. Havrda and F. Charvat. Quantification methods of classification processes: Concept of structural entropy. *Kybernetica*, 3:30–35, 1967.

[80] S. Haykin. *Neural Networks: A Comprehensive Foundation.* Prentice Hall, New Jersey, 2nd edition, 1999.

[81] A. O. Hero, B. Ma, O. J. Michel, and J. Gorman. Applications of entropic spanning graphs. *IEEE Signal Processing Magazine*, 19(5):85–95, 2002.

[82] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. In *National Academy of Sciences of the U.S.A.*, volume 79, pages 2554–2558, 1982.

[83] J. J. Hopfield and D. W. Tank. Computing with neural circuits. *Science*, 233:625–633, 1986.

[84] H. C. Hsin, C. C. Li, M. Sun, and R. J. Sclabassi. An adaptive training algorithm for back-propagation neural networks. *IEEE Trans. on Systems, Man, and Cybernetics*, 25:512–514, 1995.

[85] G.-B. Huang and H. A. Babri. Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions. *IEEE Trans. on Neural Networks*, 9(1):224–229, 1998.

[86] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985.

[87] H. Hutcheson and L. R. Shenton. Some moments of an estimate of shannon's measure of information. *Communications in Statistics*, 3:89–94, 1974.

[88] A. V. Ivanov and A. Rozhkova. Proprieties of the statistical estimate of the entropy of a random vector with a probability density. *Problems of Information Transmission*, 17:171–178, 1981.

[89] M. S. Iyer and R. R. Rhinehart. A method to determine the required number of neural-network training repetitions. *IEEE Trans. on Neural Networks*, 10(2):427–432, 1999.

[90] R. Jacobs, M. Jordan, S. Nowlan, and G. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

[91] R. Jacobs, F. Peng, and M. Tanner. A bayesian approach to model selection in hierarchical mixtures-of-experts architectures. *Neural Networks*, 10(2):231–241, 1997.

[92] R. A. Jacobs. Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1:295–307, 1988.

[93] A. Jain, A. Topchy, M. Law, and J. Buhmann. Landscape of clustering algorithms. In *17th Int. Conference on Pattern Recognition*, volume 1, pages 260–263, 2004.

[94] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data.* Prentice Hall Int., 1988.

[95] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[96] R. Jenssen, T. Eltoft, and J. Príncipe. Information theoretic spectral clustering. In *Int. Joint Conference on Neural Networks*, pages 111–116, 2004.

[97] R. Jenssen, I. Hild, K.E., D. Erdogmus, J. Príncipe, and T. Eltoft. Clustering using rényi's entropy. In *Int. Joint Conference on Neural Networks*, pages 523–528, 2003.

[98] P. Jizba and T. Arimitsu. The world according to rényi: thermodynamics of multifractal systems. *Annals of Physics*, 312:17–59, 2004.

[99] E. L. Johnson, A. Mehrotra, and G. L. Nemhauser. Min-cut clustering. *Mathematical Programming*, 62:133–151, 1993.

[100] M. Jordan and R. Jacobs. Hierarchical mixture of experts and the EM algorithm. *Neural Computation*, 6:181–214, 1994.

[101] M. I. Jordan. Supervised learning and systems with excess degrees of freedom. Technical Report 88-27, COINS, MIT, 1988.

[102] S. D. Kamvar, D. Klein, and C. D. Manning. Interpreting and extending classical agglomerative clustering algorithms using a model-based approach. In *Int. Conference on Machine Learning*, pages 283–290. Morgan Kaufmann, 2002.

[103] R. Kannan, S. Vempala, and A. Vetta. On clusterings: Good, bad, and spectral. In *Annual Symposium on the Foundation of Computer Science*, pages 367–380, 2000.

[104] J. N. Kapur. *Measures of Information and Their Applications.* John Wiley & Sons, New York, 1994.

[105] E. D. Karnin. A simple procedure for pruning back-propagation trained neural networks. *IEEE Trans. on Neural Networks*, 1(2):239–242, 1990.

[106] G. Karypis. *Cluto: A Clustering Toolkit*. University of Minnesota, Department of Computer Science, 2003.

[107] G. Karypis. Cluto: Software package for clustering high-dimensional datasets, 2003. Version 2.1.1.

[108] G. Karypis, E.-H. S. Han, and V. Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *IEEE Computer*, 32(8):68–75, 1999.

[109] G. Karypis and V. Kumar. Multilevel algorithms for multi-constraint graph partitioning. Technical Report 98-019, University of Minnesota, Department of Computer Science, 1998.

[110] L. Kaufman and P. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley and Sons, New York, 1990.

[111] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas. On combining classifiers. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(3):226–239, 1998.

[112] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[113] A. Kolmogorov. Sur la notion de la moyenne. *Atti della R. Accademia Nazionale dei Lincei*, 12:388–391, 1930.

[114] I. Kontoyiannis, P. H. Algoet, Y. M. Suhov, and A. J. Wyner. Nonparametric entropy estimation for stationary processes and random fields, with application to english text. *IEEE Trans. on Information Theory*, 44(3):1319–1327, 1998.

[115] L. F. Kozachenko and N. N. Leonenko. Sample estimate of the entropy of a random vector. *Problems of Information Transmission*, 23(2):95–101, 1987.

[116] S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. of Math. Stat.*, 22:79–86, 1951.

[117] A. V. Lazo and P. N. Rathie. On the entropy of continuous probability distributions. *IEEE Trans. on Information Theory*, 24:120–122, 1978.

[118] T.-W. Lee, M. Girolami, A. J. Bell, and T. J. Sejnowski. A unifying information-theoretic framework for independent component analysis. *Computers and Mathematics with Applications*, 39(11):1–21, 2000.

[119] Y. Lee and S. Choi. Minimum entropy, k-means, spectral clustering. In *IEEE Int. Joint Conference on Neural Networks*, volume 1, pages 117–122, 2004.

[120] Y. Lee and S. Choi. Maximum within-cluster association. *Pattern Recognition Letters*, 26(10):1412–1422, July 2005.

[121] H. Li, K. Zhang, and T. Jiang. Minimum entropy clustering and applications to gene expression analysis. In *IEEE Computational Systems Bioinformatics Conference*, pages 142–151, 2004.

[122] R. Linsker. Self-organization in a perceptual network. *IEEE Computer*, 21:105–117, 1988.

[123] R. P. Lippmann. An introduction to computing with neural nets. *IEEE Acoustics, Speech, and Signal Processing Magazine*, 1987.

[124] W. A. Little and G. L. Shaw. A statistical theory of short and long term memory. *Bihavioral Biology*, 14:115–133, 1975.

[125] G. D. Magoulas, M. N. Vrahatis, and G. S. Androulakis. Effective backpropagation with variable stepsize. *Neural Networks*, 10:69–82, 1997.

[126] G. D. Magoulas, M. N. Vrahatis, and G. S. Androulakis. Improving the convergence of the backpropagation algorithm using learning rate adaptation methods. *Neural Computation*, 11(7):1769–1796, 1999.

[127] J. Marques de Sá. *Pattern Recognition: Concepts, Methods ans Applications.* Springer-Verlag, 2001.

[128] J. Marques de Sá. *Applied statistics using SPSS, STATISTICA and MATLAB.* Springer, 2003.

[129] J. Marques de Sá. Nonparametric density and entropy estimation (with a focus on the parzen window method) - tutorial text. http://gnomo.fe.up.pt/~nnig/papers/DEE.pdf, 2006.

[130] D. Matula. Cluster analysis via graph theoretic techniques. In R.C.Mullin, K.B.Reid, and D.P.Roselle, editors, *Proc. Luisiana Conference on Combinatorics, Graph Theory and Computing*, pages 199–212, 1970.

[131] D. Matula. k-components, clusters and slicings in graphs. *SIAM J. Appl. Math.*, 22(3):459–480, 1972.

[132] W. S. McCulloch and W. Pitts. A logical calculus of the ideias imminent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[133] M. Meila and J. Shi. A random walks view of spectral segmentation. In *Eighth Int. Workshop on Artificial Intelligence and Statistics*, 2001.

[134] C. E. Metz. Basic principles of ROC analysis. *Seminars in Nuclear Medicine*, 8(4):283–298, 1978.

[135] M. L. Minsky and S. A. Papert. *Perceptrons.* MIT Press, 1969.

[136] A. Mokkadem. Estimation of the entropy and information of absolutely continuous random variables. *IEEE Trans. on Information Theory*, 35(1):193–196, 1989.

[137] R. Morejon. *An Information-Theoretic Approach to Sonar Automatic Target Recognition.* PhD thesis, University of Florida, 2003.

[138] R. A. Morejon and J. C. Principe. Advanced search algorithms for information-theoretic learning with kernel-based estimators. *IEEE Trans. on Neural Networks*, 15(4):874–884, 2004.

[139] M. Nagumo. Über eine klasse von mittelwerten. *Japanese Journal of Mathematics*, 7(71-79), 1930.

[140] NCI60. Stanford NCI60 cancer microarray project. http://genome-www.stanford.edu/nci60/, 2000.

[141] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, volume 14, 2001.

[142] G. S. Ng, A. Wahab, and D. Shi. Entropy learning and relevance criteria for neural network pruning. *Int. Journal of Neural Systems*, 13(5):291–305, 2003.

[143] H. Nyquist. Certain factors affecting telegraph speed. *Bell System Technical Journal*, 3:332–333, 1924.

[144] L. Paninski. Estimation of entropy and mutual information. *Neural Computation*, 15:1191–1253, 2003.

[145] L. Paninski. Estimating entropy on m bins given fewer than m samples. *IEEE Trans. on Information Theory*, 50(9):2200–2203, 2004.

[146] E. Parzen. On the estimation of a probability density function and mode. *Annals of Mathematical Statistics*, 33:1065–1076, 1962.

[147] A. Pelagotti and V. Piuri. Entropic analysis and incremental synthesis of multilayered feedforward neural networks. *Int. Journal of Neural Systems*, 8(5-6):647–659, 1997.

[148] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[149] J. Príncipe and D. Erdogmus. From adaptive linear to information filtering. In IEEE, editor, *Symposium on Adaptive Systems for Signal Processing, Communications, and Control*, pages 99–104, 2000.

[150] J. Príncipe, J. W. Fisher, and D. Xu. *Unsupervised Adaptive Filtering*, chapter Information Theoretic Learning, pages 265–319. John Wiley & Sons, 2000.

[151] J. Príncipe and D. Xu. An introduction to information theoretic learning. In *Int. Joint Conference on Neural Networks*, pages 1783–1787, 1999.

[152] J. Príncipe, D. Xu, Q. Zhao, and J. W. Fisher. Learning from examples with information theoretic criteria. *Journal of VLSI Signal Processing Systems*, 26(1-2):61–77, 2000.

[153] H. Proença and L. A. Alexandre. UBIRIS: A noisy iris image database. In *Int. Conference on Image Analysis and Processing*, volume 1, pages 970–977, 2005.

[154] A. Rényi. On measures of entropy and information. *Selected Papers of Alfred Rényi*, 2:565–580, 1976.

[155] A. Rényi. Some fundamental questions of information theory. *Selected Papers of Alfred Rényi*, 2:526–552, 1976.

[156] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *IEEE Int. Conference on Neural Networks*, pages 586–591, 1993.

[157] I. Rivals and L. Personnaz. A statistical procedure for determining the optimal number of hidden neurons of a neural model. In *Symposium on Neural Computation*, 2000.

[158] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:368–408, 1958.

[159] F. Rosenblatt. *Principles of Neurodynamics.* Spartan Books, New York., 1962.

[160] M. Rosenblatt. Remarks on some nonparametric estimates of a density function. *Annals of Mathematical Statistics*, 27:832–835, 1956.

[161] D. Rumelhart, G. Hinton, and R. Williams. Learning representations by back-propagation errors. *Nature*, 323:533–536, 1986.

[162] G. Sanguinetti, J. Laidler, and N. D. Lawrence. Automatic determination of the number of clusters using spectral algorithms. In *Int. Workshop on Machine Learning for Signal Processing*, pages 55–60, 2005.

[163] I. Santamaria, D. Erdogmus, and J. Principe. Entropy minimization for supervised digital communications channel equalization. *IEEE Trans. on Signal Processing*, 50(5):1184–1192, 2002.

[164] J. M. Santos, L. A. Alexandre, and J. Marques de Sá. The Error Entropy Minimization Algorithm for Neural Network Classification. In A. Lofti, editor, *Int. Conference on Recent Advances in Soft Computing*, pages 92–97, 2004.

[165] J. M. Santos, L. A. Alexandre, and J. Marques de Sá. Modular neural network task decomposition via entropic clustering. In *Int. Conference on Intelligent Systems Design and Applications*, pages 62–67. IEEE Computer Society Press, 2006.

[166] J. M. Santos, J. Marques de Sá, and L. A. Alexandre. Batch-sequential algorithm for neural networks trained with entropic criteria. In *Int. Conference on Artificial Neural Networks*, LNCS 3697, pages 91–96. Springer Verlag, 2005.

[167] J. M. Santos, J. Marques de Sá, and L. A. Alexandre. Neural Networks Trained with the EEM Algorithm: Tuning the Smoothing Parameter. *WSEAS Transactions on Systems*, 4(4):295–300, 2005.

[168] J. M. Santos, J. Marques de Sá, and L. A. Alexandre. LEGClust - a clustering algorithm based on layered entropic subgraphs. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2006. Accepted for publication.

[169] J. M. Santos, J. Marques de Sá, L. A. Alexandre, and F. Sereno. Optimization of the Error Entropy Minimization Algorithm for Neural Network Classification. In C.H.Dagli, A. L. Buczak, D. L. Enke, M. J. Embrechts, and O. Ersoy, editors, *Intelligent Engineering Systems Through Artificial Neural Networks*, volume 14, pages 81–86. ASME Press, 2004.

[170] K. H. Schindler and M. Sanguineti. Bounds on the complexity of neural-network models and comparison with linear methods. *Int. Journal of Adaptive Control and Signal Processing*, 17:179–194, 2003.

[171] N. N. Schraudolph. *Optimization of Entropy with Neural Networks.* PhD thesis, University of California, 1995.

[172] T. Schürmann and P. Grassberger. Entropy estimation of symbol sequences. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 6(3):414–427, 1996.

[173] M. Shahjahan and K. Murase. A pruning algorithm for training cooperative neural network ensembles. *IEICE Trans. on Information and Systems*, E89ŰD(3):1257–1269, 2006.

[174] C. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[175] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[176] S. Shwartz, M. Zibulevsky, and Y. Y. Schechner. Fast kernel entropy estimation and optimization. *Signal Processing*, 85:1045–1058, 2005.

[177] F. Silva and L. Almeida. Speeding up backpropagation. In E. R., editor, *Advanced Neural Computers*, pages 151–158, 1990.

[178] L. M. Silva, J. Marques de Sá, and L. A. Alexandre. Neural Network Classification using Shannon's Entropy. In *European Symposium on Artificial Neural Networks*, pages 217–222, 2005.

[179] B. W. Silverman. *Density Estimation for Statistics and Data Analisys*, volume 26. Chapman & Hall, 1986.

[180] W. Sun, J. Lu, and Y. He. Information entropy based neural network model for short-term load forecasting. In *Transmission and Distribution Conference and Exhibition: Asia and Pacific, 2005 IEEE/PES*, pages 1–5. IEEE, 2005.

[181] R. A. Tapia and J. R. Thompson. *Nonparametric Probability Density Estimation.* The Johns Hopkins University Press, 1978.

[182] G. Thimm and E. Fiesler. Pruning of neural networks. Technical Report RR 97-03, Institute for Perceptive Artificial Intelligence, 1997.

[183] H. L. Tsai and S. J. Lee. Entropy-based generation of supervised neural networks for classification of structured patterns. *IEEE Trans. on Neural Networks*, 15(2):283–297, 2004.

[184] C. Tsallis. Possible generalization of boltzmann-gibbs statistics. *Journal of Statistical Physics*, 52(1-2):479–487, 1988.

[185] J. C. A. van der Lubbe. *Information Theory.* Cambridge University Press, 1997.

[186] O. Vasicek. A test for normality based on sample entropy. *Journal of the Royal Statistical Society: Series B*, 36:54–59, 1976.

[187] D. Verma and M. Meila. A comparison of spectral clustering algorithms. Technical Report UW-CSE-03-05-01, Washington University, 2003.

[188] J. D. Victor. Binless strategies for estimation of information from neural data. *Phys. Rev. E*, 66(5):51903–51918, 2002.

[189] R. Vilalta, M. K. Achari, and C. F. Eick. Class decomposition via clustering: a new framework for low-variance classifiers. In *IEEE Int. Conference on Data Mining*, pages 673–676, 2003.

[190] P. Viola, N. N. Schraudolph, and T. J. Sejnowski. Empirical entropy manipulation for real world problems. In *Advances in Neural Information Processing Systems*, volume 8. MIT Press, 1996.

[191] P. A. Viola. *Alignment by Maximization of Mutual Information.* PhD thesis, MIT, 1995.

[192] T. P. Vogl, J. K. Mangis, J. K. Rigler, W. T. Zink, and D. L. Alkon. Accelerating the convergence of the back-propagation method. *Biological Cybernetics*, 59(4-5):257–263, 1988.

[193] P. J. Werbos. *Roots of Backpropagation.* Wiley, 1994.

[194] B. Widrow and M. E. Hoff. Adaptive switching circuits. In *1960 IRE WESCON Conv. Records*, volume 4, pages 96–104. Institute of Radio Engineers (now IEEE), 1960.

[195] R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1:270–280, 1989.

[196] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. *IEEE Trans. on Pattern Analysis and Machine Learning*, 15(11):1101–1113, 1993.

[197] D. Xu. *Energy, Entropy and Information Potential for Neural Computation.* PhD thesis, University of Florida, 1999.

[198] D. Xu and J. Príncipe. Learning from examples with quadratic mutual information. In *Neural Networks for Signal Processing VIII, IEEE Signal Processing Society Workshop*, pages 155–164, 1998.

[199] D. Xu and J. Príncipe. Training mlps layer-by-layer with the information potential. In *Intl. Joint Conference on Neural Networks*, pages 1716–1720, 1999.

[200] X. Xu. Dbscan. http://ifsc.ualr.edu/xwxu/.

[201] H. H. Yang and S. ichi Amari. Adaptive on-line learning algorithms for blind separation: Maximum entropy and minimum mutual information. *Neural Computation*, 9(7):1457–1482, 1997.

[202] H. H. Yang, S. ichi Amari, and A.Cichocki. Information-theoretic approach to bss in non-linear mixture. *Signal Processing*, 64(3):291–300, 1998.

[203] L. Yu, S. Wang, and K. Lai. An integrated data preparation scheme for neural network data analysis. *IEEE Trans. on Knowledge and Data Engineering*, 18(2):217–230, 2006.

[204] H. C. Yuan, F. L. Xiong, and X. Y. Huai. A method for estimating the number of hidden neurons in feed-forward neural networks based on information entropy. *Computers and Electronics in Agriculture*, 40(1-3):57–64, 2003.

[205] L. Yujian. A simple method to estimate the size of feedforward neural networks. In *IEEE Int. Conference on Systems, Man and Cybernetics*, volume 2, pages 1322–1326, 2003.

[206] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An efficient clustering method for very large databases. In *ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, pages 103–114, Montreal, Canada, 1996.

[207] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, 1(2):141–182, 1997.