UNIVERSITAT JAUME·I

Departament de Llenguatges i Sistemes Informàtics

Universitat Jaume I

Data Reduction Techniques in Classification Processes

Ph. D. THESIS

Presented by:

María Teresa Lozano Albalate

Supervised by:

Dr. J. Salvador Sánchez Garreta

Prof. Filiberto Pla Bañón

Castellón, July 25th 2007

UNIVERSITAT JAUME•I

Departament de Llenguatges i Sistemes Informàtics
Universitat Jaume I

Técnicas de Reducción de Datos en Procesos de Clasificación

Tesis Doctoral

Presentada por:
María Teresa Lozano Albalate

Dirigida por:
Dr. J. Salvador Sánchez Garreta
Dr. Filiberto Pla Bañón

Castellón, 25 de Julio de 2007

A mis padres, Bernabé Lozano y Eloína Albalate, quienes me han enseñado desde el ejemplo el valor de la constancia y el trabajo.

Abstract

The learning process consists of different steps: building a Training Set (TS), training the system, testing its behaviour and finally classifying unknown objects. When using a distance based rule as a classifier, i.e. 1-Nearest Neighbour (1-NN), the first step (building a training set) includes *editing* and *condensing* data. The main reason for that is that the rules based on distance need many time to classify each unlabelled sample, x, as each distance from x to each point in the training set should be calculated. So, the more reduced the training set, the shorter the time needed for each new classification process. This thesis is mainly focused on building a training set from some already given data, and specially on condensing it; however different classification techniques are also compared.

The aim of any condensing technique is to obtain a reduced training set in order to spend as few time as possible in classification. All that without a significant loss in classification accuracy. Some new approaches to training set size reduction based on prototypes are presented. These schemes basically consist of defining a small number of prototypes that represent all the original instances. That includes those approaches that select among the already existing examples (selective condensing algorithms), and those which generate new representatives (adaptive condensing algorithms).

Those new reduction techniques are experimentally compared to some traditional ones, for data represented in feature spaces. In order to test them, the classical 1-NN rule is here applied. However, other classifiers (fast classifiers) have been considered here, as linear and quadratic ones constructed in dissimilarity spaces based on prototypes, in order to realize how editing and condensing concepts work for this different family of classifiers.

Although the goal of the algorithms proposed in this thesis is to obtain a strongly reduced set of representatives, the performance is empirically evaluated over eleven real data sets by comparing not only the reduction rate but also the classification accuracy with those of other condensing techniques. Therefore, the ultimate aim is not only to find a strongly reduced set, but also a balanced one.

Several ways to solve the same problem could be found. So, in the case of using a rule based on distance as a classifier, not only the option of reducing the training set can be afford. A different family of approaches consists of applying several searching methods. Therefore, results obtained by the use of the algorithms here presented are compared in terms of classification accuracy and time, to several efficient search techniques.

Finally, the main contributions of this PhD report could be briefly summarised in four principal points. Firstly, two selective algorithms based on the idea of surrounding neighbourhood. They obtain better results than other algorithms presented here, as well as better than other traditional schemes. Secondly, a generative approach based on mixtures of Gaussians. It presents better results in classification accuracy and size reduction than traditional adaptive algorithms, and similar to those of the LVQ. Thirdly, it is shown that classification rules other than the 1-NN can be used, even leading to better results. And finally, it is deduced from the experiments carried on, that with some databases (as the ones used here) the approaches here presented execute the classification processes in less time that the efficient search techniques.

Resumen

El proceso de aprendizaje consta de diversos pasos: construcción de un conjunto de entrenamiento, entrenamiento del sistema, evaluación de su comportamiento y, finalmente, clasificación de objetos desconocidos. Cuando se utiliza como clasificador una regla basada en la distancia, por ejemplo la del vecino más cercano (1-NN; Nearest Neighbour), el primer paso (construcción de un conjunto de entrenamiento) incluye, entre otras, dos nuevas etapas: *editado* y *condensado* de los datos. La razón principal para la utilización de estas etapas es que las reglas basadas en la distancia requieren mucho tiempo para clasificar cada muestra x a etiquetar, dado que se debe calcular la distancia de x a cada ejemplo del conjunto de entrenamiento. Así, cuanto más se reduzca el conjunto de entrenamiento, menor será el tiempo necesario para cada nuevo proceso de clasificación. Esta tesis está enfocada principalmente a la construcción del conjunto de entrenamiento a partir de unos datos dados y, especialmente, en el condensado de dicho conjunto; sin embargo, también se comparan diferentes técnicas de clasificación.

El objetivo final de cualquier técnica de condensado es obtener un conjunto de entrenamiento que requiera el mínimo tiempo posible para llevar a cabo la etapa de clasificación. Todo ello, sin una pérdida significativa sobre la efectividad del clasificador. Dentro de este contexto, la presente tesis presenta diversos algoritmos nuevos para la reducción del conjunto de entrenamiento basados en prototipos. Estos esquemas básicamente consisten en definir un pequeño número de prototipos que representen a todas las instancias originales. Esto incluye tanto a los algoritmos que seleccionan entre los ejemplos ya existentes (condensado selectivo), como a aquellos que generan nuevas representaciones (condensado adaptativo).

Esas nuevas técnicas de reducción se comparan experimentalmente con algunas de las tradicionales, para datos representados en espacios de características. Para evaluarlos, se aplica la clásica regla de clasificación 1-NN. Sin embargo, también se han considerado otros clasificadores (clasificadores rápidos), como el lineal y el cuadrático, construidos en espacios de disimilaridad basados en prototipos, para observar como funcionan los conceptos de editado y condensado para esta familia de clasificadores.

A pesar de que el objetivo de los algoritmos propuestos en esta tesis es conseguir una gran reducción del conjunto de entrenamiento, su comportamiento es empíricamente evaluado sobre once bases de datos reales mediante la comparación no sólo del porcentaje de reducción sino también de la precisión de clasificación con los obtenidos por otras técnicas. Por tanto, el objetivo no es sólo encontrar un conjunto fuertemente reducido, sino también un equilibrio entre reducción y precisión.

Se pueden encontrar distintos modos de resolver un mismo problema. Por ejemplo, en el caso de utilizar una regla basada en la distancia, no únicamente existe la opción de reducir el conjunto de entrenamiento. Otra posibilidad consiste en aplicar diferentes métodos de búsqueda del vecino más próximo. Por ello, los resultados obtenidos por el uso de los algoritmos aquí presentados se comparan en términos de precisión de clasificación y tiempo con varias técnicas de búsqueda eficiente.

Finalmente, la mayor contribución de esta tesis puede ser brevemente resumida en cuatro puntos principales. En primer lugar, dos algoritmos selectivos basados en el concepto de vecindad envolvente, los cuales obtienen mejores resultados que otros algoritmos aquí presentados, y mejores también que otros esquemas tradicionales. En segundo lugar, una técnica adaptativa basada en mezclas (mixtures) de Gaussianas, la cual presenta mejores resultados en precisión de clasificación y reducción de tamaño que otros algoritmos adaptativos tradicionales, y similares a los del método LVQ. En tercer lugar se muestra como se pueden utilizar otras reglas de clasificación, diferentes al clasificador 1-NN, arrojando incluso mejores resultados. Y, finalmente, tal y como se deduce de los experimentos llevados a cabo, en el caso de algunas bases de datos (como las utilizadas aquí) los algoritmos aquí presentados consiguen una ejecución de los procesos de clasificación en un tiempo menor que las técnicas de búsqueda eficiente.

Acknowledgements

Tengo mucha gente a quien agradecer, empezando por mi familia: Bernabé, Eloína, Laura y Víctor, por su apoyo y ayuda en todo lo que hago. Debo mi agradecimiento también a mis directores, Salvador y Filiberto. A Pedro, director del proyecto con el que tuve la FPI, a José, por su paciencia y ayuda, y a mi tutor de doctorado, José Miguel, que realmente fue quien me animó a dar el paso, y adentrarme en este mundo. En definitiva, a mis compañeros del grupo de visión, empezando por Adolfo, compañero de despacho y de andanzas doctorales. Gracias especialmente a Javier, Raúl, M^{a} Ángeles, Arnoud, Ramón, Jorge, Gustavo, Joaquín, Isabel y Silvia. Y a todos aquellos que alguna vez pasaron por aquí viniendo de otros sitios: a Rosa, Vicente, Roberto, Luís, Nadia... Al Departamento de Lenguajes y Sistemas Informáticos, especialmente a Pablo, Óscar y Amparo. Y a tanta gente de la UJI que bien con su sonrisa, sus palabras o su compañía han hecho más agradable este camino. Entre ellos, Anna, Zoe, María, Gabriel y Toni.

Thanks to the ewi group (TU Delft): Bob, Pavel, Davidt and Carmen among others. Ela, thanks for your positiveness! Y a otra gente que conocí en Holanda, como Mónica, Irene Morata, Xose, Olano, Felipe, Irene Sequeiros, Alberto y Alberto Taboas por crear aquellos momentos que nunca olvidaré. A Fran, Etel, Vincent, Nieves, Marco, Fernando y Víctor por hacer más agradable la repetición de la jugada.

Je voudrais remercier aussi au LAAS-CNRS (Toulouse), spécialement à Michel, Omar, Andrés, Patrick, Nestor, Ángel et Juan. Y a otra gente que conocí allí, como el grupo de la Ponsan Bellvue, entre ellos a Anna (hay un dicho que dice que allí donde vayas encontrarás a alguien de Almazora), y del INSA, entre ellos a Maria, Andreea, Gustave, Quim, Albert, Sergio... Y por supuesto a la Casa de Aragón en Toulouse, quienes ocupan un rinconcito en mi corazón, especialmente Isabel, Violeta, Dorita y Fernando entre otros muchos.

Tinc molt que agrair també a la meua professora particular d'anglès, Mónica. Y al resto de mi *colla* (esos que dicen que como no entienden a que me dedico, creen que hago algo importante) que día tras día han soportando mis diferentes estados de ánimo, sin darme de lado por ello. Muchas gracias por vuestro apoyo. Ellos son además de Mónica, Carlos, Sergio, Estela, Sonia, Vicente, Ester, Vicente, Eliana, Manuel y Luisa, sin olvidarme de esas *nuevas adquisiciones* como son David, Lucas, Iker y desde ayer, Blai.

Y como no, a esa gente maña, de nacimiento u adopción, que tan bien me han acogido en *mi nueva universidad*, concretamente en el Departamento de Informática e Ingeniería de Sistemas y en esos DIISasters. Algunos de todos ellos son Licri, Arturo, Miguel Ángel, Elsa, Javier Campos, Javier Martínez, Fernando, Raquel, M^{a} Ángeles, María, Ana Cris, Darío, Diego, Juan Pablo, Pili, José, Cristian, Lili y Javier Mínguez, entre tantos otros.

Contents

List of Figures

List of Tables

List of Algorithms

List of Symbols

General

x	an unlabelled sample/object (from the set to classify) or an instance/example (from the training set)
p	a prototype/representative (from the condensed set)
S	a set of new samples to classify
X	a training set
R	a condensed/reduced set
n	training set size
r	condensed set size/number of codebooks
d	number of features
J	number of classes (in dissimilarities also: number of classifiers used to separate those classes)
Θ	set of (possible) classes
θ_i	class label of instance i
k	number of neighbours used (in k-d tree: dimensionality of the representation space)
N_j	cardinality of the class c_j
M	number of mixture components
P	nearest neighbour rule expected error

P^*	Bayes error
d_{nn}	distance to the nearest neighbour

Efficient Algorithms

dc	discriminant coordinate
v	split point
l	number of branches for each node
p	node (in the algorithm by [Fukunaga & Narendra(1975)])
M_p	instance selected to represent the subset of the node
R_p	radius of a node: maximum distance from the representative M_p, to each instance in the subset represented in the node
mu	distance used in the vp-tree in order to divide the training set in subsequent subsets

Dissimilarities

d	a dissimilarity measure
$d(x, z)$	dissimilarity measure between x and z
$D(x, R)$	vector of dissimilarities computed between x and the representatives in the condensed set R
$D(X, R)$	matrix of dissimilarities computed between each instance in X and the representatives in R
$D(T, R)$	dissimilarity space
$m_{(1)}, m_{(2)}$	mean vectors
$S_{(1)}, S_{(2)}$	estimated class covariance matrices
$p_{(1)}, p_{(2)}$	class prior probabilities

List of Abbreviations

TS	Training Set
ES	Edited Set
CS	Condensed Set
NN	Nearest Neighbour
1-NN	1-Nearest Neighbour
NCN	Nearest Centroid Neighbour/Neighbourhood
1-NCN	1-Nearest Centroid Neighbour
RSP	Reduction by Space Partition
LVQ	Learning Vector Quantisation
EM	Expectation-Maximisation
***k*-d tree**	*k*-Dimensional tree
vp-tree	Vantage Point-tree
GNAT	Geometric Near-neighbour Access Tree
AESA	Approximation and Elimination Search Algorithm
LAESA	Linear Approximation and Elimination Search Algorithm
NLC	Linear Normal density based Classifier
NQC	Quadratic Normal density based Classifier
FLD	Fisher Linear Discriminant

LP	Linear Programming
SRQC	Strongly Regularised Quadratic Classifier
SVM	Support Vector Machine

Part I

Introduction

Chapter 1

Introduction

Contents

Data Reduction Techniques in Classification Processes is the title of this PhD report. Therefore, it is the main goal we have been looking to achieve. *Data Reduction Techniques* are approaches in charge of diminish the quantity of information in order to reduce both memory and execution time. Traditionally, the concept of *Data Reduction* have received several names, e.g. editing, condensing, filtering, thinning, etc, depending on the objective of the data reduction task. There are two different possibilities depending on the *object* of the reduction. The first one is to reduce the quantity of instances, while the second one is to select a subset of features from the available ones. The later, *feature selection*, is not considered here, but just the former: *prototype selection*. Therefore, our main objective in this thesis is to introduce, analyse and test several techniques devoted to the reduction of the number of instances.

Classification Processes are referred to processes which classify/label a new sample using a discrimination function learned from a set of instances in a training set. And that is what we reduce: the training set. Therefore, we try to represent the complete training set by some representatives as much effectively as possible, in the sense of maintaining the same classification accuracy. So, we look for results in which, memory and time needs are reduced, while the original classification accuracy is preserved.

1.1 Motivation and General Objectives

Currently, in many domains (e.g. in multispectral images, text categorisation, biometrics or retrieval of multimedia databases) the size of the data sets is so extremely large that real-time systems cannot afford the time and storage requirements to process them. Under these conditions, classifying, understanding or compressing the available information can become a very problematic task. This problem is specially dramatic in the case of using some learning algorithms based on distances, such as the Nearest Neighbour (NN) rule [Dasarathy(1991)]. The basic NN scheme must search through all the available training instances (large memory requirements) to classify a new input sample (which is slow during classification). In addition, since the NN rule stores every instance in the Training Set (TS), noisy instances are stored as well, which can considerably degrade the classification accuracy.

Among the many proposals to tackle this problem, traditional methods rely on removing some of the training instances (i.e. they comply with data reduction). In the Pattern Recognition literature, the methods leading to a reduction of the training set size are generally referred to as *prototype selection* [Devijver(1982)]. Two different families of prototype selection approaches can be defined. Firstly, the *editing* approaches eliminate erroneously labelled instances from the original training set and avoid possible overlapping among regions from different classes. Secondly, the *condensing* algorithms aim at selecting (or modifying) a small subset of instances without a significant degradation of the resultant classification accuracy.

The many existing proposals in relation to *condensing* can be categorised into two main groups. First, the schemes that merely select a subset of the original instances [Dasarathy(1994), Aha et al.(1991), Hart(1968), Toussaint et al.(1985), Tomek(1976)] (selective schemes) and secondly, the ones that modify them [Chen & Jozwik(1996), Sánchez(2004), Kohonen(1995), Chang(1974)] (adaptive schemes). Here we will introduce several algorithms focusing on the problem of appropriately reducing the training set size by both, selecting some of the already existing instances, and generating a subset of new representatives. The primary aim is to obtain a considerable reduction rate in size, but without a significant decrease in classification accuracy. In brief we want to find a balance or a trade-off between these two objectives.

In this study, we will compare various selection and generation methods of a Condensed Set (CS), in combination with three classification strategies based on the Euclidean distance representation derived in the initial feature spaces. In addition, the behaviour of algorithms presented here will be compared to some effective search techniques.

The condensing methods used in our study are the traditional/classic ones, including the Chen's, Hart's, LVQ and RSP3 techniques [Chen & Jozwik(1996), Hart(1968), Kohonen(1996), Sánchez(2004)], as well as new techniques implemented during the development of this thesis. They include *MaxNCN*, *Iterative MaxNCN*, *Iterative kNeighbours*, *Consistent*, *Reconsistent*, *Centroide*, *WeightedCentroide* and *MixtGauss*, discussed in Chapters 3 and 4. We focuss on a few classification methods. The first one is the 1-NN rule, which assigns a new object to the class of the nearest neighbour in the resulting condensed set. The second and third methods are the Fisher Linear Discriminant (FLD) and the Quadratic Normal Density Based Classifier (NQC) trained in dissimilarity spaces. These are vector spaces in which every dimension encodes a dissimilarity to a given prototype as explained in Chapter 2. The effective search techniques used in our experiments are: k-dimensional tree, the proposal by Fukunaga and Narendra, vantage point-tree, GNAT, and LAESA and KAESA [Bendtley(1975), Friedman(1977), Fukunaga & Narendra(1975), Yianilos(1993), Brin(1995), Vidal(1986), Vidal(1994), Micó et al.(1994), Micó et al.(1996)]

1.2 Description of the Databases

The *UCI machine learning database* repository [Merz & Murphy(1998)] have been widely used in classification studies, e.g. in the works of [Wilson & Martinez(2000)]. That is the reason why eleven real data sets (see Table 1.1) have been taken from there to assess the behaviour of the algorithms introduced in this thesis.The selection of these databases is done to represent several training set sizes, different dimensionality of the databases and several number of classes.

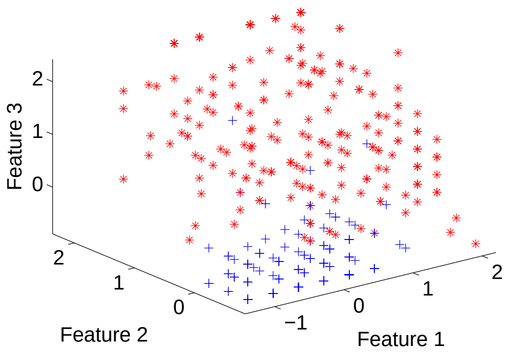
In this section we describe the data used in the experiments presented later on. For each database a plot representing the three first features is shown. We know that only three features are not representative for the data set, but it still gives some insight.

Cancer

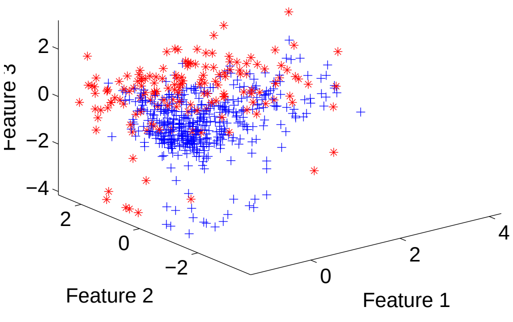
Cancer is a database offering 683 instances with information about 9 features. The number of classes in which these examples are classified is two. The first class is represented by 445 instances, and the second class is represented by 238 of them. See Figure 1.1 (a).

Diabetes

Pima Indians Diabetes database present 768 instances with information about 8 numeric-valued attributes divided in two classes (tested either positive or negative

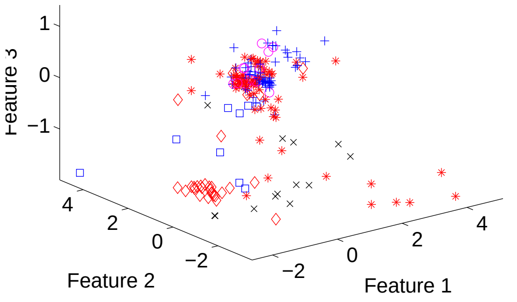


(a)

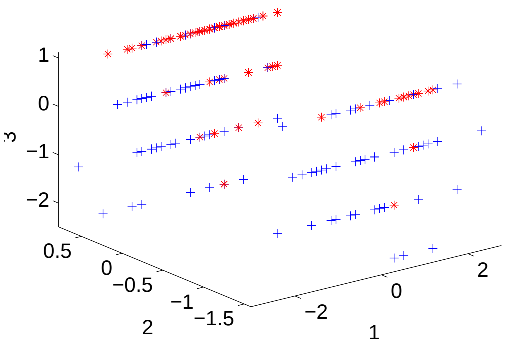


(b)

Figure 1.1: Representation of the Cancer (a) and Diabetes (b) databases.



(a)



(b)

Figure 1.2: Representation of the Glass (a) and Heart (b) databases.

Table 1.1: Data sets used in our experiments.

Data set	No. classes	No. features	TS size	Test set size
Cancer	2	9	546	137
Diabetes	2	8	614	154
Glass	6	9	171	43
Heart	2	13	216	54
Liver	2	6	276	69
Vehicle	4	18	677	169
Vowel	11	10	422	106
Wine	3	13	142	36
Phoneme	2	5	4 323	1 080
Satimage	6	36	5 148	1 287
Texture	11	40	4 400	1 100

for diabetes). 500 of the examples belong to one class and 268 examples represent the other class. See Figure 1.1 (b).

Glass

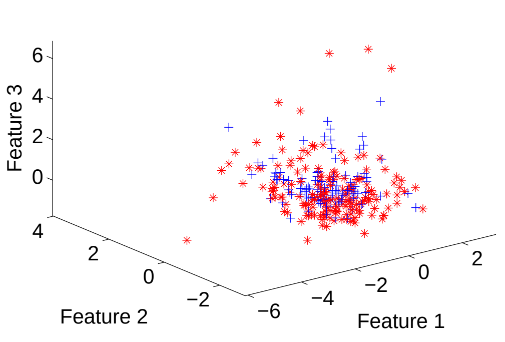
In the Glass Identification database there are 214 instances that represent 6 different types of glass (defined in terms of their oxide content: Na, Fe, K, etc). Each class is represented by 70, 76, 17, 13, 9 and 29 examples respectively. It has 9 numeric-valued features, and it comes from the USA Forensic Science Service. Three of them are shown in Figure 1.2 (a).

Heart

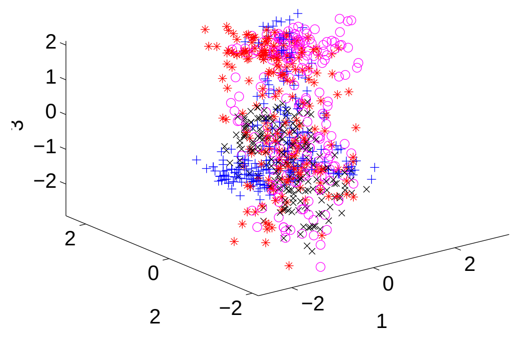
Heart Disease databases are a set of 4 databases (Cleveland, Hungary, Switzerland, and the VA Long Beach), which consist of 270 examples divided in two classes (the first one with 150 instances, and the second one with 120 of them) and 13 attributes. See Figure 1.2 (b).

Liver

Liver-Disorders database represent 2 classes with 345 instances (male patients: 145 in one class, and 200 in the other one) and 6 numeric-valued features. Its



(a)



(b)

Figure 1.3: Representation of the Liver (a) and Vehicle (b) databases.

representation is shown in Figure 1.3 (a).

Vehicle

This database has a size of 846 examples with information about 18 features. The total number of instances is divided in 4 classes: 218 instances represent the first class, 212 the second one, 217 the third one, and 199 the fourth one. See a graphical representation in Figure 1.3 (b).

Vowel

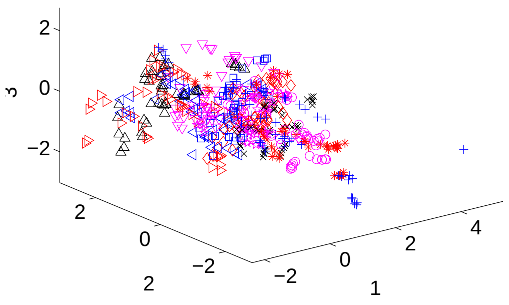
528 instances divided in 11 equal-sized classes (48 examples per class) conform the Vowel database. The 3 first of its 10 features are shown in Figure 1.4 (a).

Wine

Wine Recognition database uses chemical analysis to determine the origin of wines. For that, 13 continuous attributes are reported for 178 instances representing 3 classes (59 examples for the first one, 71 for the second one, and 48 for the last class). See Figure 1.4 (b).

Phoneme

Phoneme data set was first used to develop a real time system for the automatic speech recognition in French and Spanish. The 5404 instances (23 were discarded)


(a)

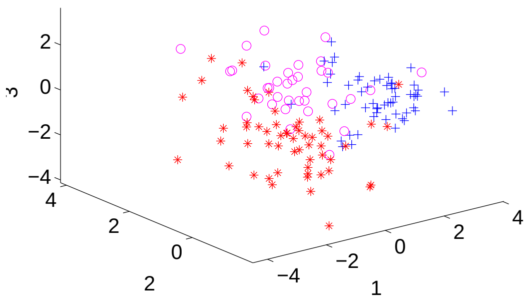
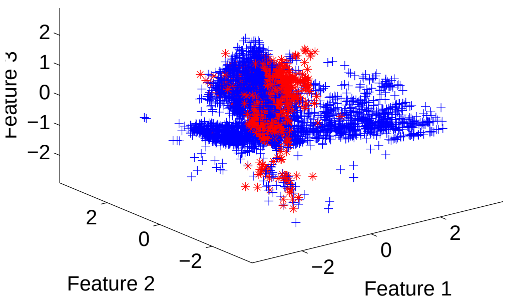

(b)

Figure 1.4: Representation of the Vowel (a) and Wine (b) databases.



(a)

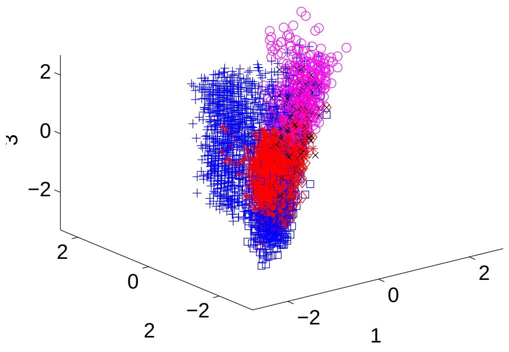

(b)

Figure 1.5: Representation of the Phoneme (a) and Satimage (b) databases.

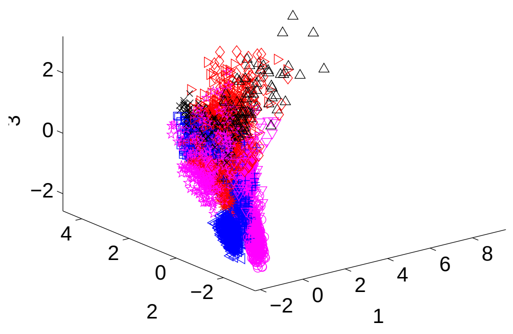


Figure 1.6: Representation of the Texture database.

were obtained by three different time observations in the pronunciation of 1809 separated syllables. The 5 features give some information about the 5 first harmonic, and the goal is to distinguish nasal and oral vowels (3818 and 1586, respectively). Phoneme database is represented in Figure 1.5 (a).

Satimage

Features in Satimage data base are derived from an image. Each one of the 6435 instances in this database is composed by four spectral bands of the same scene. The feature vector corresponds to a squared region of 3x3 pixels. Therefore, 36 features represent each one of the nine pixels in each of the four spectral images. 6 different classes compose this database, 1533 examples representing the first class, 703 the second class, 1358 the third one, 626 the forth one, 707 the fifth one, and 1508 the last one. See Figure 1.5 (b).

Texture

Texture database consist of 5500 instances that belong to 11 different classes or textures, with 500 examples each one. The feature vector associated to each instance consists of 40 different features, representing ten moments, each one for four different orientations (0^o, 45^o, 90^o and 135^o). See the representation in three dimensions in Figure 1.6.

1.2.1 Database Normalisation Process

If the databases were not normalised, the weight given to each feature would be different, depending on the range of each one of them, and the distances among the different vectors. So, the first step in order to prepare the data for our experiments has consisted of normalising the databases. Therefore, the data sets are first edited in order to avoid erroneous instances that could develop in a wrong normalisation. After the editing, all databases have been normalised by a unit variance (and then the Euclidean distances are computed).

Finally, for each database, the 5-fold cross-validation method is employed to obtain a division: each database is divided into five blocks, using four folds as the training set and the remaining block as the test set. So, 80% of the available patterns are used for training purposes and 20% for the test set in each fold.

1.3 Organization of the PhD report

Taking into account the objectives established in Section 1.1, this thesis report has been structured into several parts, each one consisting of a few chapters. The first part describes the motivation and objectives of the study, characterises the databases used and introduces the state of the art. The second part is the effective contribution of this thesis, according to the previously described objectives. It presents the algorithms proposed for data reduction. The third part presents results and conclusions. It summarises the findings, discusses the results and also presents the general concluding remarks. A forth part is added, as it is required in the PhD regulations, with all the important points of this thesis reported now in Spanish. Finally, the bibliography used along the preparation of this thesis is shown.

- Part I consists of Chapters 1 and 2.

    – Chapter 2 describes the state of the art. First, some guidelines about the learning process are presented in order to introduce the reader to the topic of the thesis. Secondly, the NN rule is described. Thirdly, prototype selection is introduced, and the state of the art in condensing techniques is specifically developed. Next, classical and recent nearest neighbour efficient search techniques are explained. Finally, classification techniques based on dissimilarities are developed.

- Part II, which presents the contribution of our work, consists of Chapters 3 and 4.

  - In Chapter 3, the Nearest Centroid Neighbour (NCN) rule is first clarified. Later, some non-adaptive condensing schemes based on this rule are introduced.
  - Chapter 4, introduces a number of adaptive condensing algorithms. Some of these schemes are based on the NCN rule. Nevertheless, the most important contribution shown in this chapter is a scheme based on mixtures of Gaussian probability distributions. Therefore, some ideas about mixtures of Gaussians in multi-modal class distributions are also presented.

- Part III is devoted to a comparative analysis of the methods introduced earlier, finalised by conclusions. It consists of Chapters 5, 6, 7 and 8.

  - Chapter 5 presents a quantitative comparison between some classical and new selective and adaptive algorithms based on the NCN rule introduced in Chapters 3 and 4. For that, a classification technique based on distances (1-NN) have been used. Finally, some conclusions are presented.
  - Chapter 6 focusses on a quantitative comparison using classification techniques based on dissimilarities (Fisher Linear Discriminant: FLD; and Normal density based Quadratic Bayes Classifier: NQC). The comparison includes some classical algorithms, the best ones based on the NCN rule and the adaptive approach based on mixtures of Gaussians (presented in Chapter 4). Chapter 6 finishes with some conclusions related to the commented results.
  - Chapter 7 presents a quantitative comparison of the results obtained by exhaustive search over reduced sets with those attained by nearest neighbour efficient search techniques over original sets. These results are analysed based on classification accuracy and time requirements.
  - Chapter 8 presents general conclusions referring to the complete thesis, and points out to its main contributions to Pattern Recognition. Then, possible extensions, and new lines of research are discussed. The thesis is finalised with a list of the already published articles from the research presented here.

- Part IV consists of Chapter 9.
  - Chapter 9 presents some requirements from the PhD regulations for this thesis reported in a different language than the official ones, to be in Spanish or *Valencià*. Mainly, motivation and general objectives, approach and methodology used, contributions, conclusions, and future work.

Chapter 2

State of the Art

Contents

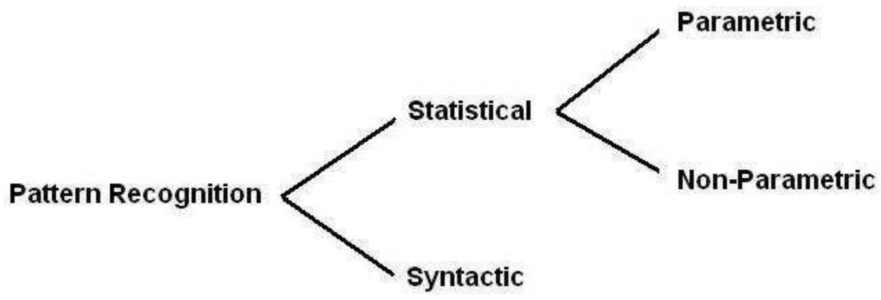
In *Pattern Recognition* two different branches can be differentiated depending on the space of representation used; see Figure 2.1. On the one hand, there is the *Syntactic Pattern Recognition* [Pavlidis(1977), Fu(1982)], which is based on the *Theory of Formal Languages.* According to this theory, a grammar rules the relations among models, like the natural language relates sentences. Therefore, classification in *Syntactic Pattern Recognition* consists of testing whether a chain can be generated from a determined grammar.

On the other hand, there is the *Statistical Pattern recognition* [Devijver(1982), Fukunaga(1990), Devroye(1996)], which is based on the *Decision Theory.* According to it, the representation space is a vector space, and it does not assume any structural relation among the different features. A model references a point in the representation space, whose coordinates form a vector of features.

The goal of any statistical classifier in the design step is to establish a set of frontiers in the representation space in order to separate models belonging to different classes (frontiers of decision). Therefore, in the classification step, the class of a sample is determined depending on the class of the region where it is located.

Within the *Statistical Pattern recognition* vision, two different branches are recognised (see Figure 2.1): *parametric* and *non-parametric* approaches. The *parametric*

Figure 2.1: Pattern recognition division.



approximation uses an a priori knowledge about the probability distribution of each class in the representation space (determined by a series of parameters). Therefore, these distributions define the frontiers of decision.

On the contrary, the *non-parametric* approaches do not have any a priori knowledge about the probability distribution in the representation space. The only information they have is a set of examples, called the Training Set (TS), whose labels should be known. Among the non-parametric statistical classifiers, the techniques based on neighbouring criteria should be highlighted, as they present several advantages over other non-parametric methods like, for example, their conceptual simplicity.

A neighbourhood-based decision function traditionally considered as a good classification rule is the k-NN. It has a number of advantages that favour its use. Firstly, it can easily be implemented, and it is conceptually simple. Secondly, its behaviour is asymptotically optimal [Cover & Hart(1967)], and thirdly, its expected error is bounded [Duda et al.(2001)]. In spite of these positive properties, there is a negative one to report: its high computational cost. This weakness of the k-NN rule is one of the points that we address in this thesis. Two alternative approaches to this problem can be used to diminish the corresponding computational cost associated to the k-NN rule. The first one is based on the selection or generation of a set of representatives from the training set. This reduction of the training set size basically accelerates the application of the k-NN rule, possibly without a loss of the effectiveness of a classifier.

The second alternative of reducing the computational cost of the k-NN rule is the use of the so-called nearest neighbour efficient search techniques [Moreno(2004)].

These algorithms make use of metric properties and look for the k nearest neighbours in a faster way than by computing distances to all of the training examples. Their behaviour is compared to the use of the *condensing* algorithms presented in this thesis.

Finally, other classifiers can be used that may lead to a better classification accuracy than the k-NN rule. As we focus on the *condensing* techniques, these classifiers are also defined on the distances to the representatives from the optimised condensed sets. Linear and quadratic classifiers are the simplest examples [Pękalska(2005)].

Taking into account this objective, this chapter presents the state of the art of *editing* and *condensing* approaches, search techniques and classifiers. It is organised as follows. Section 2.1 presents an introduction to proximity-based classification techniques, especially focusing on prototype selection: *editing* and *condensing*. Section 2.3 refers to some traditional *condensing* techniques, such as Learning Vector Quantisation (LVQ), and the Hart's and Chen's approaches. Section 2.4 presents the state of the art in relation to the Nearest Neighbour Efficient Search Techniques. Finally Section 2.5 takes the reader to the classification techniques based on dissimilarities.

2.1 Classification Techniques based on Neighbourhood

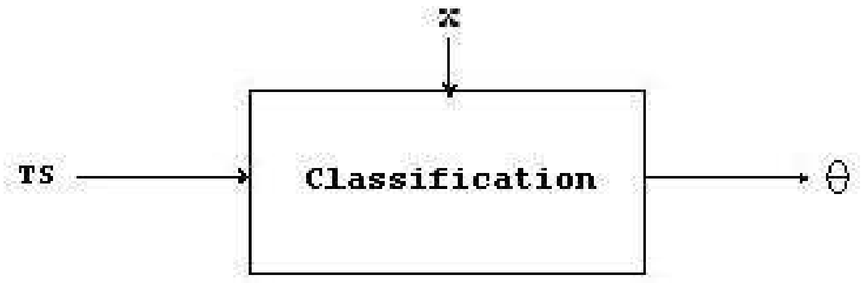
As stated above, the k-NN rule is generally considered as a good classifier. It has a number of advantages, namely:

1. It can easily be implemented and it is conceptually simple.
2. Its behaviour is asymptotically optimal [Cover & Hart(1967)].
3. Its expected error is bounded [Duda et al.(2001)].

Here, we will briefly explain some features of these three advantages. Firstly, we highlight its easiness of implementation and its conceptual simplicity. Imagine that two new fruits are first shown to someone. Then, when another unknown piece of fruit is presented, the individual will try to classify the new one by comparing it to the firstly shown pieces. So, the idea behind algorithms based on proximity is as follows. The classification of a new item x could be estimated based on the already known classifications of the elements sufficiently near to x, because observations that are close to each other will belong to the same class (or at least will have almost the same posterior probability distributions on their respective classifications).

Let $\{X, \Theta\} = \{(x_1, \theta_1), (x_2, \theta_2\}, \dots, (x_n, \theta_n)\}$ be a training set with n instances $\{x_i\}_{i=1}^n$ and their labels $\{\theta_i\}_{i=1}^n$. Let x be a new sample with an unknown class

Figure 2.2: Classification incomes and outcomes.



label. Assume $(x', \theta') \in \{X, \Theta\}$ is the nearest instance to the sample x. Then, the NN rule would be:

$$\delta_{NN}(x) = \theta' \Leftrightarrow d(x, x') = \min_{i=1..n} d(x, x_i) \tag{2.1}$$

Figure 2.2 represents the outcomes and incomes in a classification process. An implementation of a classifier based on the NN rule is shown in Algorithm 2.1. In case more details are needed, the reader is referred to [Chang(1974)].

Algorithm 2.1: Pseudo-code for a classifier using the NN rule.

```
S=TS
for $new\_object_i = eachobject(S)$ do
  $x_{NN} = NearestNeighbour(new\_object_i, X)$
  $new\_object_i.class = x_{NN}.class$
end for
```

Considering the asymptotic optimal behaviour of the NN rule, we must say that, in addition to its conceptual simplicity, the NN rule has a good behaviour when applied to non-trivial problems. In fact, the k-NN rule is asymptotically optimal in the Bayes sense [Dasarathy(1991)]. In other words, the k-NN rule performs as well as any other possible classifier, provided that there is an arbitrarily large number of representative prototypes available and the volume of the k-neighbourhood of x is arbitrarily close to zero for all x.

Given that the above conditions are fulfilled, the NN rule expected error P is bounded according to:

$$P^* \leq P \leq P^*(2 - \frac{J}{J-1}P^*), \tag{2.2}$$

where J is the number of classes and P^* is the Bayes error. We shall see that while the NN rule is a sub-optimal procedure (the NN rule usually leads to an error rate higher than the minimum possible), with an unlimited number of instances, the error rate is never worse than twice the Bayes error rate. In this sense, at least half of the classification information in an infinite data set resides in the nearest neighbourhoods. More details can be found in [Duda et al.(2001)].

In spite of a number of advantages, the k-NN rule has a serious drawback, namely a high computational cost.

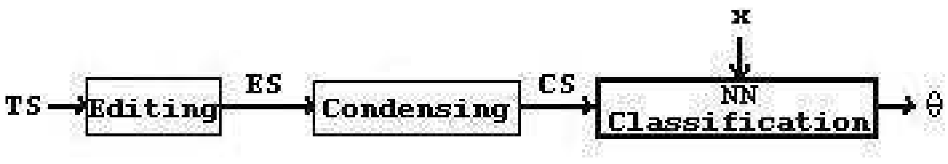
This drawback is the consequence of the need of storing a high number of instances in order to obtain an effective application of the NN rule. One of the two alternatives that can be used to reduce the computational cost associated to the k-NN rule is based on the selection or generation of a set of representatives from the training set. This reduction of the training set size is done to speed up the execution/application of the k-NN rule, ideally without a loss of the effectiveness of the classifier.

2.2 Prototype Selection

A small set of prototypes has the advantage of a low computational cost and small storage requirements for a similar or sometimes even an improved classification performance. Various ways of designing a proper set of representatives have been studied so far. Two families of such optimisation procedures are *editing* and *condensing.* As these methods lead to the reduction of the training set size, they are generally referred to as *prototype selection* methods [Devijver(1982)].

In the introduction to Chapter 2, we described the different steps of the learning process. When the NN rule is used as a classifier, the learning process could consist of two more steps to be accomplished: *editing* and *condensing.* Briefly, *editing* focuses on removing noisy instances, as well as close-border examples, while *condensing* takes care of maintaining only the representative instances (or generating new representative prototypes). The general idea is presented in Figure 2.3. It is shown there that the training set is the input to *editing*, whose output is the Edited Set (ES). This edited set is the input to *condensing*, whose output is the Condensed Set (CS). And finally, the resulting condensed set and the unknown sample to classify, x, are the classification inputs needed to obtain the final result: the class θ to which x belongs.

Figure 2.3: Classification steps using the NN rule as a classifier.



Algorithm 2.2: Pseudo-code of the Wilson's editing algorithm.

```
$X = TS$
$S = X$ (* initialisation *)
**for** $x_i = eachinstance(X)$ **do**
  $kNearestNeighbours(x_i, X - \{x_i\})$
  **if** $\delta_{kNN}(x_i)! = \theta_i$ **then**
    $S = S - \{x_i\}$
  **end if**
**end for**
```

2.2.1 Wilson´s Editing

In the learning process, *editing* is the step in charge of increasing the accuracy of predictions, when there is a great amount of noise in the training data. A basic *editing* algorithm removes noisy instances, as well as close border cases, eliminating a possible overlap between the regions from different classes and leaving smoother decision boundaries. Wilson introduced the first *editing* method [Wilson(1972)]. Briefly, the k-NN rule is used to estimate the class of each example in the training set followed by removing those examples whose true class labels do not agree with the ones judged by the k-NN rule.

Every instance from the training set (except for the x considered at each step) is used to determine the k nearest neighbours. That is, the estimation method is based on a *leave-one-out* procedure.

Let $\{X, \Theta\} = \{(x_1, \theta_1), (x_2, \theta_2), \ldots, (x_n, \theta_n)\}$ be a training set with n instances and J possible classes, and let k be the number of nearest neighbours to determine for each instance x. Algorithm 2.2 shows that the Wilson's *editing* scheme is easily implementable and understandable.

The Wilson's algorithm tries to eliminate mislabelled instances from the training set as well as those close to the decision boundaries. In general, it performs very well. In addition, it offers a good trade-off between classification accuracy and size reduction.

2.2.2 Other Editing Techniques

Many researchers have addressed the problem of *editing* by proposing alternative schemes. Some of their works are introduced here. [Tomek(1976)] proposed to apply the idea of the Wilson's algorithm repeatedly until no more instances can be removed. Tomek also proposed the $(All - k)$-NN editing scheme [Tomek(1976)]. It uses a set of the l-NN rules, with l ranging from 1 to k. In general, both algorithms achieve a higher storage reduction than the Wilson's editing, but similar in the classification accuracy. They are however higher at the computational efforts.

The generalised editing [Koplowitz & Brown(1981)] consists of removing some "suspicious" instances from the training set and also changing the class labels of some of them. Its purpose is to cope with all types of imperfections of the training instances (mislabelled, noisy and atypical cases). Recently, the generalised editing and Wilson's algorithm have been jointly used for the depuration method [Barandela & Gasca(2000)].

In the case of editing algorithms based on the leaving-one-out error estimate (the Wilson's scheme and its relatives), the statistical independence between test and training instances cannot be assumed because their functions are interchanged. In order to achieve this statistical independence, classification of instances can be performed in a hold-out manner. Thus, the Holdout editing [Devijver(1982)] consists of randomly partitioning the initial training set into $b > 2$ blocks of instances, $B_1, ..., B_b$, and then eliminating cases from each block using only two independent blocks at the same time. [Devijver(1982)] also introduced the Multiedit algorithm, which basically corresponds to an iterative version of the Holdout scheme using the 1-NN rule.

A genetic algorithm [Kuncheva(1995)] was also applied to define an edited set for the NN rule. Two different criteria were employed as the fitness function: the apparent error rate and a criterion based on the certainty of the classification. The empirical results show that the latter criterion led to a subset of the initial training set that provides higher classification accuracy in comparison to the whole original set, with random selection and with the Wilson's technique.

[Sánchez et al.(1997)A] presented an editing algorithm based on proximity graphs, such as the Gabriel graph and the relative neighbourhood graph. The first one computes the corresponding graph structure and then eliminates instances incorrectly classified by its graph neighbours. On the other hand, a combined editing-condensing scheme was also introduced to remove internal instances as well as border cases by using the concept of graph neighbours.

The rationale of the k-NN editing rule proposed by [Hattori & Takahashi(2000)] is very similar to that of the Wilson's scheme. In this method, the condition for an instance x to be included in the edited set is that all the k nearest neighbours must be from the class to which x belongs. Accordingly, this condition is much more severe than that in Wilson's algorithm and, as a consequence, the number of instances in the resulting edited set is equal to or less than in the Wilson's edited set.

The ACC filtering technique introduced by [Keung & Lam(2000)] tries to find centre instances of compact regions by considering the classification performance of each example in the training set. Each training instance is classified by its nearest neighbour. If it is correctly classified, then classification accuracy of its nearest neighbourhood will be increased. After processing all the training instances, the algorithm discards examples with the accuracy lower than a certain threshold. As centre instances are usually neighbours of other instances from the same class, they generally gain a high accuracy, thus are being retained by ACC.

2.3 Condensing Techniques

The *condensing* step aims at selecting a small subset of instances without a significant degradation in classification accuracy, in order to reduce both, storage and time required to process the selected data set. Within the *condensing* perspective, the many existing proposals can be categorised into two main groups. First, the selective schemes merely select a subset of the original instances [Aha et al.(1991), Dasarathy(1994), Tomek(1976), Toussaint et al.(1985), Hart(1968)] (selective family), while the adaptive schemes modify or generate them [Sánchez(2004), Chang(1974), Chen & Jozwik(1996), Kohonen(1995)] (adaptive or generative family). Apart from this, principal differences between *condensing* schemes are caused by the method used to correctly estimate which instances or prototypes are needed.

This section presents some traditional *condensing* techniques, such as the Hart's algorithm, which belongs to the family of selective schemes and the Chen's and LVQ approaches, which belong to the generative family (those algorithms applying prototype selection by *generating* new/non-existing prototypes). It is to note that the Chen's and LVQ techniques allow one to choose the condensed set size, while in the Hart's approach, this size is determined automatically.

Algorithm 2.3: Pseudo-code of the Hart's condensing algorithm.

```
$X = TS$
$S = \emptyset$ (* initialisation *)
**repeat**
  **for** $x_i = eachinstance(X)$ **do**
    $NearestNeighbour(x_i, S)$
    **if** $\delta_{NN}(X_i)! = \theta_i$ **then**
      $X = X - x_i$
      $S = S + x_i$
    **end if**
  **end for**
**until** $(EliminatedInstances(X) == 0)$ $OR$ $(X == \emptyset)$
```

2.3.1 Hart's Condensing

The Hart's algorithm [Hart(1968)] represents the first *condensing* proposal for the 1-NN rule. In this initial approximation, the idea of *consistency with respect to the training set* is used.

Definition A set of instances X is said to be consistent with respect to another set S, if X correctly classifies every instance in S, by using the 1-NN rule.

Using this definition of consistency, a condensed set should be a reduced and consistent set of instances.

As it can be seen in Algorithm 2.3, the Hart's *condensing* approach eliminates from the training set these examples which are not needed for the correct classification of the remaining instances by means of the 1-NN rule. This method is justified by the following observation. If an instance is incorrectly classified, this may happen because it is near the decision boundary and, consequently, it should not be eliminated from the training set.

The algorithm is simple and fast, as a consistent set is found in a very few iterations. The cost is linear: $\mathcal{O}(|X| * no\_of\_iterations)$. In addition, in most of the cases, the condensed set size is considerably small in comparison to the original training set size, provided that the training set has been previously edited in order to avoid overlaps between different class regions.

The weakness of the Hart's method lies in the impossibility of judging whether the resulting condensed set is the smallest consistent set. In fact, depending on the order in which the instances from the original training set are processed, different condensed sets can be obtained. All of them are consistent, but will likely have different sizes. This means that many of the instances selected by the Hart's procedure are not needed to keep the consistency property in the condensed set.

2.3.2 Chen's Algorithm

The Hart's condensing scheme cannot establish the desired condensed set size. In other words, it is not possible to control the size of the resulting subset. It could be interesting to solve some problems where the goal is not only the reduction of the computational effort, but also the effectiveness of a classifier. Chen [Chen & Jozwik(1996)] proposed a simple condensing scheme which allows one to control the resulting condensed set size.

In this algorithm, Chen introduces the idea of a *diameter* of a training set.

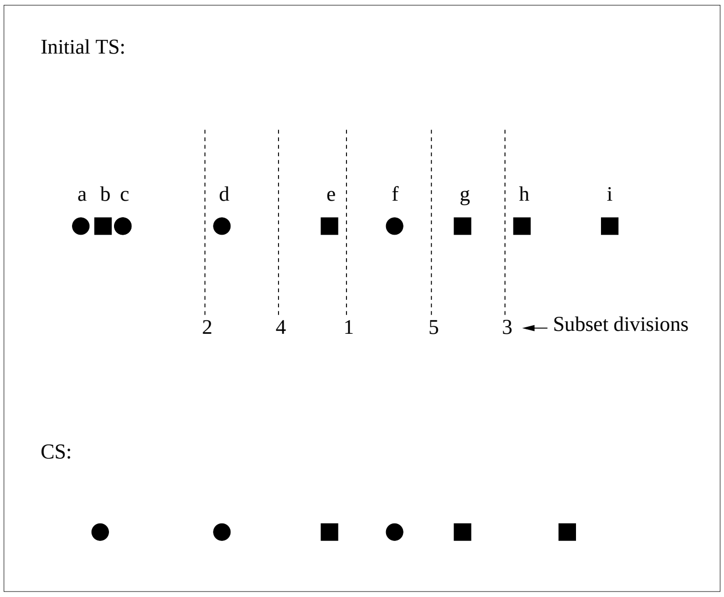
Definition The *diameter* of a training set is the distance between its two most distant instances.

Basically, the strategy relies on dividing the initial training set into successive subsets which are defined based on the notion of diameter. This process is repeated until the number of subsets reaches the number previously established as the condensed set size. After that, each resulting subset is replaced by a new prototype. The new representative is located at the subset gravity centre and obtains the label of the class more represented in this subset.

The Chen's scheme is illustrated in an intuitive way in Figure 2.4. The example is based on a training set with nine instances distributed into two classes, in a unidimensional space of representation. Let us choose a condensed set size of 6. The first step involves the search of the two farthest instances —in this case the ones marked with letters a and i. They define the *diameter* for the initial set. The middle position between a and i divides the original set into two subsets: $\{a, b, c, d, e\}$ and $\{f, g, h, i\}$. This middle position is represented in this figure by the dashed line marked by 1. The next division should be done in a subset containing instances belonging to both classes. As both subsets fulfil this condition, the one with the greatest diameter, $\{a, b, c, d, e\}$, will be divided. So, the line 2 divides it into two new subsets. Now, there are three subsets in total: $\{a, b, c\}$, $\{d, e\}$ and $\{f, g, h, i\}$. As the latter is a mixed subset with the greatest diameter, this is the one to be divided now (line 3). In the same way, divisions 4 and 5 are carried out. The process is stopped here, as the desired condensed set size of 6 is reached. Finally, the condensed set prototypes are obtained as the gravity centres of all the subsets in an area. The labels assigned to the generated representatives correspond to the classes more represented in the final subset.

Let $n_d \leq n$ be the condensed set size we would like to obtain from the initial

Figure 2.4: Illustration of the performance of the Chen's condensing algorithm.



training set, of the size n. Taking into account this notation, Algorithm 2.4 shows the Chen's condensing scheme.

The main interesting point about the Chen's algorithm in comparison to other condensing algorithms is quite clear. As it is possible to establish the condensed set size, the effectiveness of a classifier can be controlled. As a consequence, by means of the Chen's scheme, it is intended to obtain an adequate balance between computational needs and the required accuracy of the classifier in the given classification problem.

2.3.3 RSP Family

Sánchez introduced the family of RSP (Reduction by Space Partition) algorithms [Sánchez(2004)], which are based on the idea of the Chen's algorithm. The main difference between the Chen's and one of the RSP approaches, RSP3, is that in the former, any subset containing a mixture of instances belonging to different classes can be chosen to be divided. On the contrary, in RSP3, the subset with the highest overlapping degree (defined as a ratio of the average inter-class distance and the average intra-class distance) is the one picked to be split. Furthermore, for RSP3, Algorithm 2.4: Pseudo-code of the Chen's condensing algorithm.

```
(* initialisation *)
$n_c = 1; i = 1; C(i) = X; D = X$
(* division *)
$(p_1, p_2) = FarthestInstances(D)$
**while** $n_c < n_d$ **do**
  $(D_1, D_2) = Divide(D)$
  $n_c = n_c + 1; C(i) = D_1; C(n_c) = D_2$
  $I_1 = SubsetsWithDifferentClassInstances(C)$
  $I_2 = C - I_1$
  **if** $I_1 != \emptyset$ **then**
    $I = I_1$
  **else**
    $I = I_2$
  **end if**
  $(q_{j1}, q_{j2}, j) = MaximumDiameterEnds(I)$
  $D = C(j); p_1 = q_{j1}; p_2 = q_{j2}; i = j$
**end while**
(* representation *)
**for** $C(i) = C(1)..C(n_d)$ **do**
  $C(i).centre = CalculateGravityCentre(C(i))$
  $C(i).class = MaximumClass(C(i))$
**end for**
```

the splitting process continues until every subset is homogeneous (i.e., all instances in a given subset are from the same class) and finally, each subset is represented by its gravity centre. This procedure can be summarised as in Algorithm 2.5.

2.3.4 Learning Vector Quantisation Family

LVQ is an adaptive condensing technique in statistical pattern recognition, which allows one to choose the number of prototypes. It is used in many applications such as speech recognition, even when the prior probabilities for the classes are very different. In general, the classes can be described by a relatively small number of prototypes p_i placed within each class region of the decision boundary by means of neighbourhood measures.

In the initialisation step, the prototypes are placed within the training set, by maintaining the same number of representatives in each class. Since the class borders

Algorithm 2.5: Pseudo-code of the RSP3 algorithm.

```
(* initialisation *)
$n_c = 1; D = X$
(* division *)
$I_1 = SubsetsWithDifferentClassInstances(D)$
while $I_1 != \emptyset$ do
   $(p_1(j), p_2(j)) = MaximumDiameterEnds(I_1)$
   $D = C(j); p_1 = q_1(j); p_2 = q_2(j)$
   $(D_1, D_2) = Divide(D)$
   $n_c = n_c + 1; C(j) = D_1; C(n_c) = D_2$
   $I_1 = SubsetsWithDifferentClassInstances(C)$
   $I_2 = C - I_1$
end while
(* representation *)
for $C(i) = C(1)..C(n_c)$ do
   $C(i).centre = CalculateGravityCentre(C(i))$
end for
```

Algorithm 2.6: Pseudo-code of the LVQ approach.

```
(* initialisation *)
$t = 0$
$D(t) = ExtractNInstances(n_m, X)$
(* learning stage *)
repeat
   $x = ExtractOneInstance(X)$
   $D(t+1) = D(t) + x$
   $t = t + 1$
until $(t = total\_learning\_steps)$
```

are represented by segments of midplanes between examples of neighbouring classes (borders of the so-called Voronoi tessellations), it may also seem to be a good strategy to approximate the class borders, using the fact that the average distances between the adjacent prototypes should be the same on both sides of the borders. A general LVQ procedure can be written as follows:

Let $x \in X$ be an input sample, let p_c be the nearest codebook vector p_i to x, and let $p_c(t)$ represent the codebook vector p_c at the step t. The learning process in the basic version of the LVQ, i.e. the LVQ1 algorithm, consists of updating the position of p_c. If the class label of the codebook vector p_c matches the class label of the training instance x, then the codebook vector is moved towards x. Otherwise, it is moved away from the given input sample. The modifications to the codebook vector p_c are performed according to the following general rule:

$$\begin{aligned} p_c(t+1) &= p_c(t) + \alpha(t)[x(t) - p_c(t)] && \text{if } class(x) = class(p_c), \\ p_c(t+1) &= p_c(t) - \alpha(t)[x(t) - p_c(t)] && \text{if } class(x) \neq class(p_c), \\ p_i(t+1) &= p_i(t) && \text{for } i \neq c \end{aligned} \tag{2.3}$$

where $0 < \alpha(t) < 1$ denotes the corresponding learning rate, which may be either constant or decrease monotonically with time.

In the case of LVQ2, two codebook vectors, p_i and p_j, the nearest neighbours to an input sample x, are updated simultaneously. While p_i must belong to the correct class, p_j must belong to a wrong class. Moreover, x must fall into a "window" defined around the midplane of p_i and p_j. x is said to fall in a "window" of relative width w if

$$min(\frac{d_i}{d_j}, \frac{d_j}{d_i}) > \frac{1-w}{1+w} \tag{2.4}$$

where d_i is the Euclidean distance from x to p_i, and d_j is the Euclidean distance from x to p_j. A relative width w from 0.2 to 0.3 is recommended. The adjustments in the LVQ2 algorithm can be expressed as follows:

$$\begin{aligned} p_i(t+1) &= p_i(t) + \alpha(t)[x(t) - p_i(t)], \\ p_j(t+1) &= p_j(t) - \alpha(t)[x(t) - p_j(t)] \end{aligned} \tag{2.5}$$

The LVQ2 algorithm can be improved in order to include corrections which ensure that p_i continues to approximate the class distributions, leading, however to a longer learning process. Thus, the general modifications in the LVQ3 scheme are performed according to the LVQ2 conditions. Moreover, if x, p_i and p_j belong to the same class, then the learning rule is defined as:

$$p_k(t+1) = p_k(t) + \varepsilon\alpha(t)[x(t) - p_k(t)] \tag{2.6}$$

for $k \in \{i, j\}$. In [Kohonen(1996)], the authors recommend the applicable values of ε between 0.1 and 0.5, and stated that the optimal value of ε depended on the size of the "window".

In the experiments, a variant of the original LVQ algorithms family, different from the ones explained so far is used, namely the OLVQ1, *Optimised-Learning-Rate* LVQ. The basis of this method is the LVQ1 in such a way that an individual learning rate $\alpha_i(t)$ is assigned to each prototype p_i. Several prototypes are assigned to each class such that they approximately minimise the misclassification error in the 1-NN classification. The following equations define the process:

$$\begin{array}{ll} p_c(t+1) = p_c(t) + \alpha_c(t)[x(t) - p_c(t)] & \text{if } class(x) = class(p_c), \\ p_c(t+1) = p_c(t) - \alpha_c(t)[x(t) - p_c(t)] & \text{if } class(x) \neq class(p_c), \\ p_i(t+1) = p_i(t) & \text{for } i \neq c \end{array} \tag{2.7}$$

where $x(t)$ is an input sample.

In [Kohonen(1996)], the "optimal" values of $\alpha_i(t)$ are determined by the following recursion:

$$\alpha_i(t) = \frac{\alpha_i(t-1)}{1 + s(t)\alpha_i(t-1)}, \tag{2.8}$$

where $s(t) = +1$ if x and p_c belong to the same class (the classification of the prototype p_c is correct) and $s(t) = –1$ if x and p_c belong to different classes (the classification is wrong). Therefore, it is necessary to stop the learning process after some "optimal" number of steps. The *Optimised-Learning-Rate* LVQ may be stopped after, typically, 200 iterations.

2.3.5 Other Condensing Techniques

Many researchers addressed the idea of *condensing* by proposing several approaches. Some of their algorithms are recalled in this subsection. A family of learning algorithms based on instances were presented by [Aha et al.(1991), Aha(1992)]. Instance Based learning algorithm 1 (IB1) was simply the 1-NN technique, and was used as a baseline.

On the contrary, the IB2 approach is incremental, starting with a condensed set initially empty, and adding to it each instance in the training set that is not correctly classified by the examples already in the condensed set. This technique keeps border points. In fact, it is very similar to Hart's condensing. The main difference is that in the Hart's approach, the initial condensed set is started with an example of each class and the process is repeated until no more instance is added to the condensed set. So, the Hart's condensing necessarily classifies every instance in the training set correctly, meanwhile the IB2 algorithm does not accomplish it. More than that, as erroneous examples are usually misclassified and consequently saved in the condensed set, the IB2 technique is extremely sensitive to noise.

In order to avoid so extreme sensitivity, the IB3 version was introduced. The point of this approach is to keep only *acceptable* misclassified examples. In order to know which misclassified instance is *acceptable*, its bounds on accuracy and frequency are statistically compared. If the corresponding difference is bigger than a range, it is saved in the condensed set. If it is lower than that range, it is dropped. Other examples are retained in the condensed set during the process, and if they do not prove to be *acceptable*, then they are dropped at the end. Due to this *acceptability* clause, the IB3 achieves greater reduction in the condensed set size, and higher accuracy than IB2.

In relation to ordered removal, a family of algorithms was introduced under the names DROP1-DROP5 in [Wilson & Martinez(1997), Wilson & Martinez(2000)]. The DROP family is formed by a set of condensing algorithms which are independent on the order in which instances are presented. In general, what these algorithms do, is initialise a set R with the instances from the training set, and use a basic rule to decide whether it is safe to remove an instance from the progressively reduced set, R. An example is removed when R results in the same level of generalisation, but with lower storage requirements. The basic idea is, not to see if an instance x, is correctly classified to eliminate itself, but to see if its associates (the examples who have x as one of their k neighbours) are correctly classified without x. With such a general behaviour, noisy instances are removed. A noisy instance usually has associates that come from a different class, and such associates will be at least as likely correctly classified as without this noisy example.

Among all the algorithms in the DROP family, the DROP3 is the version selected by the authors [Wilson & Martinez(2000)] as the one with the best trade-off between size reduction and classification accuracy. Firstly, the DROP3 version removes noisy instances, as well as close border points (smoothing the decision boundary slightly): any instance misclassified by its k nearest neighbours is removed. Secondly, the examples are sorted by the distance to their nearest *enemy* remaining in R. And, finally, the basic rule is applied: remove an instance x, if at least as many of its associates in the training set would be correctly classified without x. As a result, instances far from the decision boundaries are removed first. So, examples internal in the clusters are removed early in the process, even if there were noisy instances nearby.

The Decremental Encoding Length (DEL) algorithm, which was introduced by [Wilson & Martinez(2000)], is the same as DROP3, except that it uses a heuristic encoding length to decide in each case whether an example can be removed. First, noisy instances from R are removed, i.e. an example is removed if it is misclassified by its k nearest neighbours and by its removing the encoding length cost does not increase. Secondly, the remaining instances are sorted by the distance to their nearest *enemy* (farthest, first). Finally, the remaining instances are removed provided that the cost function is decreased. Such a cost function is defined as

$$COST(r, n, m) = F(r, n) + rlog_2(J) + F(m, n - r) + mlog_2(J - 1) \tag{2.9}$$

where n is the number of instances in the training set, r is the number of examples in R, m is the number of examples misclassified by the ones in R, J is the number of classes, and $F(r, n)$ is the cost of encoding to be presented next.

$$F(r, n) = log^*(\sum_{j=0}^{r} J_j^n) = log^*(\sum_{j=0}^{r} \frac{n!}{j!(n-j)!}) \tag{2.10}$$

where log^* is the sum of the positive terms of $log_2(m)$, $log_2(log_2(m))$, etc.

In case more information about algorithms recalled in this subsection is needed, the reader is referred to the articles of the corresponding algorithms, as well as to the paper by [Wilson & Martinez(2000)], which presents a summary on the state of the art of reduction techniques.

2.4 Nearest Neighbour Efficient Search Techniques

The NN rule is used in many tasks thanks to its simplicity and efficiency. The simplest algorithm for implementing this rule is the one known by *exhaustive search.* It calculates each distance from the item to classify, x, to the examples in the training set. Afterwards, the class of the instance with the minimum distance is assigned to x. This algorithm is however not very advisable in some situations, i.e. when the training set is large, when the distance calculations require a considerable amount of time, or simply in those cases where the data dimensionality is so high that even the Euclidean distance calculation results in a very demanding task. In such cases, the exhaustive search does not seem a good option.

Various efficient algorithms to find the nearest neighbour have been developed in order to avoid the exhaustive search. [Dasarathy(1991)] did the first compendium of nearest neighbour search algorithms. Recently [Chavez et al.(2001)] has published a complete classification involving the more important nearest neighbour search algorithms in general metric spaces.

Many of these algorithms fit in the concept of *approximation and elimination* [Ramasubramanian & Paliwal(2000)]. The objective of an algorithm based on this scheme is that the savings in distance calculations, due to the elimination of instances, balance the computational cost of the approximation and the elimination processes. Among the algorithms taking profit of the above technique, a number of well-known algorithms can be listed: the k-dimensional tree (k-d tree) [Bendtley(1975), Friedman(1977)], the proposal by [Fukunaga & Narendra(1975)], the vantage point-tree (vp-tree) [Yianilos(1993)] and the Geometric Near-neighbour Access Tree (GNAT) [Brin(1995)]. Other approaches that fit in this scheme are the family of Approximation and Elimination Search Algorithm (AESA) [Vidal(1986), Vidal(1994), Micó et al.(1994), Micó et al.(1996)], which are very efficient when the distance calculations are hard to obtain.

2.4.1 An Example of Approximation and Elimination Algorithm: the k-Dimensional Tree

The great part of the efficient search techniques is based on a data structure, which is used to store the training set. This structure is usually a tree, as it is the case of one of the well-known algorithms: K-dimensional tree (k-d tree) [Bendtley(1975), Friedman(1977)]. In spite of the fact that many improvements have been implemented, the k-d tree is the reference algorithm when using Euclidean distances.

k is the dimensionality of the representation space. The k-d tree is a binary structure whose nodes (different from leaves) contain the information about a coordinate. This information divides each subtree into two new ones until the leaves are reached, There buckets of instances are stored. The main idea in the construction of the k-d tree is to find a hyperplane which divides the set of instances, X, into two subsets, and recursively, searches for a new hyperplane in each subset. In order to obtain a tree as balanced as possible, the usual way to proceed is to choose a hyperplane in the median of the values of the *discriminant coordinate* (the one with the largest width).

The construction of the tree is as follows. The root represents the complete training set. A discriminant coordinate is chosen, and two new nodes appear in the tree, each one representing a subset. For each node, a new discriminant coordinate is calculated, and it is again divided into two new nodes. This continues until the number of instances in a subset is less than (or equal to) the maximum bucket size admitted. Finally the process stops, and there is the last node of a branch: a leave.

During the classification step, the tree is traversed according to the *branch and bound* scheme, searching for the nearest neighbour of the sample to classify x. At each node, a comparison takes place: the value of x in the discriminant coordinate (dc) for this node is compared to the split point v (median of dc). Then the branch in the nearest direction, according to dc, is chosen. On the one hand, if $x[dc] + d_{nn} \leq v$ (where d_{nn} is the distance to the nearest neighbour obtained until now), the right branch of this node cannot contain the nearest neighbour. So, searching for the nearest neighbour in the right branch is not necessary (this fact is called *to prune a branch*). On the other hand, if $x[dc] - d_{nn} \geq v$, the search for the nearest neighbour in the left branch is not necessary. When the current node is a leave, the sample is compared to each instance in the bucket.

2.4.2 Other Efficient Search Techniques

Apart from the k-d tree, several efficient search techniques have been introduced by different authors. Some of them are briefly presented here.

Vantage Point Tree

The vantage point tree (vp-tree) was introduced by [Yianilos(1993)] for searching the nearest neighbour in general metric spaces. It consist of constructing a binary tree, in which each node represents a subset of the training set instances. A *vantage point* is used in order to divide the subset of each node in two different ones (one per branch). So, each node, but the leaves, contains a *vantage point* and two branches. The left branch contains the subset within the instances which are nearer to the *vantage point* than a distance *mu* (and its corresponding subsets at each new level). The tree is constructed recursively, choosing a new *vantage point* (and a *mu*) for each node, except for the leaves. Leaves store subsets with only an instance. In order to search in the vp-tree, the classical *branch and bound* scheme is used. The kvp-tree is an extension for the vp-tree, consisting of calculating the k nearest neighbours instead of just the first nearest neighbour.

Fukunaga and Narendra Algorithm

The algorithm by [Fukunaga & Narendra(1975)] was one of the first applicable in general metric spaces. It is based in the construction of a tree representing the training set, as the k-d tree and others do. For each new level, the subset is divided in l new subsets. So, each node (except the leaves) is divided in l branches, and contains the information for a representative M_p and a radius R_p (maximum distance from M_p to each example in the subset). Each leave contains a subset of instances.

The different steps conforming the recursive search used in this data structure are presented next.

1. Let us have a sample x and a node p (which is not a leave) where x is contained. Calculate the distance from x to each representative in the children nodes of p, and update the nearest neighbour.

2. Among all the children nodes of p that have not been pruned nor visited, take the one whose distance to x will be the smallest. Prune it, if it satisfies the first rule of pruning (see Figure 2.5): $d(x, M_p) > d(x, n) + R_p$

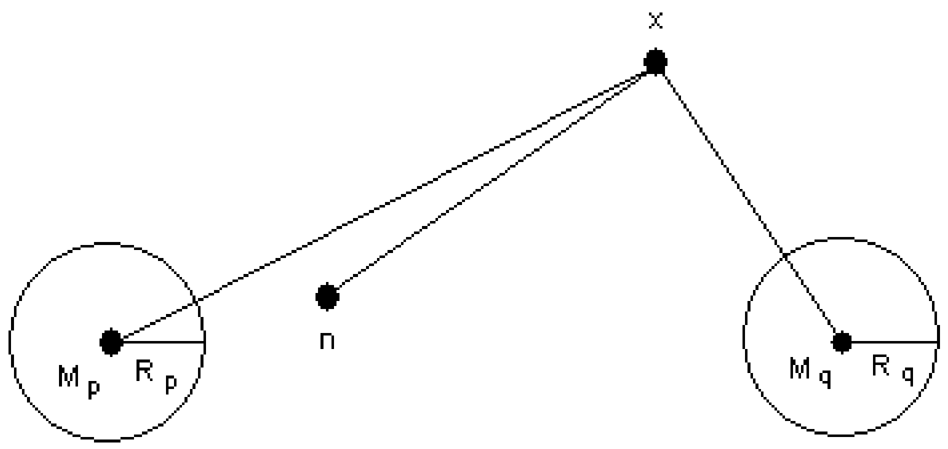


Figure 2.5: The first rule of pruning in the algorithm of Fukunaga and Narendra.

3. If p is a leave, eliminate those instances x from p, if it satisfies the second rule of pruning (see Figure 2.6): $d(e, M_p) + d(x, n) < d(x, M_p)$. Calculate the distance to x, for those examples not eliminated from the subset of p, and update the nearest neighbour. If p is not a leave, search recursively in p descendants.

4. Repeat steps 2 and 3 until all nodes of the tree have been either pruned or visited.

Geometric Near-neighbour Access Tree

[Brin(1995)] introduced a balanced tree that tries to represent the geometric structure of the training set. On the one hand, his data structure works well in high dimensionality spaces as well as in spaces where the distance calculation time in the search process is important. On the other hand, the tree construction is specially heavy in time. This is, as the author recognises in the article, the principal GNAT drawback. The main difference among this data structure, and others used for efficient search is that the number of children nodes of this tree is different for each

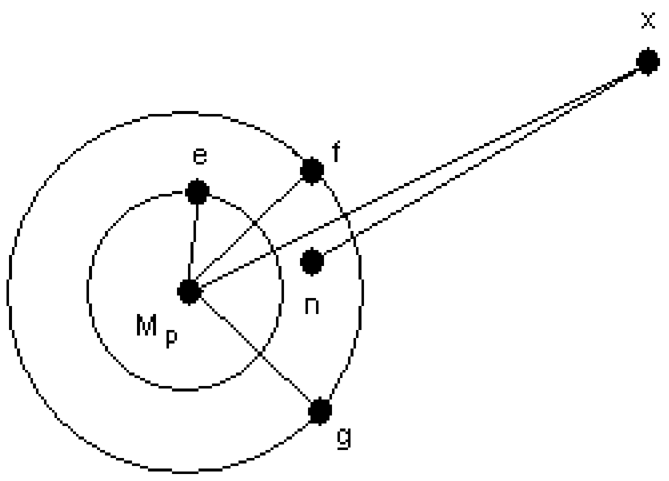


Figure 2.6: The second rule of pruning in the algorithm of Fukunaga and Narendra.

node (having a maximum and a minimum), and it is proportional to the number of instances that represent. Its construction is quite simple, and consists of several steps:

1. Let us choose k split points from the training set, $v_1, v_2, \ldots, v_k$. The value of k could change for each node. It is preferable to have the split points as much separated as possible.

2. Divide the training set into subsets, each one related to a split point p_i. Each instance, x, belongs to the subset determined by the closer split point.

3. For each pair of split points, (v_i, v_j), calculate the minimum and maximum of the distances among v_i and each x in the subset related to v_j, and the

minimum and maximum of the distances among v_j and each x in the subset related to v_i.

4. Repeat steps 2 and 3 recursively for each node, until all leaves have single instances.

Approximation and Elimination Search Algorithm

The family of AESA (Approximation and Elimination Search Algorithm) [Vidal(1986), Vidal(1994), Micó et al.(1994), Micó et al.(1996)] calculates a very reduced number of distances that does not depend on the training set size. So, on the one hand, their results are very time-efficient, when the distances are hard to calculate. On the other hand, when the distance calculations are light, the additional processing is responsible for slowing this technique. The algorithms, used in general metric spaces are based on a method of *approximation and elimination.* The construction of a tree is divided into the following steps:

1. At the beginning of the AESA algorithm, there is a preprocessing step consisting of the distance calculation among each instance in the training set, followed by the matrix storage.

2. In the classification step, any instance is chosen such as a candidate to the nearest neighbour, and its distance to the sample is computed (*approximation*). It is also eliminated from the training set, and the nearest neighbour is updated.

3. This distance is used in order to calculate an inferior level from the distance of each instance in the training set to the sample, using the triangular inequality. If the inferior level is superior to the distance to the temporary nearest neighbour, the instance is eliminated (*elimination*).

4. The process is repeated until there is no instance in the training set.

On the one hand, this algorithm is probably the one which calculates less distances in order to obtain the nearest neighbour. So, when the distance calculations are heavy (and the training set size is not quite big), AESA is the technique to be used. On the other hand, the matrix of distances calculated in the preprocessing step could have an excessive size when the training set is quite big, as the spatial complexity is quadratic in the training set size because of the matrix.

Linear Approximation and Elimination Search Algorithm

The difference between LAESA (Linear Approximation and Elimination Search Algorithm) [Micó et al.(1994), Micó et al.(1996)] and AESA is that the former reduces the spatial complexity from quadratic to linear. In order to obtain this complexity reduction, in the preprocessing step of LAESA, the pairwise distances between examples in the training set are not calculated, but the distances between examples from a small subset of the training set (*basic instances*) and each training example. These distances are stored in a rectangular matrix. So, the LAESA's complexity is linear, and the number of distances to be calculated depend on the size of this subset. The *basic instances* to be chosen should be positioned far away from each other.

During the classification step, the *basic instances* are the ones to be chosen, and their distances to the sample are calculated. Using these distances, and the ones calculated during the preprocessing step, an inferior level for each *basic instance* is obtained. When every *basic instance* has been eliminated, the levels are not updated any more, and the search uses a procedure similar to that of the AESA algorithm.

In case more information about these or other effective search algorithms is needed, the reader is referred to the articles cited for each approach, or to the PhD report by [Moreno(2004)].

2.5 Classification Techniques based on Dissimilarity

An intuitive way of determining the class of an unknown object is by analysing its similarity to a set of representatives either selected or generated from a given training set of instances with known class labels. Similarities or dissimilarities can be either directly computed from the raw object observations or based on an intermediate feature representation.

2.5.1 Dissimilarity Spaces

Definition *Similarity* is a quantitative measure which describes how similar two patterns are, based on their representations (e.g. features).

As a consequence, *similarity* takes a larger value for objects that are similar than for the ones that are dissimilar. *Dissimilarity* is, therefore, a measure describing the difference between patterns. In general, a nonnegative *dissimilarity* measure should express a degree of commonality between pairs of objects. It should be zero for two identical objects, take small values for similar objects and take large values for objects that mutually differ. It does not need to be a metric provided that it is meaningful for the problem.

Assume a training set X of n instances represented in a feature space. Let $R = \{p_1, p_2, \dots, p_r\}$ be a condensed set, called also representation set, of r optimised representatives from X. Assume a dissimilarity measure d, computed directly from the instances represented in a feature space. A *dissimilarity* representation of an example $x \in X$ is defined as a *vector of dissimilarities* computed between x and the representatives from R, i.e. $D(x, R) = [d(x, p_1), d(x, p_2), \dots, d(x, p_r)]$. Hence, for the set X, it extends to an $n \times r$ dissimilarity matrix $D(X, R)$. In general, a dissimilarity representation can be derived between strings, graphs, text documents, vectors or any other initial representation. Here, we focus on feature vector spaces, as some of our condensing methods work there, by selecting some instances or generating new prototypes.

The dissimilarity matrix $D(X, R)$ is interpreted as a data-dependent mapping $D(\cdot, R)\colon X \to \mathbb{R}^r$ from some representation X to a *dissimilarity space*, defined by the set R. This is treated as a vector space, in which each dimension corresponds to a dissimilarity to a representative from R, i.e. $D(\cdot, p_i)$. The advantage of such a representation is that any traditional classifier operating in vector spaces can be used [Pękalska & Duin(2002), Pękalska(2005), Pękalska et al.(2006)].

A vector $D(\cdot, p_i)$ defined by the dissimilarities to the representative p_i can be interpreted as an attribute. This follows since the dissimilarities to p_i will differ depending on the class membership. If the measure is metric and the dissimilarity $d(p_i, p_j)$ is small, then $d(x, p_i) \approx d(x, p_j)$ for other instances x. This is guaranteed by the backward triangle inequality [Pękalska(2005)]. As a result, only one of them, either p_i or p_j can be chosen as a representative. However, a small number of representatives might be insufficient to represent the data variability if the classes have different spreads. This occurs, when in a feature space one class is represented by a compact cloud of points, while the other class is elongated. It can be, therefore, more beneficial to inspect the vectors of dissimilarities to the entire set R, instead of looking at the nearest neighbourhood only. If the instances x and y are similar, then their dissimilarity vectors $D(x, R)$ and $D(y, R)$ should be correlated, hence lying close in a dissimilarity space.

2.5.2 Classification in Dissimilarity Spaces

In the case of small condensed sets or non-representative training sets, instead of using the 1-NN rule, a better generalisation can be achieved by a classifier built in a *dissimilarity space*. $D(T, R)$ is a representation of a vector space, called dissimilarity space, where each dimension corresponds to the dissimilarity to a representative from the set R [Pękalska(2005)]. The advantage of such a representation is that any traditional classifier operating in vector spaces can be used.

Justification for the Construction of Classifiers in Dissimilarity Spaces

A justification for the construction of classifiers in dissimilarity spaces is as follows. The property that dissimilarities should be small for examples resembling each other in practice, i.e. belonging to the same class, and large for distinct examples, gives a possibility for a discrimination. Thereby, the vector of dissimilarities $D(\cdot, p_i)$ can be seen, as explained in Section 2.5.1, as an attribute or a feature, as the construction of a dissimilarity space for a metric distance is justified [Pękalska(2005)]. One may wonder what the added value of such a representation over feature-based representation is, if the traditional classifiers designed for vector spaces may be applied in the end. Firstly, the strength of a dissimilarity representation relies on its flexibility to encode different characteristics of the data, so it is a representation where object properties can be captured more adequately. The dimensions of a dissimilarity space are defined by the dissimilarities to some instances, derived according to a specified measure. Hence, they convey homogeneous type of information. In that sense, the dimensions are equally important. This is not valid for a general feature-based representation, where features may have different characters and ranges (e.g. weight and length). Secondly, a dissimilarity measure already possibly encodes the object structure and other characteristics, the designed classifiers might be chosen as to be simple, e.g. linear models.

Defining a well-discriminating dissimilarity measure for a non-trivial learning problem is difficult. Designing such a measure is equivalent to defining good features in a traditional classification problem based on features. If a good measure can be found and a training set is representative, then the k-NN rule is expected to perform well. The decision of the k-nearest neighbour is based on local neighbourhoods and it is, in general, sensitive to noise. It means that k nearest neighbours found might not be the best representatives for making a decision to which class an object should be assigned. In cases of a small or non-representative training set, a better generalisation can be achieved by a more global classifier built in a dissimilarity space.

For instance, a linear classifier in a dissimilarity space is a weighted linear combination of dissimilarities between an object and the representation examples. The weights are optimised on the training set, and large weights emphasise instances which play an essential role during discrimination. By doing this, a more global classifier can be built, by which its sensitivity to noise representative examples is reduced. Bibliography [Paclik & Duin(2003)A, Pękalska & Duin(2002), Pękalska et al.(2002)] confirms that a linear or quadratic classifier can often generalise better than the k-NN rule, especially for a small representation set R.

Bayesian Decision Theory

Bayes formula (2.11) is often used for calculating the posterior probability.

$$P(\theta_j|x) = \frac{P(x|\theta_j)P(\theta_j)}{p(x)} \tag{2.11}$$

If we assume that θ_j is a class label and that x is a vector representation of an object, the above Bayes rule is used in pattern recognition. In such a way, observing the value of x we can convert the prior probability $P(\theta_j)$ to the posterior probability $P(\theta_j|x)$. If the likelihood of θ_j with respect to x, $P(x|\theta_j)$, is large, it is more likely that x belongs to the class θ_j (although the prior $P(\theta_j)$ plays the role here as well). Given J classes, and deriving all class posterior probabilities, $P(\theta_j|x)$, $j = 1, 2, \ldots, J$, the maximum posterior probability indicates the most probable class.

Given $J = 2$ classes, θ_1 and θ_2, the error probability is shown in (2.12). It is assumed that x is given, so the probability error is:

$$P(e|x) = \begin{cases} P(\theta_1|x) & \text{if } x \in \theta_2 \\ P(\theta_2|x) & \text{if } x \in \theta_1 \end{cases} \tag{2.12}$$

The average error probability is given by equation (2.13).

$$P(e) = \int_{-\infty}^{+\infty} P(e|x)p(x)\,dx = \int_{x\in\theta_1} P(\theta_2|x)p(x)\,dx + \int_{x\in\theta_2} P(\theta_1|x)p(x)\,dx \tag{2.13}$$

If for every x it is ensured that $P(e|x) << 1$, then the integral (2.13) must be as small as possible. From this reasoning, the *Bayes decision rule* says:

$$\begin{cases} \text{Decide}\,\theta_1, & \text{if}\,P(\theta_1|x) > P(\theta_2|x) \\ \text{Decide}\,\theta_2, & \text{otherwise} \end{cases} \tag{2.14}$$

Linear and Quadratic Bayes Normal Classifiers

In the case of small condensed sets or non-representative training set, instead of using the 1-NN rule, a better generalisation can be achieved by a classifier built in a dissimilarity space. Many traditional decision rules can be applied there, however, linear and quadratic classifiers perform well [Paclik & Duin(2003)A, Pękalska & Duin(2002), Pękalska et al.(2002), Pękalska(2005)]. Such classifiers are weighted linear (quadratic) combinations of the dissimilarities $d(x, p_i)$ between a given object x and the representatives p_i. Although the classifiers are trained on $D(X, R)$, the weights are still optimised on the complete training set.

It has been found out that Bayesian classifiers, i.e. the linear and quadratic normal density based classifiers, perform well in dissimilarity spaces, as seen in the bibliography [Paclik & Duin(2003)A, Paclik & Duin(2003)B, Pękalska & Duin(2002), Pękalska et al.(2002), Pękalska & Duin(2004)]. In general, these classifiers assume that each class can be described by a normal distribution. This means that the likelihood $P(D(x, R)|\theta_j)$ (corresponding to the likelihood $P(x|\theta_j)$ in (2.11)) is estimated by a normal distribution. Prior probabilities are usually estimated based on the class frequencies. Each object is assigned to the class which yields the highest posterior probability $P(\theta_j|D(x, R))$. If we assume that all classes are generated from the same normal distribution (with the covariance matrix determined by the average of the class covariance matrices), this leads to a linear classifier. If we assume that each class may come from a different normal distribution, as a result, a quadratic classifier is obtained.

Based on (2.14), for a two-class problem, a linear normal density based decision function (NLC), defined on the representation set R, is given by:

$$f\left(D(x, R)\right) = [D(x, R) - \frac{1}{2}(m_{(1)} + m_{(2)})]^T\, S^{-1}\,(m_{(1)} - m_{(2)}) + \log\frac{p_{(1)}}{p_{(2)}} \qquad (2.15)$$

and the quadratic function (NQC) becomes:

$$\begin{aligned} f(D(x, R)) \;&=\; \sum_{i=1}^{2} \frac{(-1)^i}{2}\,(D(x, R) - m_{(i)})^T\, S_{(i)}^{-1}\,(D(x, R) - m_{(i)}) \\ &+\; \log\frac{p_{(1)}}{p_{(2)}} + \frac{1}{2}\log\frac{|S_{(1)}|}{|S_{(2)}|}, \end{aligned} \qquad (2.16)$$

where $m_{(1)}$ and $m_{(2)}$ are the mean vectors, S is the sample covariance matrix (average of the class covariance matrices) and $S_{(1)}$ and $S_{(2)}$ are the estimated class covariance matrices, all computed in the dissimilarity space $D(X, R)$. $p_{(1)}$ and $p_{(2)}$ are the class prior probabilities. When the covariance matrices become singular, they can be regularised. In the implementation of these classifiers for multi-class problems [Duin et al.(2004)], the normal-density functions are estimated per each class and the final decision is based on the maximum a posteriori probability. Note that this is equivalent to (2.14) if two classes are present.

For equal class priors, the Linear Normal density based Classifier (NLC) is equivalent to the Fisher Linear Discriminant (FLD) obtained by maximising the *Fisher criterion*, i.e. $\max_w \frac{w^T S_B w}{w^T S_W w}$, where S_B is the between-class scatter and $S_W = J$ is the within-class scatter [Duda et al.(2001)]. For a dissimilarity representation $D(T, R)$, the Fisher linear discriminant is found as

$$f(D(x,R)) = (m_1 - m_2)^T S_W^{-1} D(x,R) - \frac{1}{2}(m_1 + m_2)^T S_W^{-1}(m_1 - m_2) \tag{2.17}$$

A multi-class Fisher linear discriminant is derived in the one-against-all strategy [Duin et al.(2004)].

Other Dissimilarity Classifiers

Although we prefer linear and quadratic Bayes normal classifiers due to the reasons explained before, several kinds of classifiers have been defined to be used in dissimilarity representations. Some of them are also presented next.

The Strongly Regularised Quadratic Classifier (SRQC) is similar to the Quadratic Normal density based Classifier (NQC) defined in the subsection above, but with a difference in the regularisation used, which is expressed by diminishing the influence of covariances with respect to variances. This means that each class covariance matrix is estimated as $S_i^k = (1-k)S_i + kp(\theta_i)$ diag (S_i), where $k \in [0,1]$. If $k = 0$, then the classifier reduces to the quadratic normal density based classifier, while if $k = 1$, then the classifier becomes the scaled nearest mean linear classifier [Fukunaga(1990), Skurichina(2001)]. So, the variation of k corresponds to a change between these two extremes.

The Support Vector Machine (SVM) is the hyperplane $f(x) = w^T x + w_0$ minimising the norm $\|w\|^2$. The linear support vector machine classifier is expressed as follows:

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i (x, x_i) + \alpha_0 \tag{2.18}$$

where $(x, x_i) = x^T x_i$ is the dot product operation and α_i are nonnegative values determined by maximising the margin $\frac{1}{2\|w\|^2}$. Note also that $w = \sum_{i=1}^{n} \alpha_i y_i x_i$. Since in the solution by a quadratic programming many α_i appear to be zero, only the instances corresponding to non-zero weights, the support vectors, contribute to the classifier. To introduce a support vector machine in a dissimilarity space, one needs to build it on $D(T, R)$. It is then a straight-forward implementation. In the most simple, linear case, it leads to the kernel K consisting of the elements $K_{ij} = \langle D(x_i, R), D(x_j, R)\rangle$. Therefore, in the formulation of a linear support vector machine, in the training step, K becomes $K = DD^T$. Other positive definite kernels can be used as well. In such a case, however, a sparse solution, provided by the method, is obtained in the complete dissimilarity space $D(\cdot, R)$. It means that for the evaluation of new objects, still the dissimilarities to all representatives from R should be computed.

Linear Programming (LP) machines solution is similar to an adaptation of the support vector machine for feature representations defined with the linear programming machines [Schölkopf et al.(2000), Smola et al.(1999)]. From the computational point of view, such a linear programming classifier is advantageous for two-class problems, since for new objects, only the dissimilarities to the objects from the condensed set have to be determined. Since multi-class problems are tackled by a number of two-class problems, in such cases, the reduction of the training set to a condensed set might be insignificant for the combined results.

The k-NN method constructed in a dissimilarity space relies on computing new dissimilarities (e.g. Minkowski distances) between object representation $D(x, R)$ in such a space. This means that indirectly another dissimilarity representation is built over the given one.

The Parzen classifier models the class conditional probabilities $P(D(\cdot, R)|\theta_i)$ by density kernel estimation, here, the normal density function. Let σ_i be the smoothing parameter in the i-th dimension. The posterior probability of the class θ_j is estimated as:

$$P(D(x, R|\theta_j) = \frac{1}{n_j} \sum_{i=1}^{n_j} \frac{1}{\sqrt{2\pi} \prod_k \sigma_k} e^{-\frac{1}{2\sigma_i}(D(x,R)-D(x_i,R))(D(x,R)-D(x_i,R))^T} \tag{2.19}$$

Part II

Contributions

Chapter 3

Non-Adaptive Condensing Algorithms

Contents

As explained in Chapter 2, the weakness of the traditionally used k-NN classification rule is its high computational cost. To solve this problem, some schemes are used in order to reduce the data in the Training Set (TS). The main goal of this thesis is to define an approach which finds a well-balanced Condensed Set (CS), that is, of an appropriate size. We would like to accelerate the application of the k-NN rule without a degradation of the classification accuracy, or at least without a considerable decrease.

A small set of representatives has the advantage of a low computational cost and small storage requirements for a similar or sometimes even an improved classification performance. Various ways of designing a proper set of examples have been studied. Two families of such optimisation procedures are editing and condensing. As these methods lead to reduce the training set size, they are generally referred to as *prototype selection* methods [Devijver(1982)].

Editing is the step in the learning process in charge of increasing the accuracy of predictions, when there is noise in the training data. A basic editing algorithm removes noisy instances, as well as close border instances, eliminating a possible overlap between regions from different classes and leaving smoother decision boundaries. Wilson introduced the first editing method [Wilson(1972)] (see Subsection 2.2.1).

The *condensing* step aims at selecting or generating a small subset of instances or prototypes without a significant degradation of classification accuracy, in order to reduce both storage and time required to process the selected data set. Within the condensing perspective, the many existing proposals can be categorised into two main groups: firstly, the selective schemes merely select a subset of the original instances [Aha et al.(1991), Dasarathy(1994), Tomek(1976), Toussaint et al.(1985), Hart(1968)] and secondly, the adaptive schemes modify them [Chen & Jozwik(1996), Sánchez(2004), Kohonen(1995), Chang(1974)]. The components of the first group are known as non-adaptive, or selective, condensing algorithms and their common feature is that they never modify an instance. On the contrary, they just choose those that would better represent the examples in the original training set. The components of the second group are known as adaptive condensing algorithms, and they are introduced in Chapter 4.

In this thesis, some new approaches to training set size reduction are presented. These schemes, in general, basically consist of defining a small number of prototypes that represent all the original instances. Although the goal of the algorithms proposed here is to obtain a strongly reduced training set, the performance is always taken into account. This chapter focuses on the problem of appropriately reducing the training set size by selecting a subset of already existing instances. Like the Hart's scheme (see Subsection 2.3.1), the new approaches presented in this chapter belong to the family of selective methods. The control over the condensed set final size depends on the process itself.

The primary aim of the proposal presented in this chapter is to obtain a considerable size reduction rate, but without an important decrease in classification accuracy. Therefore, the idea behind is to cover the whole class with the areas of influence of some instances or prototypes chosen for this purpose. This heuristic is based on the next intuitive idea: in data reduction, the selection of some representatives covering the complete area of each class favours the classification accuracy in results for samples of these classes. In addition, once the area of a class is covered by the areas of the most effective representatives, the smaller number of them is selected, the better the final result, in order to reduce the data. Therefore, in order to choose the best representatives, and small in number, some algorithms are introduced and empirically evaluated.

For that purpose, the Nearest Centroid Neighbour (NCN) rule [Chaudhuri(1996), Sánchez et al.(1997)B] is used. By means of this rule a geometrical distribution of instances in the neighbourhoods is chosen to surround the considered sample. This geometrical distribution of instances in the training set can become even more important than just the distances between them, in contrast to the idea represented e.g. in the Hart's algorithm. Until now, the NCN rule has been used for classification

[Sánchez et al.(1997)B], obtaining better results than other traditional classification rules, i.e. the k-NN. The new point of view is that it is here used for condensing purposes.

Next sections explain the contribution of this thesis with respect to the family of selective condensing algorithms. Section 3.1 describes the NCN classification rule, which is used in the condensing algorithms introduced in the following sections. The algorithms *MaxNCN* and *Reconsistent* proposed by us, are respectively presented in sections 3.2 and 3.3. In both sections, some other new algorithms with a strong relation to *MaxNCN* and *Reconsistent*, respectively, are presented and their convenience is discussed.

3.1 Nearest Centroid Neighbour Rule

The main idea we try to develop in the algorithms introduced in this thesis is to cover the complete area of a class with the areas of influence of some instances or prototypes chosen for this purpose. Therefore, in order to choose effective representatives and in small quantity, for the several areas of a class, their geometrical distribution is considered. Such a geometrical distribution among examples in a training set can become even more important than just the distances between them. In this sense, the so-called *surrounding neighbourhood-based rules* [Sánchez et al.(1997)B] try to obtain more suitable information about instances in the training set and specially, for those being close to the decision boundaries. This can be achieved by taking into account not only the proximity of examples to a given input sample but also their *symmetrical distribution* around it.

Therefore, the surrounding neighbours of a sample should satisfy the two complementary conditions:

1. Distance criterion: the neighbours should be as close as possible to the object.
2. Symmetry criterion: the neighbours should be distributed as homogeneously as possible around the sample.

Chaudhuri [Chaudhuri(1996)] proposed a neighbourhood definition, the Nearest Centroid Neighbourhood (NCN) concept, that can be viewed as a particular realization of the surrounding neighbourhood. It is found as follows. Let x be a given sample whose k nearest centroid neighbours should be found in a training set $X = \{x_1, \ldots, x_n\}$. These k neighbours can be searched for through an iterative procedure in the following way:

1. Choose the first nearest neighbour of x, q_1, as its first nearest centroid neighbour.

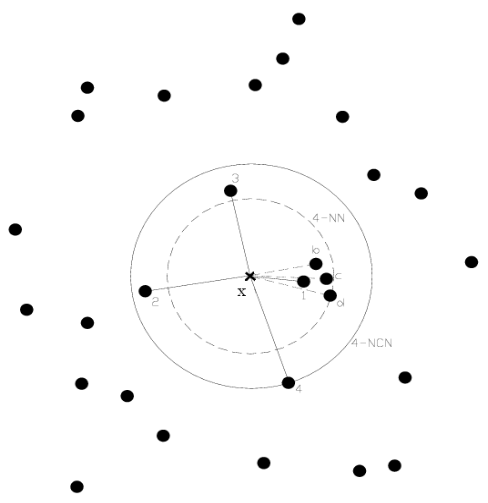


Figure 3.1: Illustration of the NCN concept.

2. Define the i-th nearest centroid neighbour, q_i, $i \geq 2$ such that the centroid of this and the previously selected neighbours, $q_1, \ldots, q_i$ are the closest to x.

The neighbourhood obtained by this algorithm satisfies some interesting properties that can be further used to reduce the training set size by selecting examples. In particular, it is worth mentioning that the NCN search method is incremental and that the instances around the given sample x have a geometrical distribution that tends to surround x, thus compensating the distribution of instances around x. It is also important to note that in general, the region of influence of the NCN rule is bigger than that of the NN rule. This can be observed in Figure 3.1, where the regions defined by the 4-nearest centroid neighbours and the 4-nearest neighbours are shown for the given object x.

3.2 MaxNCN

The *MaxNCN* technique is based on the concept of *nearest centroid neighbourhood* and relies on the NCN search algorithm, as presented above. A subset of instances is selected from the training set to guarantee their optimal geometrical distribution with respect to their nearest centroid neighbours. The use of the nearest centroid neighbourhood of a given sample can provide local information about the shape of the *probability class distribution*, which depends on the nature and class of its

Algorithm 3.1: Pseudo-code of the *MaxNCN* algorithm.

```
for $i = each\_instance(TS)$ do
  $neighbours\_number[i] = 0$
  $neighbour = next\_NCN(i)$
  while $neighbour.class == i.class$ do
    $neighbours\_vector[i] = Id(neighbour)$
    $neighbours\_number[i] + +$
    $neighbour = next\_NCN(i)$
  end while
end for
while $Max\_neighbours() > 0$ do
  $EliminateNeighbours(id\_Max\_neighbours)$
end while
```

nearest centroid neighbours, that is, on the instances in its surrounding area.

The rationale behind this approach is that the instances belonging to the same class are located in a neighbouring area and can be replaced by a single representative without significantly affecting the original boundaries. The main reason to employ the NCN rule instead of the NN is to benefit from its properties:

1. the NCN rule covers a bigger region than the NN rule,
2. the nearest centroid neighbours are located in the area of influence around a given sample, which is compensated in terms of their geometrical distribution.

MaxNCN

Initially, all training instances are considered as representatives. The algorithm attempts to replace a group of neighbouring examples of the same class by a representative. In order to decide which group of instances should be replaced, the concept of *nearest centroid neighbourhood* here used is introduced.

Definition The *nearest centroid neighbourhood* of an instance x is considered to be the set of neighbours of x, calculated by means of the NCN rule until reaching a neighbour with a class label different from that of x.

Definition The instances belonging to the *nearest centroid neighbourhood* of x are their *nearest centroid neighbours.*

Therefore, for each example x in the training set its *nearest centroid neighbourhood* is computed. The instance with the largest number of *nearest centroid neighbours*

Algorithm 3.2: Pseudo-code of the *Iterative MaxNCN* algorithm.

```
**while** $eliminated\_instances > 0$ **do**
  **for** $i = eachinstance$ **do**
    $neighbours\_number[i] = 0$
    $neighbour = next\_NCN(i)$
    **while** $neighbour.class == i.class$ **do**
      $neighbours\_vector[i] = Id(neighbour)$
      $neighbours\_number[i] + +$
      $neighbour = next\_NCN(i)$
    **end while**
  **end for**
  **while** $Max\_neighbours() > 0$ **do**
    $EliminateNeighbours(id\_Max\_neighbours)$
  **end while**
**end while**
```

is defined as a representative of its corresponding group. Since this group lies in the area of influence defined by the nearest centroid neighbourhood distribution, all its members can now be removed from the training set.

Next, given the remaining examples, the algorithm updates the number of their *nearest centroid neighbours* (if some were previously eliminated as belonging to the group of an already existing representative). This is repeated until there is no group of instances to be replaced by a representative. This basic scheme is called *MaxNCN*, and it is presented in Algorithm 3.1.

Iterative MaxNCN

A straightforward extension of the *MaxNCN* algorithm consists of iterating the general process until no more elements were removed from the training set. So, when no more instances have a neighbourhood to represent, new neighbourhoods are calculated among the non-eliminated examples. Algorithmically, the *iterative MaxNCN* version can be presented as Algorithm 3.2 shows.

Iterative kNeighbours

Iterative kNeighbours is a similar approach to the *Iterative MaxNCN* version. The main difference refers to the number of neighbours allowed to be represented by a prototype, k. One of its main properties is that the limit of neighbours can be selected. We specifically chose the value for k depending on the training set size: a percentage. Algorithmically, it can be represented as in Algorithm 3.3.

Algorithm 3.3: Pseudo-code of the *Iterative kNeighbours* algorithm.

```
while $eliminated\_instances > 0$ do
  for $i = eachinstance$ do
    $neighbours\_number[i] = 0$
    $neighbour = next\_NCN(i)$
    while $neighbour.class == i.class$ and $neighbours\_number[i] < k$ do
      $neighbours\_vector[i] = Id(neighbour)$
      $neighbours\_number[i]++$
      $neighbour = next\_NCN(i)$
    end while
  end for
  while $Max\_neighbours() > 0$ do
    $EliminateNeighbours(id\_Max\_neighbours)$
  end while
end while
```

3.3 Reconsistent

The main goal of the algorithms presented in this section is the fact that they are consistent (see Definition 2.3.1), or almost consistent in the case of *Reconsistent*. That is, if the classification error estimated by assigning all instances from the training set to the class of its nearest neighbour in the condensed set, is 0; see [Dasarathy(1991), Hart(1968)]. Therefore, given a set of instances representing the class distribution, the classification rate of the training set can be used to measure the consistency of this set. As the consistent condition is maintained, the smaller the number of representatives in the condensed set, the better the final result. Note, however that algorithms presented in this section do not iterate until a maximum reduction is obtained as it is not the main objective in this section.

The approaches presented in these section are important modifications of the *MaxNCN* algorithm towards obtaining a consistent condensed set [Lozano(2004)A]. The primary idea is that the consistency of a subset with respect to the training set should lead to a better classification. By using the *MaxNCN* algorithm some instances close to the decision boundaries are possibly removed because of the order in which instances are taken during the condensing process. Different approaches implemented by us try to address this issue. Two of them are presented in this

Algorithm 3.4: Pseudo-code of the *Consistent* algorithm.

```
for $i$ = $each\_instance(TS)$ do
   $neighbours\_number[i] = 0$
   $neighbour = next\_NCN(i)$
   while $neighbour.class == i.class$ do
      $neighbours\_vector[i] = Id(neighbour)$
      $neighbours\_number[i] + +$
      $neighbour = next\_NCN(i)$
   end while
end for
while $Max\_neighbours() > 0$ do
   $EliminateNeighbours(id\_Max\_neighbours)$
end while
$count = 0$
for $i$ = $each\_instance(TS)$ do
   if $Classify(i)! = i.class$ then
      $incorrect\_class[count + +] = i$
   end if
end for
$AddToCondensedTS(incorrect\_class[])$
```

section.

Consistent

The simplest method applies first *MaxNCN* and, after that, estimates the class of each instance in the training set by the use of the NN rule as a classifier, using the reduced set. Every instance misclassified is added to the reduced set in order to create the resulting condensed set. This algorithm has been named *Consistent*. It can be represented as shown in Algorithm 3.4.

Reconsistent

The *Reconsistent* algorithm [Lozano(2004)A] is a modification of the *Consistent* version. The aim now is to reduce the condensed set size obtained by using the *Consistent* algorithm.

The procedure starts by applying the *MaxNCN* technique to a training set, yielding a reduced set, R. Then, each example in the training set is tested by the 1-NN rule with respect to the set R, as we do with the *Consistent* approach. The main difference is that all these misclassified instances form a new group, which is condensed using the set R as a reference set. In the end, this new condensed misclassified set is

Algorithm 3.5: Pseudo-code of the *Reconsistent* algorithm.

```
for $i = each\_instance(TS)$ do
   $neighbours\_number[i] = 0$
   $neighbour = next\_NCN(i)$
   while $neighbour.class == i.class$ do
      $neighbours\_vector[i] = Id(neighbour)$
      $neighbours\_number[i]++$
      $neighbour = next\_NCN(i)$
   end while
end for
while $Max\_neighbours() > 0$ do
   $EliminateNeighbours(id\_Max\_neighbours)$
end while
$count = 0$
for $i = each\_instance(TS)$ do
   if $Classify(i)! = i.class$ then
      $incorrect\_class[count++] = i$
   end if
end for
for $i = each\_instance(incorrect\_class[])$ do
   $neighbours\_number\_inc[i] = 0$
   $neighbour\_inc = next\_NCN\_inc(i)$
   while $neighbour\_inc.class == i.class$ do
      $neighbours\_vector\_inc[i] = Id(neighbour\_inc)$
      $neighbours\_number\_inc[i]++$
      $neighbour\_inc = next\_NCN\_inc(i)$
   end while
end for
while $Max\_neighbours\_inc() > 0$ do
   $EliminateNeighbours\_inc(id\_Max\_neighbours\_inc)$
end while
$AddCondensedIncToCondensedTS()$
```

added to R, resulting in the final condensed set. This set is said to be almost consistent, as its consistency is not tested a second time. The *Reconsistent* procedure is presented in Algorithm 3.5.

Chapter 4

Adaptive Condensing Algorithms

Contents

In Chapter 3 some non-adaptive condensing schemes have been introduced. A different family of techniques within the condensing perspective is the one composed of those algorithms that generate new prototypes [Sánchez(2004), Kohonen(1995), Chen & Jozwik(1996), Chang(1974)]. These techniques are known as adaptive or generative condensing algorithms and they are in charge of creating new prototypes that represent or replace the original examples.

One problem related with the use of original instances from the training set is that there may not be any vector located at the precise point that would make the most accurate learning algorithm. Correspondingly, prototypes can be artificially generated to be placed exactly where they are needed, by means of an adaptive condensing algorithm.

Two generative algorithms have already been described in Chapter 2. They are the Chen's algorithm, and the LVQ approach. Briefly, the Chen's scheme split progressively the Training Set (TS), getting some clusters that finally will be replaced by new generated prototypes. LVQ, on the contrary, modifies the already existing instances favouring the ones that belong to the same class to be nearer than they originally were.

This chapter focuses on the problem of appropriately reducing the training set size by artificially generating a subset of prototypes. In order to reduce the training set size while maintaining a similar behaviour of the classifier (i.e. classification accuracy), we accurately select the position of each new prototype depending on its area of influence. So, we try to completely cover a possible area, or areas, where a class is located. That is to say, the area, or areas, in which a sample of a class can be situated, should be perfectly defined by the addition of the area of influence of each representative belonging to that class.

The main achievement presented in this chapter is the introduction of the so-called *MixtGauss* algorithm, which generates new prototypes to represent the original instances. In order to achieve this, it uses the class-conditional *probability density function* (*pdf*) of the spatial class distribution, and *mixtures of Gaussians*. The idea behind the algorithm is to replace a cluster that could be modelled by a Gaussian distribution, just by the centre of this Gaussian.

From now on, this chapter presents the contribution of this thesis to the family of adaptive condensing algorithms. Section 4.1 presents two new generative condensing algorithms based on the Nearest Centroid Neighbour (NCN) rule, *Centroide* and *WeightedCentroide*, along with a discussion of their convenience. Section 4.2 takes care of introducing a new adaptive approach based on mixtures of Gaussians, *MixtGauss*, and also a number of extensions. Therefore, firstly, Subsection 4.2.1 introduces the reader into the topic of mixtures of Gaussians in multi-modal class distributions. Secondly, in Subsection 4.2.2 the stages of the scheme (and of its different versions) are specifically described, and the *MixtGauss* algorithm is finally presented.

4.1 Adaptive Condensing Algorithms Based on the NCN rule

An adaptive condensing algorithm is based upon the abstraction of new prototypes in order to represent the classes which the original instances belong to. After implementing and testing several selecting algorithms, the idea about using a representation of the real instances was becoming a fact. The point was to apply the same idea used before, the nearest centroid neighbourhood concept, except that in this case new prototype locations should be calculated. So, new algorithms for generating prototypes have been implemented. *Centroide* and *WeightedCentroide* are among the more interesting schemes.

4.1.1 Centroide

The main difference between the *Centroide* algorithm (Algorithm 4.1) and the *Iterative MaxNCN* approach (Algorithm 3.2) is that, instead of using an original example

Algorithm 4.1: Pseudo-code of the *Centroide* algorithm.

```
while $eliminated\_instances > 0$ do
  for $i = eachinstance$ do
    $neighbours\_number[i] = 0$
    $neighbour = next\_NCN(i)$
    while $neighbour.class == i.class$ do
      $neighbours\_vector[i] = Id(neighbour)$
      $neighbours\_number[i] + +$
      $neighbour = next\_NCN(i)$
    end while
  end for
  while $Max\_neighbours() > 0$ do
    $AddCentroid(id\_Max\_neighbours)$
    $EliminateNeighbours(id\_Max\_neighbours)$
  end while
end while
```

as a representative of a neighbouring group, the *Centroide* algorithm attempts to replace a number of neighbouring instances by a new prototype, located in their centroid. The rationale behind this is that a new artificial prototype could better represent a neighbourhood because it can be allocated at the best estimated location.

So, the algorithm firstly computes the nearest centroid neighbours of each instance x in the training set until reaching a neighbour from a different class than that of x. In order to decide which group of instances are to be replaced, the one with the largest number of neighbours is chosen. The centroid of this group is calculated and defined as a representative of the neighbourhood. Consequently, all its members can now be removed from the training set. For each instance remaining in the set, the number of its neighbours is updated if some were previously eliminated as belonging to a previously chosen group. This is repeated until there is no group of instances to be replaced by a representative. So, the reduction of the data set is done by introducing new prototypes that replace existing groups.

In order to obtain a significant size reduction, this general process is iterated until no more representatives are removed from the training set. The whole scheme is given in Algorithm 4.1.

Algorithm 4.2: Pseudo-code of the *WeightedCentroide* algorithm.

```
**while** $eliminated\_instances > 0$ **do**
  **for** $i = eachinstace$ **do**
    $neighbours\_number[i] = 0$
    $neighbour = next\_NCN(i)$
    **while** $neighbour.class == i.class$ **do**
      $neighbours\_vector[i] = Id(neighbour)$
      $neighbours\_number[i] + +$
      $neighbour = next\_NCN(i)$
    **end while**
  **end for**
  **while** $Max\_neighbours() > 0$ **do**
    $AddCentroid(id\_Max\_neighbours, neighbours\_number[id\_Max\_neighbours] + 1)$
    $EliminateNeighbours(id\_Max\_neighbours)$
  **end while**
**end while**
```

4.1.2 WeightedCentroide

The *WeightedCentroide* algorithm was developed based on the idea of the *Centroide* algorithm. The only difference between them is that in *WeightedCentroide*, each centroid replacing a group of instances has a weight. The weight of each new prototype is computed in relation to the number of instances that the centroid represents. So, e.g. if in the first iteration a centroid represents a group of six neighbours, this centroid will have a weight of 6 units. In a subsequent iteration, if a new centroid represents three centroids from the previous iteration, the weight of the new one will be the sum of the weight values of the three previous representatives. Consequently, at the end of the process, each representative in the Condensed Set (CS) will have a weight value equivalent to the quantity of these instances from the training set that it really represents. The weight is used in order to place the calculated centroid near those prototypes with a higher weight, that is to say, representing more prototypes. Consequently, the centroids are forced to be nearer the areas which consist of many original instances of the class that they are representing. Algorithm 4.2 provides this scheme.

4.2 An Adaptive Condensing Algorithm Based on Mixtures of Gaussians

Until now, we have used techniques based on geometric information in order to obtain the desired computational complexity reduction. From now on, we base our research in *probability density* estimators. In order to do this, we use the training set to calculate the *probability density* of the examples we find there, as they are the ones we know, and we suppose that they represent the complete classes. The *probability density* estimator we use is a model of mixtures of Gaussians.

The basic algorithm we present in this section is called *MixtGauss.* It is considered in the framework of mixture modelling by Gaussian distributions, while assuming a statistical independence of features. The mean vectors of the optimised Gaussians, whose mixtures are fit to model each of the class distributions, are chosen to be their representatives. In order to obtain the optimised Gaussians from the initial mixtures, the Expectation Maximisation (EM) algorithm is used. The details are given in next sections.

4.2.1 Mixtures of Gaussians in Multi-Modal Class Distributions

In general, we have a training set with n instances, where each instance x is a point, in a d-dimensional feature space, $x = [x_1, \ldots, x_d] \in \Re^d$. Then, for a given sample, the aim is to assign it to the correct class where $\Theta = \{c_1, \ldots, c_J\}$ is a finite set of the J classes in the training set. The different points of the training set follow a spatial class distribution according to their true class-conditional *pdf*s $P(x|c_j)$ and the respective *a priori* probability $P(c_j)$, $c_j \in \Theta$. So, we can say that a vector x can be then optimally classified using the Bayes rule or maximum *a posteriori* probability decision rule based on the knowledge of the components $P(c_j)P(x|c_j)$ for each class c_j.

In practice, these class-conditional *pdf* do not have any underlying structure assumed and no prior knowledge about the shapes of these *pdf*s is required to solve the problem. So, one way to approach this is by estimating a density function of the distribution.

A natural way to deal with a density estimator is to consider a mixture density of components. One approach to retain the capacity of $P(x|c_j)$ is by reflecting the local structure of the distribution by means of mixture components $P_m(x|c_j)$. This density estimator is used in the literature [Novovicova et al.(1996)]. We use it, while assuming the statistical independence of features. In fact, we calculate the estimators in a parametric way. Consequently, the Bayes rule can be used, and each component is estimated by the product of the probabilities in each feature:

$$P(x|c_j) = \sum_{m=1}^{M} \alpha_{m|j} P_m(x|c_j) = \sum_{m=1}^{M} \alpha_{m|j} \prod_{k=1}^{d} N(x_k; \mu_{m|kj}, \sigma_{m|kj}) \tag{4.1}$$

where M is the number of components and $\alpha_{m|j}$ is an *a priori* probability of the m^{th} component in the class c_j. Therefore, the Bayes classifier is:

$$C_{Bayes} = \max_{j=1..J} P(c_j)P(x|c_j) \tag{4.2}$$

where $P(c_j)$ is the *a priori* class probability.

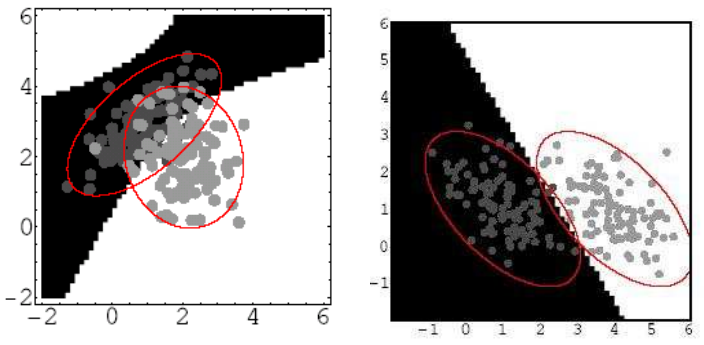
Then, we can consider the characterisation of the distribution like a parametric unsupervised learning problem [Duda & Hart(1973)] using a mixture of components of a multivariate normal distribution. We represent the multidimensional Gaussians as a product of univariate normal distributions, with $\mu_{m|kj}$ and $\sigma_{m|kj}$ as the means and standard deviations, respectively. The EM algorithm [Dempster(1977)] will be here used to estimate these unknown components.

4.2.2 MixtGauss

The final aim of the condensing process proposed here is to obtain a probability distribution representing a class, for every class present in the training set. General shapes of classes as well as the decision boundaries are kept, when each class is described by a mixture of multivariate Gaussian distributions. By the additional assumption of the statistical independence of features, the m^{th} component in a mixture modelling of the class c_j is a multivariate Gaussian distribution expressed by a product of univariate normal distributions, $N(\mu_{m|kj}, \sigma_{m|kj})$. Note that this is equivalent to a multivariate elliptic Gaussian distribution $N(\mu_{m|j}, diag(\sigma_{m|j}))$, parallel to the axes of $\Re^d$.

The multidimensional Gaussian density is a typical model used to describe a probability density. The only parameters to estimate are the means and the covariance matrices. Quadratic bounds (see Figure 4.1 (a)) related to the Mahalanobis distance to each class are obtained by introducing the estimated models in the Bayes test. In addition, if the covariance matrices of the classes are equal, the quadratic terms cancel each other, and the bounds become linear (hyperplanes; see Figure 4.1 (b)).

If the clusters have spherical shapes of the same size, and the classes are equiprobable, the optimal classifier is the nearest neighbour to the mean of each class (see Figure 4.1 (c)); the so-called nearest mean classifier. Therefore, if the algorithm here



(a) (b) (c)

Figure 4.1: Elliptic Gaussians determining quadratic bounds (a) and linear bounds (b), and spheric Gaussians determining linear bounds.

presented was in charge of covering the area of each equiprobable class with spheric-shaped Gaussians having the same size, the centres of influence would exactly be the Gaussian means.

Nevertheless, as the supposed simplifications will never take place in practical cases, we try to learn our classifier for real data by using Gaussians of different sizes and shapes.

In addition, a consistency criterion will be applied to the points representing a class. The more Gaussians are included in the point distribution, the more accurate the representation of the decision boundaries, and consequently, the more accurate the classification. For instance, in the Hart's algorithm, the consistency criterion relies on the correct 1-NN classification of all instances in the training set by using the condensed set.

However, in the method presented here a different criterion, close to the consistency definition given in Hart's algorithm (see Definition 2.3.1), will be employed.

Definition A condensed set R, is said to be *quasi-consistent* with respect to a training set X, if the estimated classification error is (sufficiently) small, when the class for each pattern in X is estimated by the NN rule for R.

Given a set of prototypes representing the class distribution, we can use the classification rate of the training set in order to measure how consistent (according to the definition given above) it is.

As the *quasi-consistent* condition is maintained, the smaller the number of prototypes needed for representing a class in the condensed set, the better the result. On

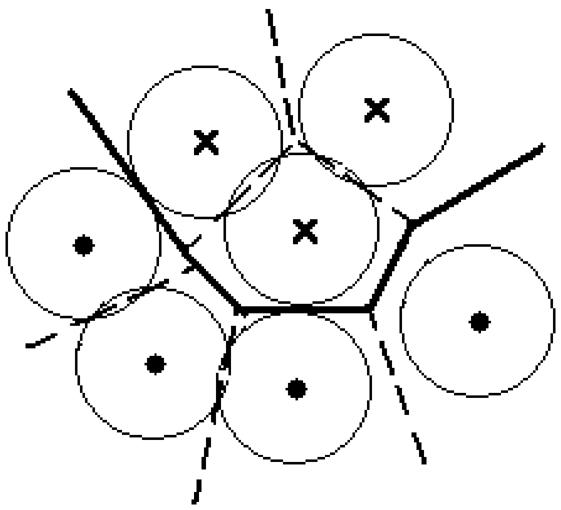


Figure 4.2: Edges of the areas of influence of the equal circle-shaped Gaussians that coincide with Voronoi boundaries.

the other hand, for some applications it is useful to be able to define the condensed set size beforehand.

To look for a condensed point distribution, imagine a problem of two Gaussian distribution classes in a Euclidean space. The best condensed set would be the mean of the Gaussian *pdf* function in each class. Therefore, let us suppose a class distribution that could be modelled by a mixture of Gaussians. The centres of these Gaussians could become the set of prototypes representing the class distribution, that is, the condensed set. Thus, if each class is covered by a mixture of Gaussians, the union of the areas of influence of all Gaussians would cover the complete area where each class is defined.

In addition, assuming equal prior probabilities for all Gaussians, if they are forced to be hyperspherical (same diagonal covariance matrix for all Gaussians with equal variance in every feature), the use of these Gaussians would be equivalent to the minimum distance classifier, 1-NN [Duda et al.(2001)], when associating each Gaussian to a given class. Therefore, the edges between the areas of influence of neighbouring Gaussians would correspond to the Voronoi boundaries (see Figure 4.2).

MixtGauss Initialisation

Note that although the EM algorithm is widely used, one needs to be aware of its drawbacks [Figueiredo & Jain(2002)]. As a method working in local neighbourhoods, it is sensitive to initialisation because the likelihood function of a mixture model is not unimodal. Another important issue in mixture modelling is the selection of the number of components. With too many components, the mixture may over-fit the data, while a mixture with too few components may not be flexible enough to approximate the true underlying model.

We use mixture of Gaussians in order to obtain an estimation of the probability density function. We assume that all the components have the same functional form. For example, they are all d-variate Gaussians, each one being thus fully characterised by the parameter vector. And the more Gaussian components are included in a mixture, the more accurate the representation of the classes and the decision boundaries.

Taking into account these facts, we have to provide a number of Gaussians M to represent each class distribution. This number corresponds to the number of prototypes generated in each class for the final condensed set. At the initial stage, each class is represented by M Gaussians. We present three different initialisations.

1. In the first one, each parameter range is divided by $(M - 1)$ (in this way we obtain as many interval bounds as Gaussians we need). At each step, a Gaussian is represented, with all the parameter values for this step. So, a d-dimensional diagonal, represented by Gaussian centres is obtained. For each Gaussian, the initial variance is set up to a tenth of the range in each dimension.

2. In the second initialisation, each class is represented by M Gaussians located on the mean of the instances from that class. By adding random disturbances $(\frac{Range}{500} * random(0, 1)$, for each dimension/feature), different mean vectors are created, and the Gaussians are shifted away. So, a mixture of Gaussians for representing each class distribution can be obtained.

3. In the third case, the M initial modes are randomly chosen from the training examples. This idea is based on the random distribution of the representations.

Among these three initialisations, the second one was chosen as the best, in order to compare the results to other algorithms. So, in the final *MixtGauss* algorithm the initial components are the M Gaussians located at the class means, randomly perturbed.

MixtGauss Optimisation

In the *MixtGauss* optimisation one estimates the probability density independently for each class, and represents this probability density by the chosen components. The standard method used to fit finite mixture models to the observed data, hence to estimate the parameters of the class distributions, is the well-known EM algorithm [McLachlan & Basford(1988), McLachlan & Krishnan(1997)]. This is a general *Maximum Likelihood* (ML) optimisation procedure for problems with hidden variables or missing data [Dempster(1977)]. So, after the initialisation stage, an iterative optimisation process based on the EM algorithm, is carried out in order to determine an optimal location of the mixtures of Gaussians for each class independently. This iterative optimisation procedure converges to a *maximum likelihood* estimate of the mixture parameters. Accordingly, to fit a mixture of Gaussians to each class distribution, we iterate between the following two steps:

E-step Compute the contributions of the instances x^t (where t is the number of the iteration) to the class c_j, belonging to the set $\{x^1, \dots, x^{N_j}\}$, where N_j is the cardinality of the class c_j. The conditional *pdf* for the m^{th} component is:

$$P_m(x^t|c_j) = \frac{\alpha_{m|j} \prod_{k=1}^{d} N(x_k^t; \mu_{m|kj}, \sigma_{m|kj})}{\sum_{l=1}^{M} \alpha_{l|j} \prod_{k=1}^{d} N(x_k^t; \mu_{l|kj}, \sigma_{l|kj})} \tag{4.3}$$

This function is normalised such that $\sum_{m=1}^{M} \alpha_{m|j} = 1$. The multivariate Gaussians are represented as a product of univariate normal distributions, with the means $\mu_{m|kj}$ and the standard deviations $\sigma_{m|kj}$.

M-step Compute the parameters of the m^{th} component for each value x^t that exists in the class c_j:

$$\alpha_{m|j} = \frac{1}{N_j} \sum_{t=1}^{N_j} P_m(x^t|c_j) \qquad \mu_{m|kj} = \frac{\sum_{t=1}^{N_j} P_m(x^t|c_j) x_k^t}{\sum_{t=1}^{N_j} P_m(x^t|c_j)} \tag{4.4}$$

$$\sigma_{m|kj} = \frac{\sum_{t=1}^{N_j} P_m(x^t|c_j)(x_k^t - \mu_{m|kj})^2}{\sum_{t=1}^{N_j} P_m(x^t|c_j)} \tag{4.5}$$

Both steps are iteratively repeated for each class. The final prototypes of the sought condensed set are the mean vectors of the Gaussian distributions determined by the EM algorithm at the end of the loop in which the process is stopped.

MixtGauss Optimisation Speed Control

Two different implementations in relation to speed control have been tested.

1. The first one does not take into account the differences in the *optimisation speed* of each class. So, the EM optimisation loop is repeated until the stopping criterion is achieved.

2. In the second approach, as we have realised that several speeds are used by the class optimisation, we have tried to control them in order to force the ones with more velocity to wait for the others.

The second version is the one which obtains better results, and it is explained below.

After each iteration of the optimisation process described in the previous subsubsection (Subsubsection MixtGauss Optimisation), the components of each class have been modified. At this point their consistency improvements are tested in order to know whether any of the classes adapt their components in a higher speed than others. The main idea is that if the components of a class converge more quickly than the components of a neighbouring class, the former ones could catch the samples in the borders, some of them belonging to the later class, which is still approaching to its local optimum.

Therefore, this unbalanced speed is detected by testing the classification accuracy for each class at this point of the optimisation process. In order to measure the accuracy, some steps are followed:

1. Firstly, each current mode is taken as being a representative in the new reduced set.

2. Secondly, the labels of the instances in the original training set estimated, and compared to their actual labels.

3. Finally, the classification accuracy is calculated, by using the ratio of the number of correct classifications in that class, to the total number of examples in the same class. If this percentage is deteriorated for a class in comparison to the previous iteration, this class is forced to wait for the next repetition of the loop to move/re-adapt its components.

The EM loop is repeated for each class, and the classification accuracy is tested until the stopping criterion is reached.

MixtGauss Stopping Criterion

The iterative optimisation process described in the previous subsubsections could cause an inadequate overlap between the Gaussian components from different classes, thus deteriorating the final classification accuracy. Therefore, this process should stop at a certain point. Three alternatives are here presented.

1. In the first version, we limit the iterations just by the quantity of repetitions that one wants to run, as it is done in the LVQ algorithm [Kohonen(1996)].

2. In the second version, by applying the *quasi-consistency* criterion presented at the beginning of this Subsection (Subsection 4.2.2), the loop in the *MixtGauss Optimisation* is repeated while classification accuracy increases for any class. When no class improves its results, the process is stopped. The stopping criterion is as follows. After the modification of components of all clases, their *quasi-consistency* is tested. In order to do that, we take the accuracy rates calculated in the way explained in the Subsubsection *MixtGauss Optimisation Speed control*, by using the 1-NN rule. These calculations are compared to the previous iteration ones. The process is stopped when no class yields an increase in performance, trying to obtain the minimum classification error. Meanwhile, the movement of the Gaussians is carried out until a balanced position is found.

3. A new version, the third one, is also proposed by using the Bayes classifier as the estimator for the stopping criterion instead of the 1-NN rule. We name it *MGBayes*.

Finally, the second version of the stopping criterion, using the quasi-consistency criterion based on the 1-NN rule, is adopted for the final *MixtGauss* algorithm, as the results were better in general (anyway quite similar to the Bayes-based consistency criterion version). The complete technique is summarised in Algorithm 4.3 presented below.

MixtGauss Algorithm

Here we present the *MixtGauss* algorithm (Algorithm 4.3), which is based on the process described in the previous section. The option chosen for each step is recalled here in order to clarify the steps that this algorithm consists of.

1. Firstly, in the initialisation step M components are located on the class means, and shifted away by the addition of some random disturbances.

Algorithm 4.3: Pseudo-code of the *MixtGauss* algorithm.

```
(* initialisation *)
for $c = 1..number\_of\_classes$ do
   $mean[c] = CalculateMean(c, TS)$
   for $g = 1..number\_of\_Gaussians\_per\_class$ do
      $Gaussians[c, g] = mean[c] + RandomDisturbance()$
   end for
end for
$current\_accuracy = CalculateAccuracy()$
(* optimisation *)
repeat
   $previous\_Gaussians[c, g] = Gaussians[c, g]$
   $Gaussians[c, g] = EMstep()$
   $previous\_accuracy = current\_accuracy$
   $current\_accuracy = CalculateAccuracy()$
   $classes\_improve = 0$
   for $c = 1..number\_of\_classes$ do
      if $current\_accuracy[c] > previous\_accuracy[c]$ then
         $classes\_improve = classes\_improve + 1$
      else
         if $current\_accuracy[c] < previous\_accuracy[c]$ then
            for $g = 1..number\_of\_Gaussians\_per\_class$ do
               $Gaussians[c, g] = previous\_Gaussians[c, g]$
            end for
         end if
      end if
   end for
until $classes\_improve == 0$
```

2. Secondly, the EM optimisation is applied to each class independently; see Formula 4.3 and Formula 4.4.

3. When an iteration of the EM loop is finished for every class, the error is compared to the error from the previous repetition of the loop in order to recognise if a class is moving its nodes faster than others. If accuracy have been decreased for any of the classes, it is forced to wait until the rest of the classes iterates on the EM loop once more.

4. If no class improves its classification accuracy ratio, the balance position have

already been reached and the stopping criterion forces the process to stop.

The process introduced can be viewed as an adaptive condensing scheme, in which the prototypes of the resulting set will correspond to the centres of the final Gaussians. Algorithmically, it can be summarised as it is shown in Algorithm 4.3.

MixtGauss Convergence Issues

It is known that the application of the EM algorithm over a mixture of Gaussians representing a single class has a convergent behaviour. Based on this fact, we are going to show that the *MixtGauss* algorithm is convergent too.

Let us have J different classes in a database. Thus, one independent EM process is run for each class, in which the components of each class are independently adapted, as the EM iterations are completely separated in time and have different information for each class. At the end of each iteration of the EM for every class, an external test in charge of calculating the improvement of the classification accuracy for each class is run. Depending on the results of this external test, the state of the EM process for a class could be set to *stand by* or *carry on*. The state *stand by* forces the EM process for a class to maintain its position and variables, while EM processes for other classes continue iterating.

Therefore, as the list of states for a class, as well as the EM algorithm process for each class, is not changed at all and act independently of each other classes, then, the convergence of the EM process of every class is warranty by the convergence of the EM algorithm itself. Thus, the *MixtGauss* algorithm presents a convergent behaviour.

When none of the classes improves its classification accuracy, it is because a balance situation is achieved. This can happen by two reasons: one of them is that the EM process for each class finally converges; the other one is that a quasi-consistency equilibrium point between classes is attained. Independently of which reason has been fulfilled to find the balanced position, the process is stopped, as every mode is situated in a location that optimises the class representation.

Part III

Comparative Analysis and Conclusions

Chapter 5

Comparative Analysis among Condensing Techniques based on the NCN rule

Contents

5.1 Introduction

The usefulness of condensed sets, optimised by various approaches discussed in the previous chapters will now be evaluated in a classification task. Traditionally, the 1-NN rule, assigning an unknown object to the class of its nearest neighbour among the Condensed Set (CS), is used for this purpose. In contrast to its conceptual simplicity, this rule has a good behaviour when applied to non-trivial problems. In fact, the k-NN rule performs as well as any other possible classifier, provided there is an arbitrarily large number of representative prototypes available and the volume of the neighbourhood of each object is arbitrarily close to zero (asymptotically optimal [Dasarathy(1991)]).

From now on, this chapter describes the experimental setup (Section 5.2), presents the quantitative results for the condensing techniques based on the Nearest Centroid

Neighbour (NCN) rule and others (Section 5.3), and draws some conclusions (Section 5.4).

5.2 Experimental Setup

Some new and classical selective and generative algorithms have been presented in the previous chapters. Now they are run, on the eleven databases described in Chapter 1. In order to do that, each database was normalised by a unit variance, and after that, divided in 5 different sets in order to be able to repeat the experiments 5 times for each database, and to obtain the average. It is done in a five-fold cross-validation scheme. See Chapter 1 for details.

As in the case of the Chen's and RSP3 approaches, the new algorithms tested in this chapter (*MaxNCN*, *Iterative MaxNCN*, *Iterative kNeighbours*, *Consistent*, *Reconsistent*, *Centroide* and *WeightedCentroide*) need to be applied in practice to overlap-free (no overlapping between different class regions) data sets. Thus, as a general rule and according to the previously published results [Sánchez(2004), Wilson & Martinez(2000)], the Wilson's editing has been considered to properly remove possible overlapping between the classes. The parameter involved, k, has been obtained in our experiments by performing a five-fold cross-validation experiment using only the Training Set (TS) and computing the average classification accuracies for different values of k and comparing them to the "no editing" option. The best edited set (including the non-edited training set) is thus selected as an input for the different condensing schemes.

The experiments are conducted to compare the condensed sets obtained from the execution of the algorithms *MaxNCN*, *Iterative MaxNCN*, *Iterative kNeighbours*, *Consistent*, *Reconsistent*, *Centroide* and *WeightedCentroide*, among other new approaches, to the Chen's scheme, the RSP3 approach and the Hart's condensing, in terms of both training set size reduction and classification accuracy. The 1-NN rule is used in order to calculate the classification accuracy. All resulting condensed sets are also compared to the original training set.

Each of these algorithms, except for the case of the Chen's condensing, chooses automatically the size of the resulting condensed set. So, it is asked for the same condensed set size as the *MaxNCN* algorithm obtains, as the results show that it reaches a good balance between classification accuracy and size reduction.

5.3 Quantitative Results

This section presents the experimental results. As it is not possible to present all the results in a unique table, a number of them are shown. They report the 1-NN classification accuracy results for the sets in comparison. They represent the original training set and different condensed sets obtained with the classical algorithms explained in Chapter 2, and with the new techniques based on the NCN rule, presented in Chapters 3 and 4.

Table 5.1 reports the 1-NN classification accuracy rate and, in square brackets, the reduction size rate and the equivalent number of instances or representatives, for the original training set and the condensed sets obtained by already existing algorithms: the classical Chen's and Hart's approximations, and the RSP3 version. The average values for each method on the eleven data sets are also included. The highest classification accuracy and the highest size reduction rates for each database and for the average are highlighted in bold. So, among the average classification accuracy rate obtained with the *Original* training set, and with the Chen, Hart and RSP3 condensed sets, the highest one is reached by the RSP3 condensed set. It is to note that the average accuracy rate reached by RSP3 is even slightly higher than the *Original* training set accuracy, while obtaining a reduction of 52.83% of the *Original* training set size. In spite of the fact that for four of the databases (Vehicle, Vowel, Satimage and Texture), the highest accuracy rate is reached by the *Original* training set, even in those cases where the accuracy rates obtained by the RSP3 condensed sets are very near to the *Original* accuracy. The highest average reduction rate in Table 5.1 is obtained by the Chen's algorithm.

Table 5.2 reports the 1-NN classification accuracy rate and, in square brackets, the size reduction rate and the equivalent number of representatives, for the condensed sets obtained by some of the selective algorithms developed in this thesis: the *MaxNCN*, the *Iterative MaxNCN*, the *Iterative kNeighbours*, the *Consistent* and the *Reconsistent*. The average values for each method on the eleven data sets are also included. The highest classification accuracy and the highest size reduction rates for each database and for the average, are highlighted in bold. The highest classification accuracy rate belongs to the *Consistent* condensed set. The highest average reduction rate is reached by the *Iterative MaxNCN* condensed set.

Looking carefully at Table 5.2, it can be observed that although *Iterative MaxNCN* and *Iterative kNeighbours* obtain the best results in reduction, they obtain the worst results in accuracy. On the contrary, *Consistent* and *Reconsistent* reach the best results in accuracy, but they reach the worst results in reduction, in comparison to other algorithms in the table. It is not strange, as it was already to be expected in general: when the higher reduction is achieved, the lower accuracy is reached. Mean-

Table 5.1: Classification accuracy of the 1-NN rule applied to the original training set and the condensed sets obtained with the already existing algorithms: Chen's, Hart's and RSP3 (in %). The corresponding training set size reduction (in %) and the equivalent number of instances or representatives are given in square brackets.

	Original	Chen	Hart	RSP3
Cancer	95.17	**95.75**	94.44	94.44
	[546]	[**95.24**≡26]	[92.12≡43]	[91.03≡49]
Diabetes	70.06	**73.44**	71.62	72.53
	[614]	[**86.97**≡80]	[79.15≡128]	[67.10≡202]
Glass	69.66	62.55	66.37	**70.16**
	[171]	[**59.06**≡70]	[51.46≡83]	[33.33≡114]
Heart	76.26	**78.88**	78.49	78.16
	[216]	[**83.80**≡35]	[76.39≡51]	[68.98≡67]
Liver	62.59	58.86	60.05	**62.65**
	[276]	[**76.81**≡64]	[67.03≡91]	[51.45≡134]
Vehicle	**69.40**	63.94	66.92	68.56
	[677]	[**60.27**≡269]	[50.96≡332]	[23.93≡515]
Vowel	**98.12**	53.53	94.59	97.53
	[422]	[74.41≡108]	[**74.88**≡106]	[22.75≡326]
Wine	94.41	79.10	92.73	**95.49**
	[142]	[**92.96**≡10]	[92.25≡11]	[70.42≡42]
Phoneme	71.00	71.28	70.81	**71.98**
	[4323]	[**91.56**≡365]	[88.60≡493]	[61.21≡1677]
Satimage	**82.96**	76.09	80.77	82.42
	[5148]	[**86.54**≡693]	[84.83≡781]	[54.74≡2330]
Texture	**98.89**	70.87	96.93	98.78
	[4400]	[89.32≡470]	[**91.52**≡373]	[36.23≡2806]
Average	80.78	71.30	79.43	**81.15**
		[**81.54**]	[77.20]	[52.83]

while, *MaxNCN* does not get the highest nor the lowest result for any database in neither accuracy nor reduction. In fact, it is one of the best balanced results:

only a negative percent difference of 3 in comparison to the best result in average accuracy (and a positive percent difference of 9 in average reduction), and a negative percent difference of 12 in comparison to the best result in average reduction (and

Table 5.2: Classification accuracy of the 1-NN rule applied to the condensed sets obtained with the selective algorithms based on the NCN rule: *MaxNCN*, *Iterative MaxNCN*, *Iterative kNeighbours*, *Consistent* and *Reconsistent* (in %). The corresponding training set size reduction (in %) and the equivalent number of representatives are given in square brackets.

	MaxNCN	It.MaxNCN	It.kNeighbours	Consistent	Reconsistent
Canc.	92.0	76.0	78.6	**94.0**	93.4
	[95.2≡26]	[**99.3**≡4]	[98.7≡7]	[90.8≡50]	[93.8≡34]
Diab.	67.7	56.8	57.2	71.5	**71.8**
	[87.0≡80]	[**98.1**≡12]	[96.1≡24]	[75.4≡151]	[81.8≡112]
Glass	64.1	53.3	46.6	**69.4**	**69.4**
	[59.1≡70]	[**73.1**≡46]	[72.0≡48]	[44.4≡95]	[46.8≡91]
Heart	69.3	65.2	62.8	**75.6**	74.8
	[83.8≡35]	[96.8≡7]	[**98.2**≡4]	[72.2≡60]	[77.8≡48]
Liver	58.9	51.0	47.5	**61.5**	61.0
	[76.8≡64]	[96.1≡11]	[**96.4**≡10]	[64.1≡99]	[68.5≡87]
Vehi.	64.4	51.1	55.1	**66.9**	66.0
	[60.3≡269]	[**84.8**≡103]	[81.1≡128]	[46.5≡362]	[49.2≡344]
Vowel	90.8	78.2	72.7	95.3	**95.5**
	[74.4≡108]	[84.9≡64]	[**85.1**≡63]	[69.4≡129]	[71.8≡119]
Wine	91.6	86.7	84.3	91.5	**92.1**
	[93.0≡10]	[**97.9**≡3]	[**97.9**≡3]	[88.0≡17]	[90.9≡13]
Phon.	69.1	59.2	58.4	**71.4**	70.2
	[91.6≡365]	[**98.5**≡64]	[98.5≡67]	[84.3≡679]	[88.9≡480]
Sati.	79.2	57.9	56.5	**80.7**	80.3
	[86.5≡693]	[**98.2**≡94]	[98.1≡97]	[79.8≡1042]	[82.7≡890]
Text.	95.3	74.0	74.8	**97.5**	97.3
	[89.3≡470]	[98.8≡53]	[**98.8**≡52]	[86.0≡617]	[87.8≡535]
Avr.	76.58	64.47	63.14	**79.57**	79.23
	[81.54]	[**93.29**]	[92.79]	[72.82]	[76.35]

+12 in average accuracy).

Comparing Table 5.2 to Table 5.1, in general, the results in accuracy are higher for the already existing algorithms and the *Original* training sets. On the other hand, the results in size reduction are, in general, higher for the algorithms presented in

Table 5.3: Classification accuracy of the 1-NN rule applied to the condensed sets obtained with the generative algorithms based on the NCN rule: *Centroide* and *WeightedCentroide* (in %). The corresponding training set size reduction (in %) and the equivalent number of prototypes are given in square brackets.

	Centroide	WeightedCentroide
Cancer	69.61	69.61
	[98.90≡6]	[98.90≡6]
Diabetes	45.44	49.88
	[**97.88**≡13]	[97.56≡15]
Glass	44.40	44.40
	[74.27≡44]	[74.27≡44]
Heart	62.17	62.17
	[97.69≡5]	[97.69≡5]
Liver	53.14	52.61
	[**96.01**≡11]	[95.65≡12]
Vehicle	48.11	48.95
	[**84.19**≡107]	[83.31≡113]
Vowel	80.85	80.91
	[82.46≡74]	[82.46≡74]
Wine	85.56	85.56
	[97.89≡3]	[97.89≡3]
Phoneme	58.19	58.01
	[**98.03**≡85]	[97.99≡87]
Satimage	58.23	59.59
	[98.08≡99]	[**98.10**≡98]
Texture	72.45	72.65
	[98.80≡53]	[98.80≡53]
Average	61.65	62.21
	[**93.11**]	[**92.96**]

this thesis, aiming at the higher efficiency in reduction than in accuracy.

Table 5.3 reports the 1-NN classification accuracy rate and, in square brackets, the size reduction rate and the equivalent number of prototypes, for the condensed sets obtained by the generative algorithms based on the NCN rule, developed in this thesis (*Centroide* and *WeightedCentroide*). The last line shows the average values for

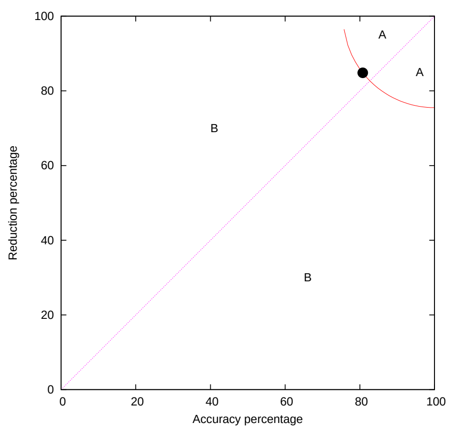


Figure 5.1: Representation of areas, depending on their proximity to the ideal point (and their balance) in comparison to a given point.

each method on the eleven data sets. The results obtained with these two algorithms are quite similar. So, only the size reduction rates higher than the ones from the other algorithm have been highlighted in bold. In the average results we highlighted both results in order to make the reader note that both results are very good ones. In relation to accuracy, no-result is highlighted as none of them is good enough in comparison to those from other tables.

Several comments can be made from the results in these three tables. It is important to note that, in terms of reduction rate, the *Iterative MaxNCN*, *Centroide*, *WeightedCentroide* and *Iterative kNeighbours* eliminate much more instances than any other scheme. As expected, classification accuracy strongly depends on the number of representatives in the condensed set. Correspondingly, the RSP3, *Consistent*, Hart's, and *Reconsistent* algorithms (in this order), as well as the original training set, obtain the highest classification accuracy almost without exception for all the data sets. However, they also retain more representatives than the rest of the algorithms.

Finally, we would like to emphasise that while it is true that three databases get the highest accuracy and highest reduction with the same algorithm, in Table 5.1, whether reduction is compared to the results in any of the other tables, it is lower than the reduction rate for almost every algorithm. It is not by chance that the highest accuracy and reduction rates are not reached at the same time in any database. It is one more proof about the fact that in order to get a higher reduction rate, a classification accuracy rate will be sacrificed. Maybe it is better not to re-

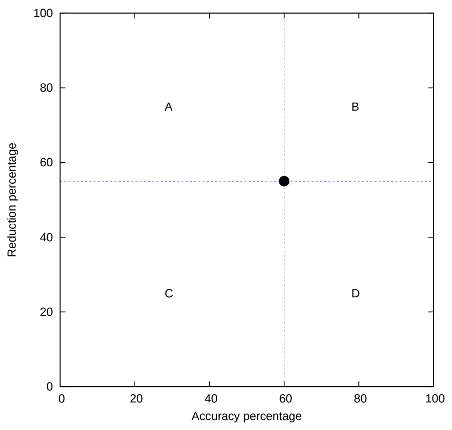


Figure 5.2: Representation of areas depending on their improvements/deteriorations in comparison to a given point.

duce so strongly/dramatically the training set size, however just to reduce until an acceptable cardinality, as judged for each problem.

In order to assess the performance of these two competing goals simultaneously, classification accuracy and size reduction, Figure 5.1 represents the normalised Euclidean distance between a reference pair (accuracy, reduction) and the ideal case (100, 100), which represents 100% accuracy and 100% reduction, in such a way that the "best" approach can be deemed as the one that is nearer (100, 100). So, points in region A correspond to those procedures that have achieved better results (in the sense that they are nearer the ideal point) than the reference algorithm used to draw the arch. Points in region B represent those methods that have obtained worse results. This technique is usually referred to as *Data Envelopment Analysis* [Charnes(1978)]. In figures from 5.3 to 6.1 the nearest algorithm to the ideal point (100, 100), have been chosen as a reference. The approaches closest to the diagonal in these plots are the ones that get the best balance between accuracy and reduction rates.

In addition, Figure 5.2 should be interpreted as follows. Points in region B represent the methods that have obtained a higher reduction rate and also achieved a higher accuracy than a certain reference algorithm. Points in region C correspond to those procedures that have achieved worse results in both accuracy and reduction rates. Finally, points in regions A and D indicate intermediate situations, that is, methods that have improved only one value (either accuracy percentage or reduction rate, respectively). In the plots shown next, the algorithm chosen as a reference is

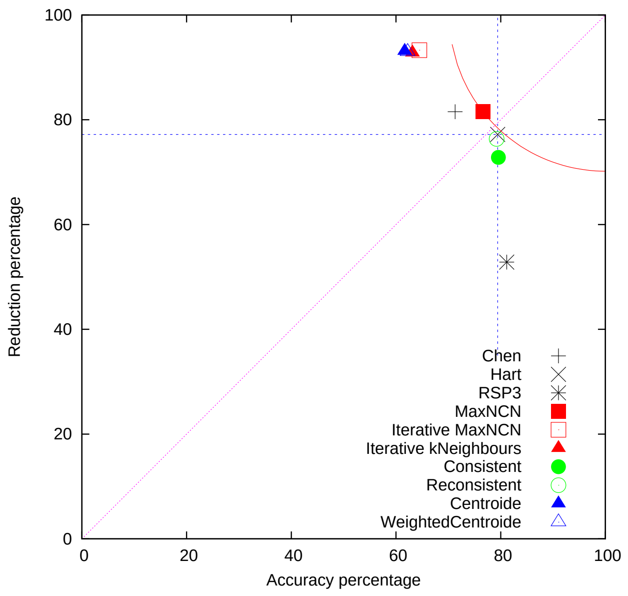


Figure 5.3: Average of accuracy and reduction rates over the eleven databases experimented.

the one obtaining the best results among Chen's, Hart's and RSP3 algorithms.

Thus, in Figure 5.3, which shows the average accuracy and reduction rates, it is possible to see that the *MaxNCN*, Hart's, *Reconsistent*, *Consistent* and Chen's algorithms represent a good trade-off between classification accuracy and size reduction rates. Concerning the remaining algorithms we can comment that *Iterative MaxNCN*, as well as *IterativekNeighbours*, *WeightedCentroide* and *Centroide* obtain similar results: a good reduction rate but not the best accuracy rate. Anyway, comparing the *Iterative* version of *MaxNCN* to the Hart's condensing, the positive percent difference in reduction (21) is bigger than the negative percent difference in accuracy (15). The difference is even more positive to our algorithms if we compare the *Iterative MaxNCN* algorithm to the Chen's approach (a positive percent difference of 12 in reduction, and a negative percent difference of 6 in accuracy).

The RSP3 version, on the contrary, does not obtain such an important reduction, but it gets an improvement in classification accuracy. Anyway, just taking a look at Figure 5.3 one can realise that the improvement in accuracy is not so important as the loss in reduction (a percent difference of 1.6 in accuracy and 20 in reduction) in comparison to the *Consistent* approach.

Finally, it is to be noted that several alternatives to the algorithms introduced here have also been analysed, although some of them had a behaviour similar to that of *MaxNCN*. Other alternatives, as for example *MaxNN*, relying on the NN rule instead of the NCN technique, had a performance systematically worst than *MaxNCN*. In general, algorithms implemented by using the NN as a condensing rule obtained, as expected, very poor results in comparation to the approaches that use the NCN rule. Also, by limiting the neighbourhood size of each representative, we tried to improve the results of *MaxNCN*. In fact, as a result some increase in both accuracy and reduction rates was achieved although differences were not significant.

The nearest representation to the ideal point (100, 100) is the one of the algorithm *MaxNCN*. The next closest one is that of the Hart's approach, and the next one is that of the *Reconsistent* condensing. In addition, the reader can note that these three representations are the nearest to the diagonal, hence they obtain a good balance between classification accuracy and size reduction.

Taking into account each graphical method shown in Figure 5.3, we can summarise that the algorithms reaching the best results are *MaxNCN*, Hart and *Reconsistent*. Among them, *MaxNCN* is proclaimed as the best one, as its average representation is the nearest to the ideal point (100, 100); it is quite near to the diagonal too. In addition while comparing it to the Hart's condensing, the percent difference is more positive in reduction (4.34) than negative in accuracy (2.85).

These are the comments in relation to the average results. For each database, the results for the different algorithms are presented in several figures. The first one (Figure 5.4) shows the classification accuracy and the size reduction for the Cancer database. The results here represented correspond to the average of a five-fold cross-validation estimate of accuracy and percentage of size reduction. This has been done in the same way for every database used in the experiments included in this thesis.

Looking at the Cancer plot (Figure 5.4), the reader can observe a good behaviour, in general, of the condensing algorithms. It is to note that *Centroide* and *WeightedCentroide* lead to identical results. Consequently *WeightedCentroide* and *Centroide* are plotted at the same position in the accuracy-reduction space. So, as such, an open triangle mark for *WeightedCentroide* is covered by the filled triangle mark for *Centroide*. All of the representations are near to the coordinate axes, and relatively near to the ideal point (100, 100). For this database, the best result belongs to the Chen's condensing. Very near to the Chen's result are the *Reconsistent*, *MaxNCN*, Hart, *Consistent* and RSP3 representations. All of them are relatively near to the ideal point, and to the diagonal representing the balance between both accuracy and size reduction.

About Diabetes (Figure 5.5), the first thing one can observe is that in general the results are not as good as for Cancer. In fact, almost no database in the experiments

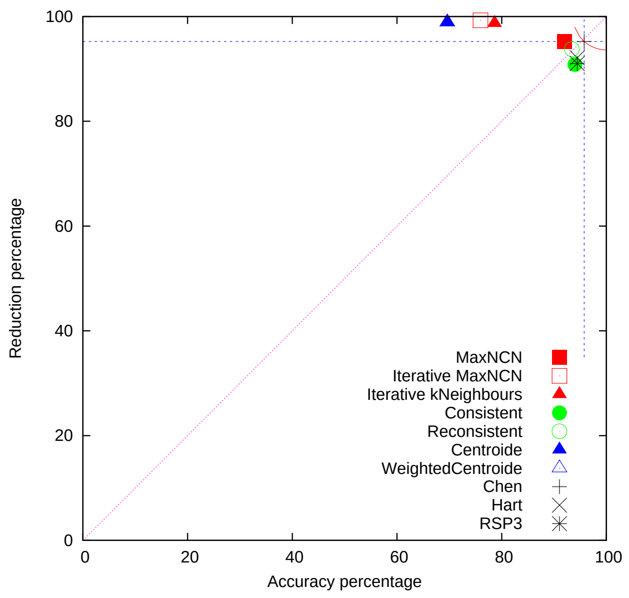


Figure 5.4: Accuracy and reduction rates for the Cancer database.

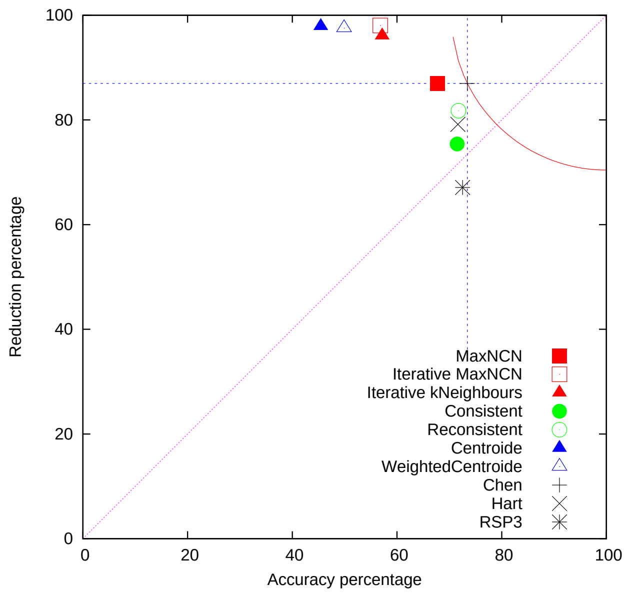


Figure 5.5: Accuracy and reduction rates for the Diabetes database.

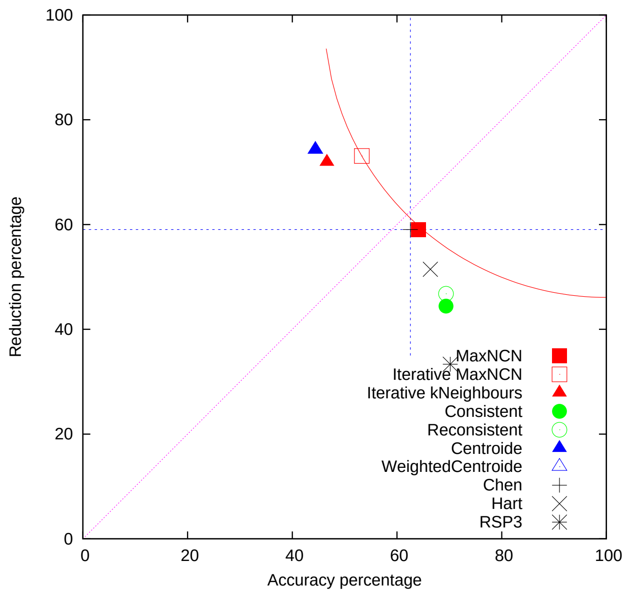


Figure 5.6: Accuracy and reduction rates for the Glass database.

obtains such good results (only Wine and Texture seem to show a similar behaviour). The nearest result to the ideal point (100, 100) is for the Chen's algorithm. In spite of the fact that the algorithms *Iterative kNeighbours*, *Iterative MaxNCN*, *WeightedCentroide* and *Centroide* obtain a very good reduction rate, the rest of the algorithms obtain better balanced results.

Figure 5.6 shows the results for the Glass database.

The algorithm which obtains the nearest result to the ideal (100, 100) is *Iterative MaxNCN*. Very near to its distance to the ideal point is *MaxNCN*, which is much more balanced. So, the best result in the Glass database is obtained by the *MaxNCN* algorithm, being very near to it, the Chen's approximation. The results for *Centroide* and *WeightedCentroide* are equal, so, their representations appear in the same position, as well as in Figure 5.7. It shows results for the Heart database. There, the nearest representation to the point (100, 100) is the Chen's condensing one, which is at the same time, quite well balanced.

The results for the Liver data are presented in Figure 5.8. The nearest result to the ideal point is the *Centroide* representation. Very near to its distance to (100, 100) are the *Iterative MaxNCN*, *MaxNCN* and Chen's representations. *MaxNCN* and Chen's are much more balanced than the other two. It is the reason why,

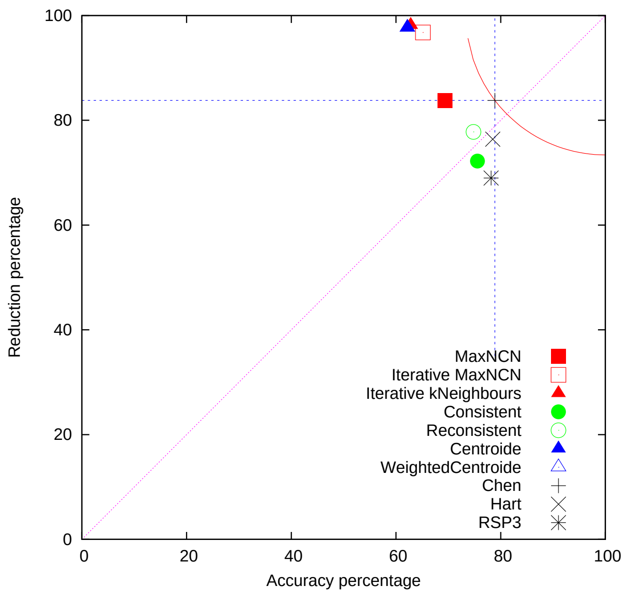


Figure 5.7: Accuracy and reduction rates for the Heart database.

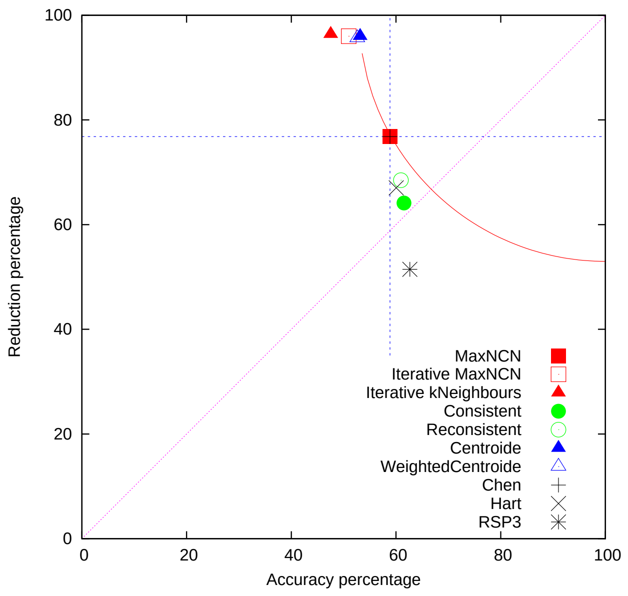


Figure 5.8: Accuracy and reduction rates for the Liver database.

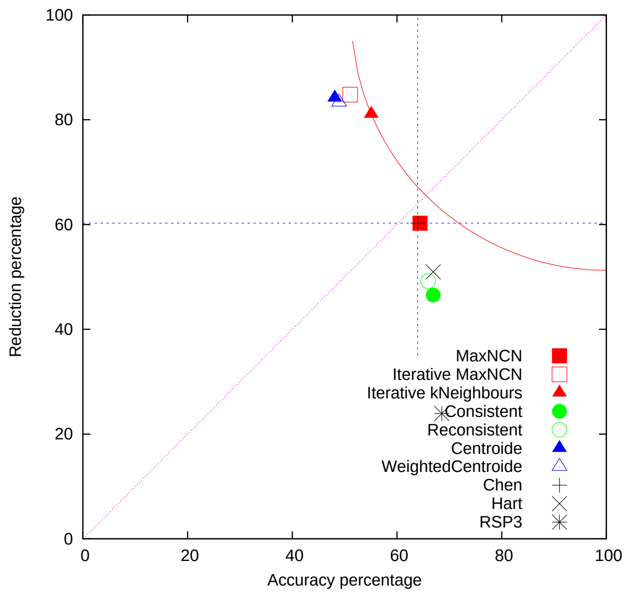


Figure 5.9: Accuracy and reduction rates for the Vehicle database.

they could be determined as the best condensing algorithms for the Liver database. Near to this results are *Reconsistent*, Hart and *Consistent* representations. For the Vehicle database (Figure 5.9) the nearest result to the ideal point is obtained by the *Iterative kNeighbours* algorithm. The most balanced results are reached by the Chen's and *MaxNCN* approaches. So, *MaxNCN* can be concluded as the best condensing algorithm for the Vehicle database (as, coinciding with the Chen's method in reduction, its accuracy is higher). Very good are results for the Chen's and *Iterative kNeighbours* approximations.

The results for the Vowel database are presented in Figure 5.10. There, the *Centroide* and *WeightedCentroide* approximations obtain very similar results thus, they occupy almost the same position in the plot. We can say that the nearest representation to the ideal point belongs to the Hart's algorithm. Very near to this distance are *WeightedCentroide*, *Centroide*, *Iterative MaxNCN*, *MaxNCN* and *Reconsistent* results. Because of their balanced results, the *WeightedCentroide* and *Centroide* approaches can be reported as the best results for the Vowel database. From Figure 5.11, we can say that the *Wine* database has a good behaviour in relation to training set size reduction. The Hart's approach reaches the best result, being very near the *MaxNCN* and the *Reconsistent* representations. The *Centroide*

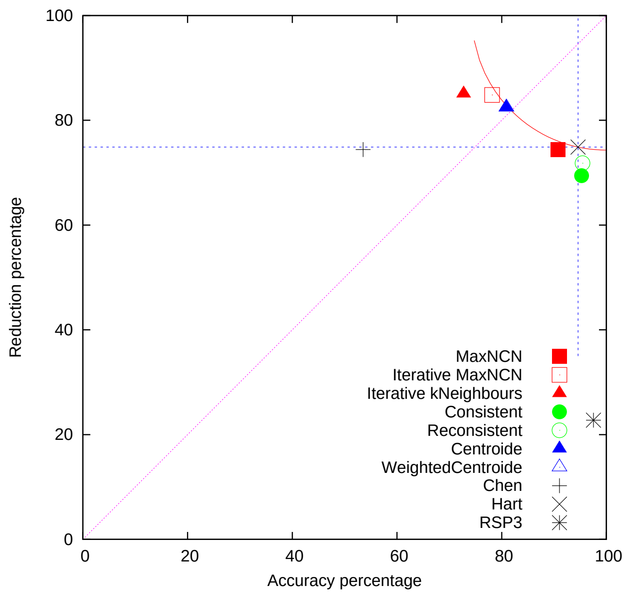


Figure 5.10: Accuracy and reduction rates for the Vowel database.

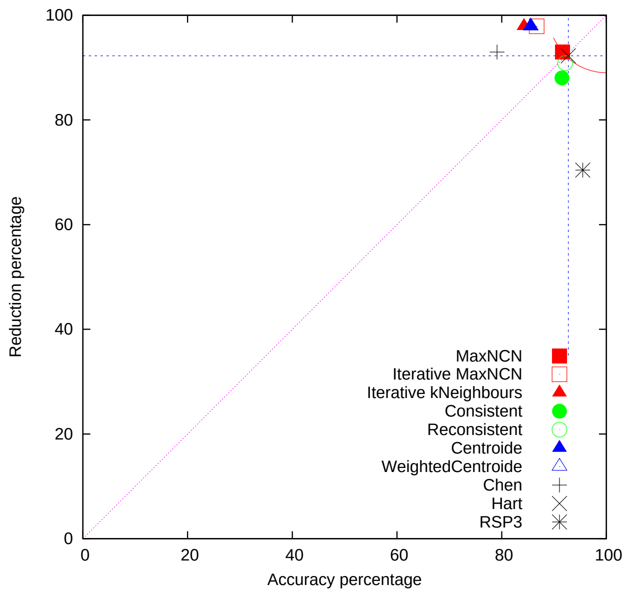


Figure 5.11: Accuracy and reduction rates for the Wine database.

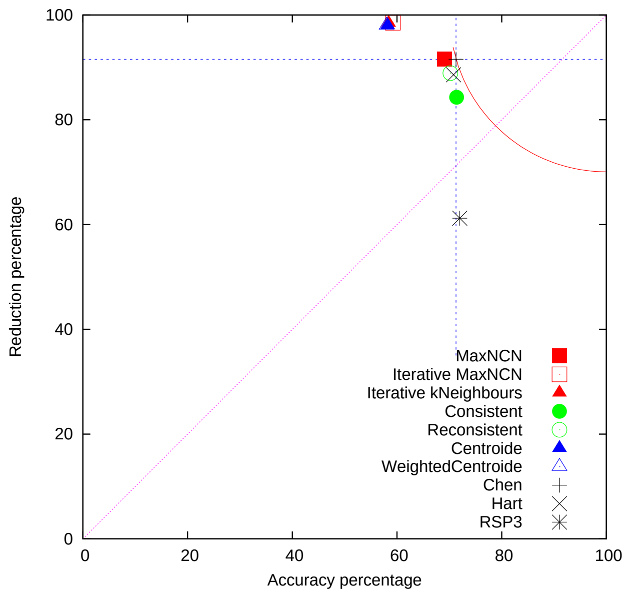


Figure 5.12: Accuracy and reduction rates for the Phoneme database.

and *WeightedCentroide* algorithms obtain very similar results hence, they occupy almost the same position.

Figure 5.12 presents the results for the Phoneme database. Results for the algorithms *Iterative kNeighbours*, *Centroide* and *WeightedCentroide* are very similar thus, they occupy almost the same position in the representation space. The nearest result to the ideal point is reached by the Chen's algorithm. *MaxNCN*, *Reconsistent* and Hart are specially near to Chen's representation. About Satimage (Figure 5.13), the best result is reached by the Hart's algorithm. Specially near to this result is the *MaxNCN* algorithm.

Figure 5.14 presents the results for the Texture database. First of all, it is to note that just at a first sight the Texture database seems to have a good behaviour with almost every condensing algorithm shown in the plot. Secondly, the results for the *Centroide* and *WeightedCentroide* algorithms are very similar, so they occupy almost the same position in the plot. Finally, we conclude that the best algorithm for this database is the Hart's. The *MaxNCN*, *Reconsistent* and *Consistent* approaches are near the best result.

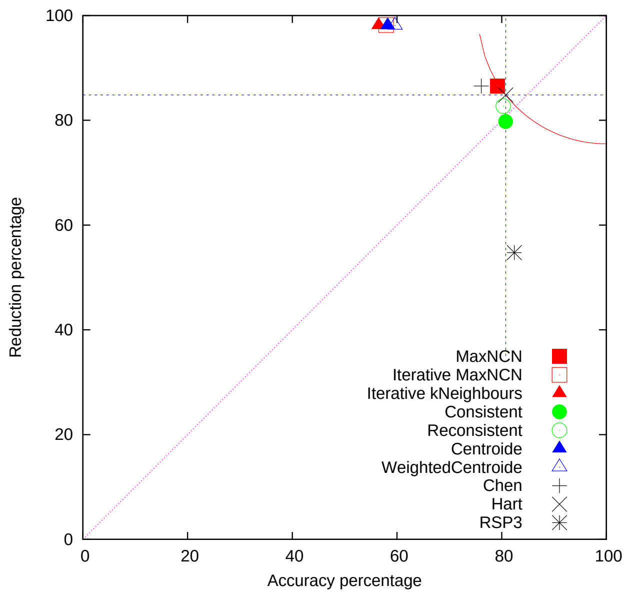


Figure 5.13: Accuracy and reduction rates for the Satimage database.

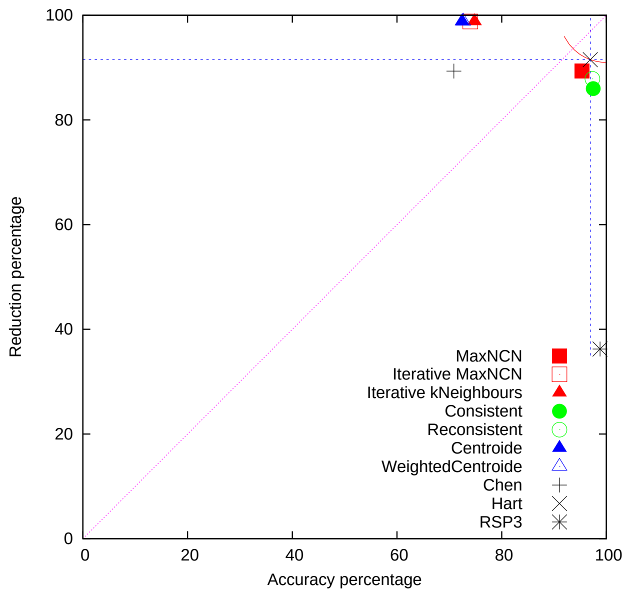


Figure 5.14: Accuracy and reduction rates for the Texture database.

5.4 Conclusions

In this chapter, new approaches to training set size reduction have been compared to some of the already existing ones, including the classical Chen's and Hart's approximations. The new algorithms primarily consist of replacing a group of neighbouring instances that belong to the same class by a single representative. In some cases, this representative is one of the already existing neighbours, and in other cases, it is a new prototype that occupies the best position. This group of representatives is built by the use of the nearest centroid neighbourhood of a given sample. In general, those cover a bigger region than the one defined by the NN.

From the experiments carried out, we can conclude that firstly, the algorithms that obtain a higher set size reduction rate are *Iterative MaxNCN*, *Iterative kNeighbours*, *WeightedCentroide* and *Centroide*. Secondly, the classical Hart's and Chen's approaches obtain good results in general. And at last but not least, the best results obtained for the experiments carried out in this thesis belong to *MaxNCN*, Hart and *Reconsistent*, (in this order) having in mind that they achieve a good balance between set size reduction and classification accuracy, in addition to being the nearest ones to the ideal point (100, 100) in an accuracy=reduction size plot.

Chapter 6

Comparative Analysis with MixtGauss, using Classification Techniques based on Dissimilarity Representations

Contents

6.1 Introduction

The usefulness of condensed sets, optimised by various approaches discussed in the previous chapters, will now be evaluated in a classification task. Traditionally, the 1-NN rule, assigning an unknown object to the class of its nearest neighbour among the Condensed Set (CS) is used for this purpose. This rule is applied as the NN approach is often used for the design of the condensed set. If the classes are represented as compact Gaussian-like clouds of similar spreads, then the 1-NN is expected to generalise well for a small condensed set. Otherwise, a large condensed set might be needed to represent the variability in the data. Alternatively, classifiers in the so-called dissimilarity spaces can be constructed.

6.2 Experimental Setup

We will compare some methods of determining a condensed set R, in combination with three feature-based classification strategies. This is done on the Euclidean distance representations $D(X, R)$ derived in feature spaces. The experiments have been conducted to compare the *MaxNCN* and *Reconsistent* approaches (the best condensing algorithms based on the Nearest Centroid Neighbour (NCN) rule here introduced), to the classical LVQ scheme (the OLVQ1 version), and the new approach based on mixtures of Gaussians, *MixtGauss*. This is done in terms of both set size reduction and classification accuracy for the condensed set. In order to calculate the classification accuracy, the first classification method used is the 1-NN rule based on the set R. The other two methods are the Fisher Linear Discriminant (FLD) and the Quadratic Normal Density Based Classifier (NQC), trained in dissimilarity spaces $D(X, R)$, as explained in Section 2.5.2.

Similarly as *MaxNCN* and *Reconsistent*, the *MixtGauss* algorithm needs to be applied to non overlapping data sets. Therefore, as a general rule, and according to the previously published results [Wilson & Martinez(2000), Sánchez(2004)], the Wilson's editing is considered to properly remove a possible overlap between the classes (see Chapter 5, Section 5.2).

In relation to the LVQ scheme, we should note that at the end of the learning process, the final placement of the prototypes, as well as their distances, are not known. Thus, their optimal cardinality cannot be determined beforehand. To fix the value of k we have to choose first a number of representatives. In order to fix the size of the resulting set, we have used a combination of editing and condensing techniques. Therefore, we use the Wilson's editing (the k parameter here is obtained as explained in Section 5.2), followed by the Hart's condensing algorithm. The resulting set size is near to the ideal. Using this number of representatives (a different number for each database), different values of k were tested and the one leading to the highest accuracy was finally chosen. For each database, different initial set sizes were used in order to compare the results for different cases. For these initial values one can use the first samples of the training set picked up from the respective classes.

6.3 Quantitative Results

Tables 6.1, 6.2 and 6.3 report the classification accuracy of the 1-NN, the FLD and the NQC, respectively, obtained by using different condensed sets and the *Original* set. The latter set is the original TS without any editing or condensing after normalisation. For each condensing method, the average performance values, computed over the eleven data sets are also included. There are no results of the FLD and

Table 6.1: Classification accuracy of the 1-NN (in %). The corresponding reduction rates of the training set size (in %) and the equivalent number of representatives are given in brackets.

	Ori.	MaxNCN	Reconsistent	LVQ	MixtGauss
Canc.	95.2	92.0	93.4	**96.9**	96.5
		[95.2≡26]	[93.8≡34]	[**99.3**≡4]	[**99.3**≡4]
Diab.	70.1	67.7	71.8	**76.6**	75.0
		[87.0≡80]	[81.8≡112]	[**98.1**≡12]	[91.9≡50]
Glass	69.7	64.1	69.4	**72.6**	64.5
		[59.1≡70]	[46.8≡91]	[71.9≡48]	[**75.4**≡42]
Heart	76.3	69.3	74.8	**84.1**	81.9
		[83.8≡35]	[77.8≡48]	[89.8≡22]	[**93.5**≡14]
Liver	62.6	58.9	61.0	**65.6**	63.7
		[76.8≡64]	[68.5≡87]	[93.5≡18]	[**97.1**≡8]
Vehi.	69.4	64.4	66.0	**72.2**	69.6
		[60.3≡269]	[49.2≡344]	[61.6≡260]	[**70.5**≡200]
Vowel	**98.1**	90.8	95.5	80.8	90.7
		[74.4≡108]	[71.8≡119]	[**76.5**≡99]	[**76.5**≡99]
Wine	94.4	91.6	92.1	95.5	**96.0**
		[93.0≡10]	[90.9≡13]	[95.8≡6]	[**97.9**≡3]
Phon.	71.0	69.1	70.2	**77.3**	73.3
		[91.6≡365]	[88.9≡480]	[**93.1**≡300]	[**93.1**≡300]
Sati.	**83.0**	79.2	80.3	82.2	81.3
		[86.5≡693]	[82.7≡890]	[88.3≡600]	[**94.8**≡270]
Text.	**98.9**	95.3	97.3	85.9	97.2
		[89.3≡470]	[87.8≡535]	[**99.5**≡22]	[90.0≡440]
Avr.	80.8	76.6	79.2	**80.9**	**80.9**
		[81.5]	[76.4]	[87.9]	[**89.1**]

the NQC for the *Satimage* and Texture sets (for the original training set), because of their high memory requirements during training [Duin et al.(2004)]. The average performances excluding these data sets are also presented. Since the adaptive techniques (the *MixtGauss* and the LVQ) allow one to determine a condensed set of any size, only a few results are shown in the tables. These correspond to the best accuracy values, found in the sets between one representative per class and the size

Table 6.2: Classification accuracy of the FLD in dissimilarity spaces and the corresponding reduction rates of the training set size in brackets. Both values are expressed in %. Averages over all data sets (1-11) and all data sets but Satimage and Texture (1-9), are also given.

	Original	MaxNCN	Reconsistent	LVQ	MixtGauss
1 Cancer	85.50	96.48	96.48	**97.22**	96.92
		[95.2]	[93.8]	[**98.2**]	[96.3]
2 Diabetes	74.23	76.05	**77.09**	77.08	76.70
		[87.0]	[81.8]	[**98.7**]	[87.0]
3 Glass	53.88	66.53	67.49	**77.17**	69.77
		[59.1]	[46.8]	[61.4]	[**93.0**]
4 Heart	60.75	80.72	78.49	**84.79**	82.60
		[83.8]	[77.8]	[98.1]	[**98.2**]
5 Liver	68.35	68.96	67.83	**70.70**	68.67
		[76.8]	[68.5]	[92.8]	[**93.5**]
6 Vehicle	77.19	79.31	79.79	**80.96**	80.02
		[60.3]	[49.2]	[67.5]	[**76.4**]
7 Vowel	41.03	95.76	**96.67**	91.42	94.28
		[74.4]	[71.8]	[**76.5**]	[**76.5**]
8 Wine	33.15	98.28	98.30	98.87	**98.90**
		[93.0]	[90.9]	[**95.8**]	[**95.8**]
9 Phoneme	70.62	69.75	70.52	**77.11**	73.94
		[91.6]	[88.9]	[98.2]	[**99.8**]
10 Satimage	****	24.49	23.90	82.11	**82.20**
		[86.5]	[82.7]	[95.3]	[**95.9**]
11 Texture	****	37.33	27.56	99.00	**99.13**
		[89.3]	[87.8]	[92.5]	[**93.8**]
Avr. (1-11)	****	72.2	71.3	**85.1**	83.9
		[81.5]	[76.4]	[88.6]	[**91.5**]
Avr. (1-9)	62.7	81.3	81.4	83.9	82.4

determined by the *MaxNCN* algorithm (as the *MaxNCN* condensed set is always smaller than the *Reconsistent* condensed set). The best rates for each database and for the average of each table are highlighted in bold.

The first observation is that, in general, the LVQ and *MixtGauss* condensing

Table 6.3: Classification accuracy of the NQC in dissimilarity spaces and the corresponding reduction rates of the training set size in brackets. Both values are expressed in %. Averages over all data sets (1-11) and all data sets but Satimage and Texture (1-9) are also given.

	Original	MaxNCN	Reconsistent	LVQ	MixtGauss
1 Cancer	80.23	96.19	96.48	**97.22**	96.34
		[95.2]	[93.8]	[**96.3**]	[**96.3**]
2 Diabetes	65.62	75.39	74.74	**77.09**	76.70
		[87.0]	[81.8]	[97.1]	[**98.7**]
3 Glass	40.15	53.81	54.40	**65.10**	64.11
		[59.1]	[46.8]	[**89.5**]	[82.5]
4 Heart	55.56	**82.97**	80.00	82.95	82.21
		[83.8]	[77.8]	[**97.2**]	[88.9]
5 Liver	46.67	62.63	60.00	**67.79**	65.20
		[76.8]	[68.5]	[85.5]	[**89.1**]
6 Vehicle	27.78	60.27	54.37	**74.81**	74.58
		[60.3]	[49.2]	[76.4]	[**91.1**]
7 Vowel	10.20	31.96	29.41	**94.59**	94.10
		[74.4]	[71.8]	[**92.2**]	[**92.2**]
8 Wine	33.15	97.73	96.05	97.76	**99.43**
		[93.0]	[90.9]	[**97.9**]	[95.8]
9 Phoneme	73.26	73.08	73.22	**76.11**	73.33
		[91.6]	[88.9]	[**95.4**]	[**95.4**]
10 Satimage	****	79.40	79.17	**81.93**	81.62
		[86.5]	[82.7]	[**98.3**]	[97.1]
11 Texture	****	96.31	96.51	**97.58**	97.53
		[89.3]	[87.8]	[95.0]	[**96.3**]
Avr. (1-11)	****	73.6	72.2	**83.0**	82.3
		[81.5]	[76.4]	[92.8]	[**93.0**]
Avr. (1-9)	48.1	70.5	68.7	81.5	80.7

approaches increase the average accuracy in comparison to the original training set accuracy. The increase is larger by using the FLD and the NQC than by using the 1-NN rule (where the increase is quite small). Anyway, it is significant that in all these cases, the performance is increased despite the high reduction of the training set (see Tables 6.1–6.3). Remember that although the FLD and the NQC are based on the distances to the representatives, they still make use of the entire training set during the training step. This is the reason why the classification accuracy increases so much when using any condensed set, in comparison to the results obtained by the use the original training set.

By using the FLD, the classification accuracy obtained for the *MaxNCN* and *Reconsistent* procedures is very similar. The *MixtGauss* approach leads to a higher accuracy than both of them. The best classification performance of the FLD relies on the *LVQ*. A similar pattern can be observed for the NQC. This differs, however, for the 1-NN rule, especially in the case of the *Reconsistent* algorithm. The smallest classification accuracy is reached for the *MaxNCN*. The *Reconsistent* approach gives better results, and the best values are obtained for the LVQ and the *MixtGauss*.

The training reduction rates, being the percentage by which the original training size is reduced, are provided in square brackets in Tables 6.1–6.3. Since the *MaxNCN* and *Reconsistent* techniques determine the number of representatives directly, their reduction rates are identical for the three classification methods used. Additionally, for each condensing method, the average reduction values computed over the eleven data sets are also included.

In relation to the training reduction rates, the average is higher for the *MaxNCN* than for the *Reconsistent* method. This is to be expected as in the second algorithm, some instances are added to the *MaxNCN* condensed set to improve the classification accuracy. It can also be observed that the average training set reduction rates are higher for the adaptive LVQ and *MixtGauss* techniques than for the selective *MaxNCN* and *Reconsistent* procedures. The highest reduction rate is the one of *MixtGauss*. This holds for the three prototype-based classification strategies used here.

It is to note that in Table 6.1 the *MixtGauss* algorithm reaches the highest classification accuracy and the highest size reduction at the same time. In relation to the classification methods considered, in general, we can observe that the FLD and the NQC perform better than the 1-NN rule, when they rely on the condensed sets determined by the adaptive schemes. Comparing both algorithms, the one with the best performance corresponds to the FLD. On the other hand, the *MaxNCN* and the *Reconsistent* procedures have their best classification accuracy with the 1-NN rule. As designed, *Reconsistent* leads to a higher NN performance than the *MaxNCN*.

These results are graphically shown in Figure 6.1. There we represent the average classification accuracy rate and the average size reduction percentage for each algorithm to compare.

The four algorithms to compare have been represented by four different type

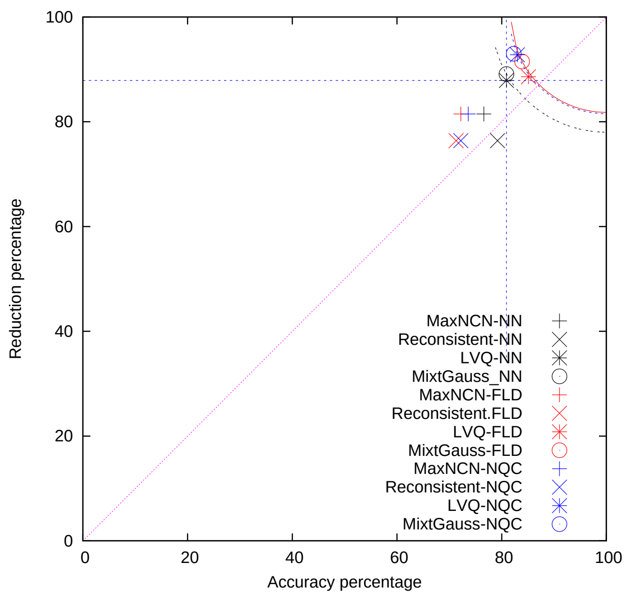
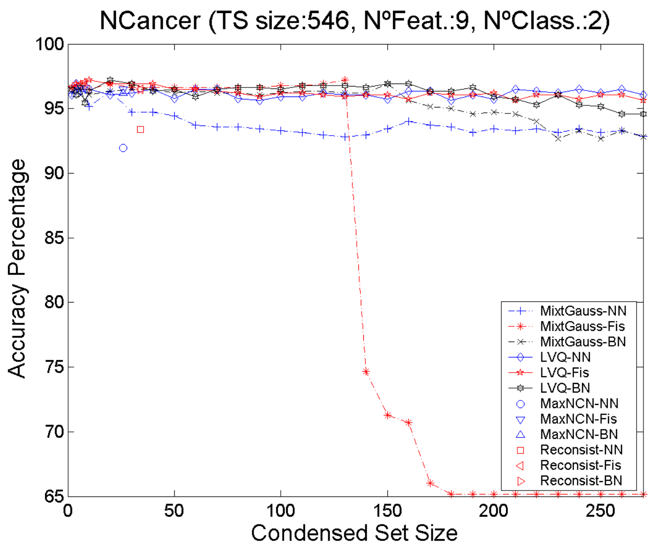


Figure 6.1: Average of accuracy and reduction rates over the eleven databases experimented.

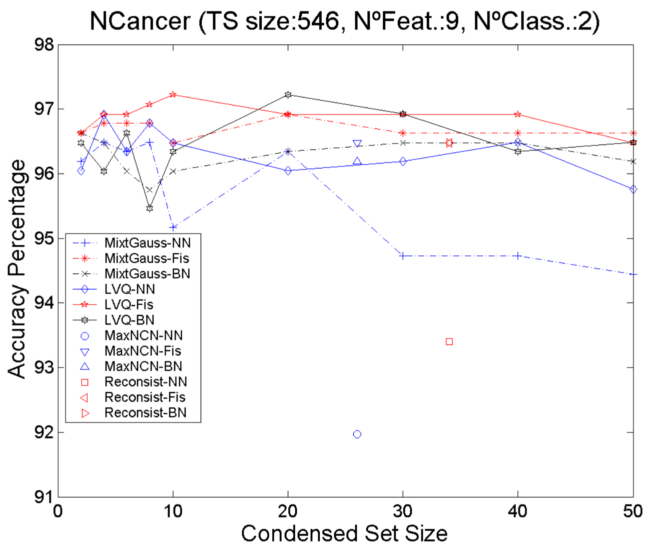
of markers, meanwhile the three classifiers have been represented by three different colours. The NN rule is represented in black. The FLD is represented in red. And finally, the NQC is represented in blue. In these three colours, three semi-circles have also been drawn. They indicate the distance from the ideal point (100, 100), to the nearest average representation using the different classifiers. The nearest one among all of them is the *MixtGauss* algorithm, using the FLD as a classifier. Among the NQC representations, the nearest to the ideal point (100, 100) is the LVQ, being very near the *MixtGauss* approach. Finally, among the NN representations the nearest one to (100, 100) is the *MixtGauss* approach, being the LVQ technique very near, with identical accuracy (and slightly lower size reduction rate). In relation to the *MaxNCN* and the *Reconsistent* algorithms, it is clearly shown in Figure 6.1, that they always get the same reduction, as only the classification method is changed.

The representation for the LVQ technique with the classical NN classifier have been chosen as a reference. So, it can be seen how the rest of combinations affecting the adaptive algorithms, will improve these reference.

These results can also be analysed by studying the plots in Figures 6.2-6.12. There, the behaviour of the algorithms investigated here is shown for the eleven



(a)



(b)

Figure 6.2: Trade-off between the resulting condensed set sizes and classification accuracy of the first NN rule for the Cancer database, starting from one instance per class until half the training size (a), and a zoom at the first cases (b).

data sets used in the experiments, respectively. Each figure is composed of two plots. Both of them represent (in the axes) the size of the condensed set, and the classification accuracy obtained. The *MaxNCN* and the *Reconsistent* algorithms only have one representative per database, as they automatically choose the condensed set size. For the case of the LVQ and the *MixtGauss* algorithms, a line is drawn as several cases are represented: from one prototype per class, until reaching half the size of the original training set, in general (Figures 6.2-6.12 (a)). In the second plot of each figure (Figures 6.2-6.12 (b)), only the results for the smallest condensed sets for each database are plotted, in order to distinguish the plots and to recognise which algorithms have a better behaviour at these first steps.

At first stages of the Cancer data set (Figure 6.2) the best results are obtained by the LVQ condensing, using the linear classifier. This combination gets good results in general for this data set, as well as *MixtGauss* do using the linear classifier, when a reduction higher than 75 per cent is achieved. A minor reduction drastically decreases the results for the last combination.

Figure 6.3 refers to the Diabetes data set. It looks as the combination *MixtGauss*-NQC has the best behaviour at first steps. Anyway, as the condensed set size increases, the classification accuracy decreases, specially for the combinations *MixtGauss*-FLD, *MixtGauss*-NQC and LVQ-NQC.

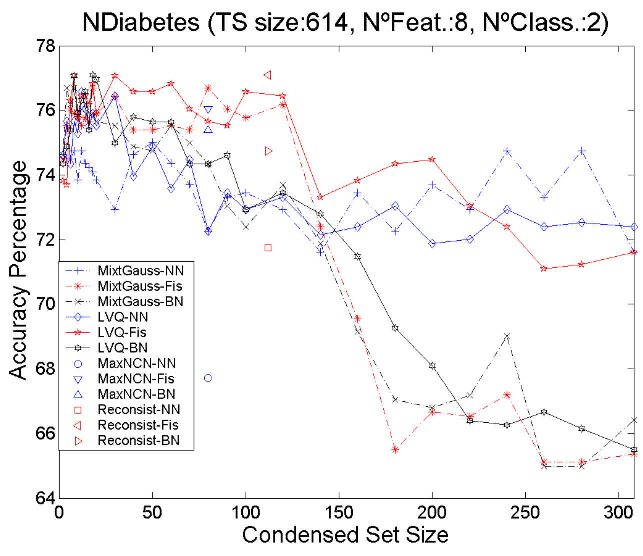
Figure 6.4 do not have a legend, as it is equal to these from other scatter diagrams, and it will disrupt the analysis of the Glass plots. Until reaching the size of 24 items in the condensed set, the best results are obtained by the combination *MixtGauss*-FLD. Quite close to it are the results for the combination LVQ-FLD.

Figure 6.5 represents the behaviour of the studied algorithms with the Heart database. The most important points to note here are, firstly, that the combination *MaxNCN*-NQC obtains better accuracy for the size that it automatically gets. And secondly, that LVQ-NQC decreases drastically from the condensed set size of 40 representatives and on.

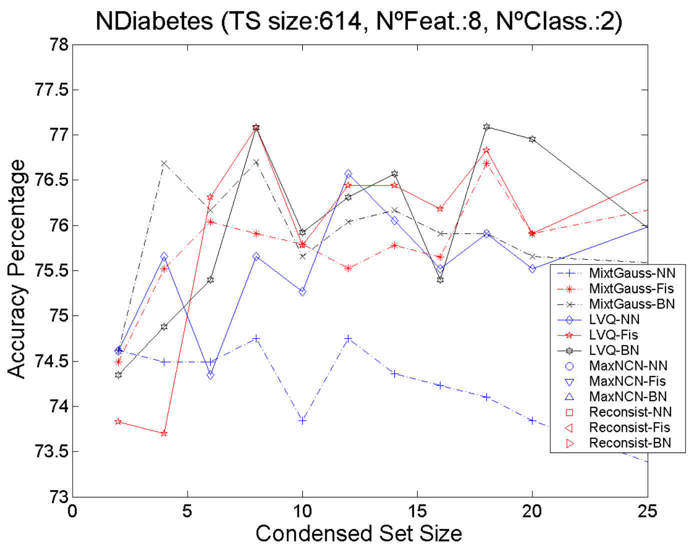
In Figure 6.6 results for the Liver data set is represented. The *MaxNCN* algorithm continue having a good behaviour, anyway it could be said that in general better results are obtained by LVQ-FLD and *MixtGauss*-FLD.

Figure 6.7, refers to the Vehicle data set. It can be observed there that the best classification accuracy is reached by the *MixtGauss*-NQC for the smallest condensed sets. For sets with more than three representatives per class, the best accuracy is reached by the LVQ-FLD. It is to note that the performance of the *MixtGauss*-FLD is always close to the best case.

In Figure 6.8 the results for the Vowel data set are shown. The three smallest condensed sets represented (1, 2 and 3 representatives per class) lead to the best performance for the combination of the LVQ-NQC and the *MixtGauss*-NQC algo-

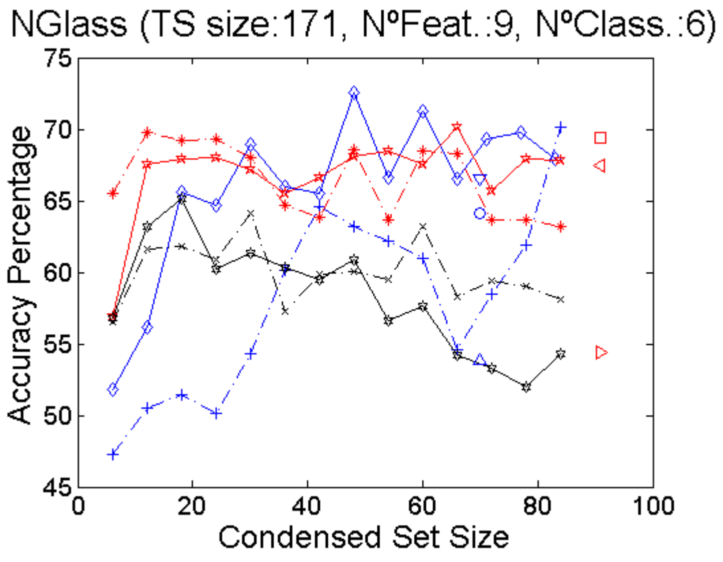


(a)

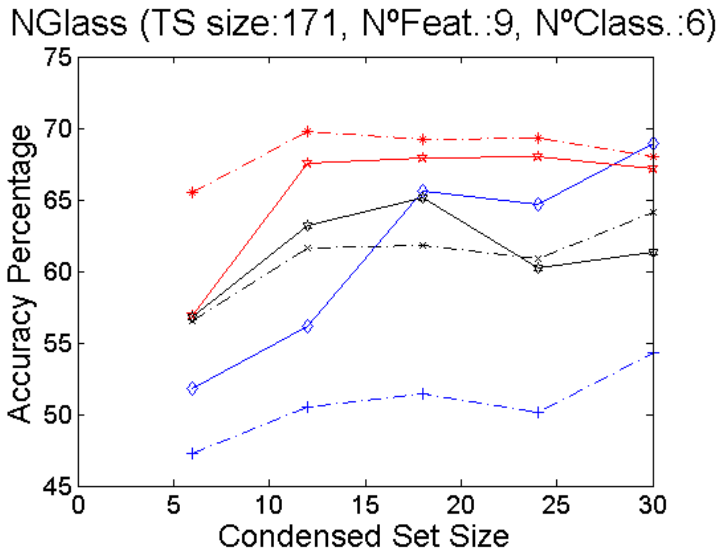


(b)

Figure 6.3: Trade-off between the resulting condensed set sizes and classification accuracy of the first NN rule for the Diabetes database, starting from one instance per class until half the training size (a), and a zoom at the first cases (b).

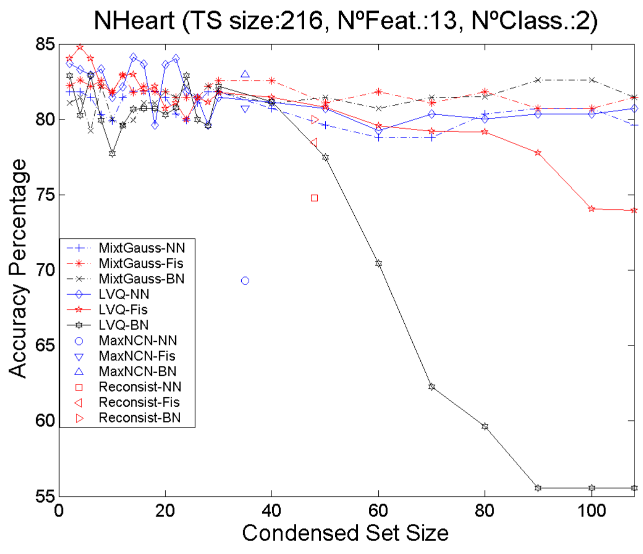


(a)

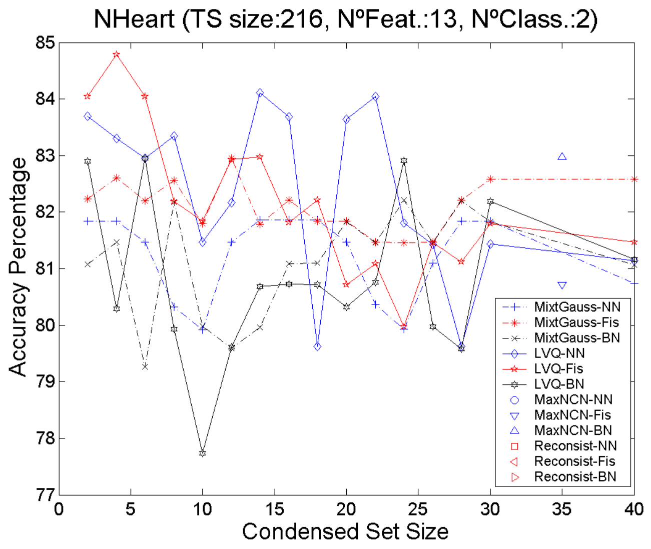


(b)

Figure 6.4: Trade-off between the resulting condensed set sizes and classification accuracy of the first NN rule for the Glass database, starting from one instance per class until half the training size (a), and a zoom at the first cases (b).

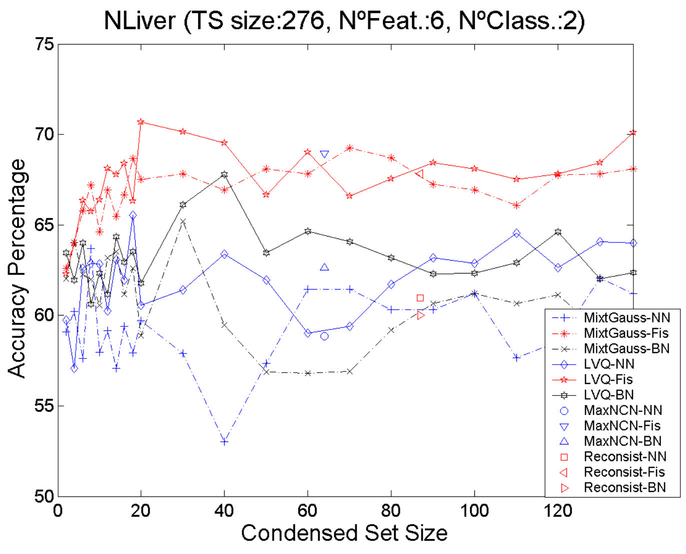


(a)

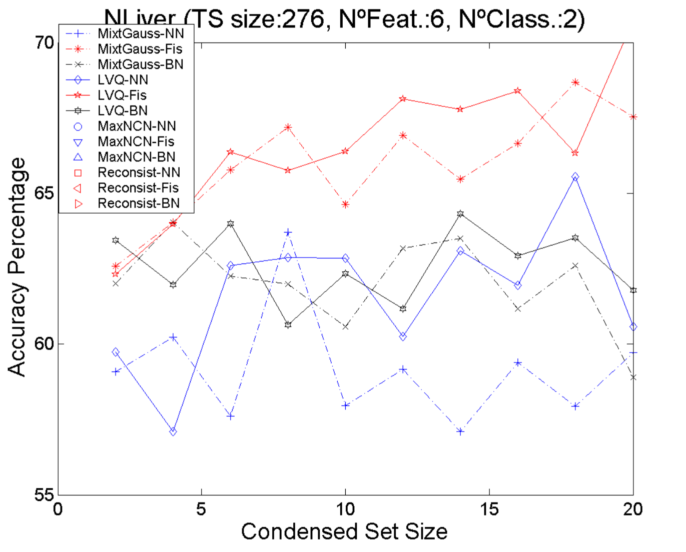


(b)

Figure 6.5: Trade-off between the resulting condensed set sizes and classification accuracy of the first NN rule for the Heart database, starting from one instance per class until half the training size (a), and a zoom at the first cases (b).

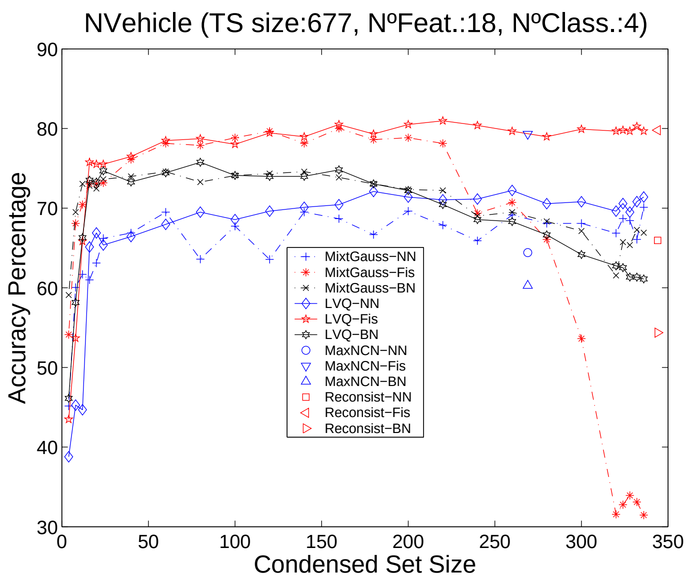


(a)

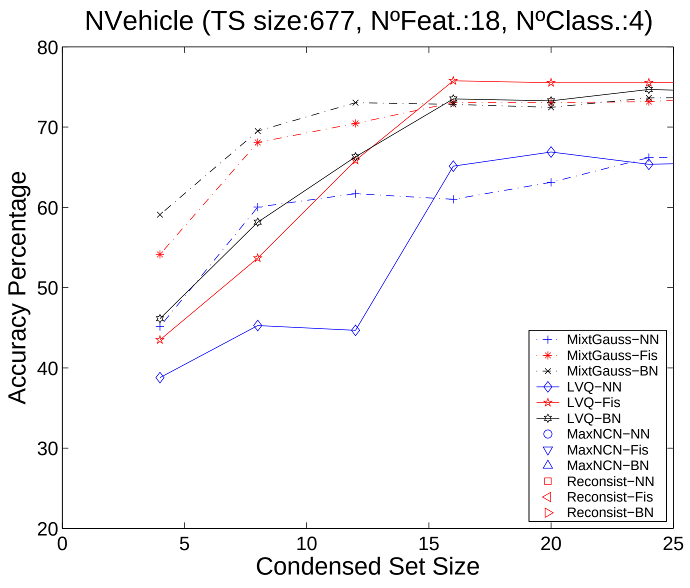


(b)

Figure 6.6: Trade-off between the resulting condensed set sizes and classification accuracy of the first NN rule for the Liver database, starting from one instance per class until half the training size (a), and a zoom at the first cases (b).

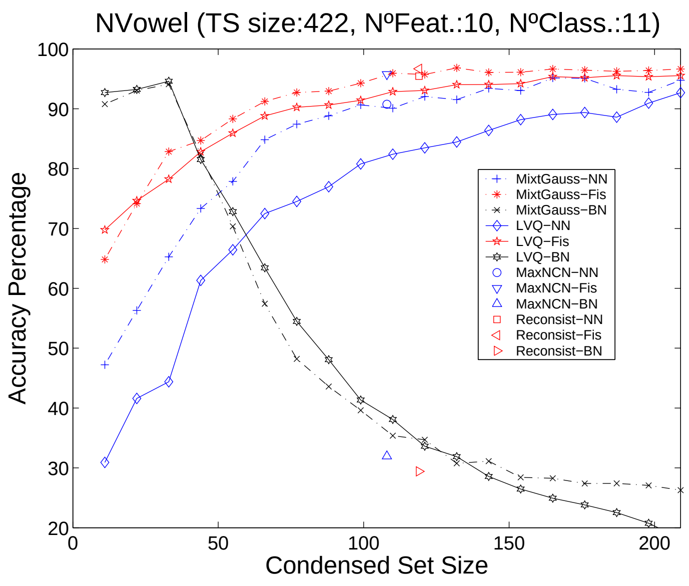


(a)

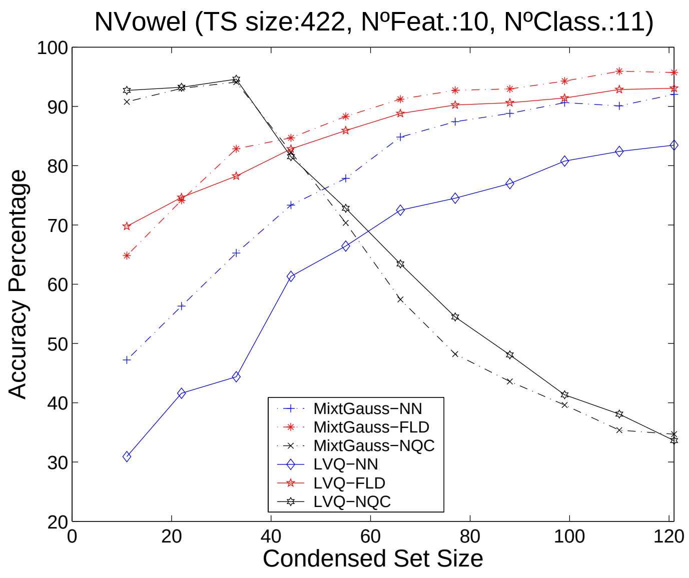


(b)

Figure 6.7: Trade-off between the resulting condensed set sizes and classification accuracy of the first NN rule for the Vehicle database, starting from one instance per class until half the training size (a), and a zoom at the first cases (b).



(a)



(b)

Figure 6.8: Trade-off between the resulting condensed set sizes and classification accuracy of the first NN rule for the Vowel database, starting from one instance per class until half the training size (a), and a zoom at the first cases (b).

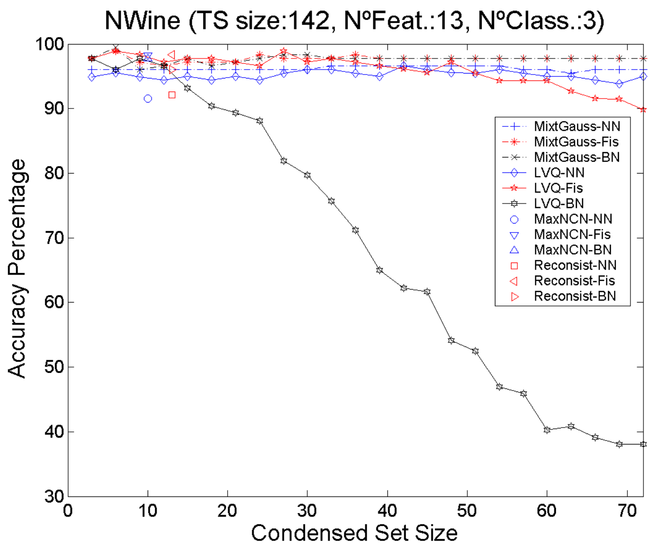
rithms. However, for larger sizes, the classification accuracy suddenly decreases. It is due to a large number of classes in relation to the number of representatives, which effectively translates into the dimension of the dissimilarity space. As a result, the number of instances per class is too small, and as the size of the condensed set increases, class covariance matrices are badly estimated. Hence, the classification accuracy is diminished. The next best results belong to the *MixtGauss*-FLD, the LVQ-FLD and the *MixtGauss*-NN, in that order.

In general, a good behaviour is observed in the Wine database (Figure 6.9) with almost every algorithm. Only LVQ-NQC reports a decrease from 12 representatives on. The *MaxNCN*-FLD and the *Reconsistent*-FLD combinations obtain better results than others for their automatic calculated condensed set size. LVQ-FLD, *MixtGauss*-FLD and *MixtGauss*-NQC, as well as *MaxNCN*-FLD and *Reconsistent*-FLD, can be reported as obtaining the best results for the Wine database.

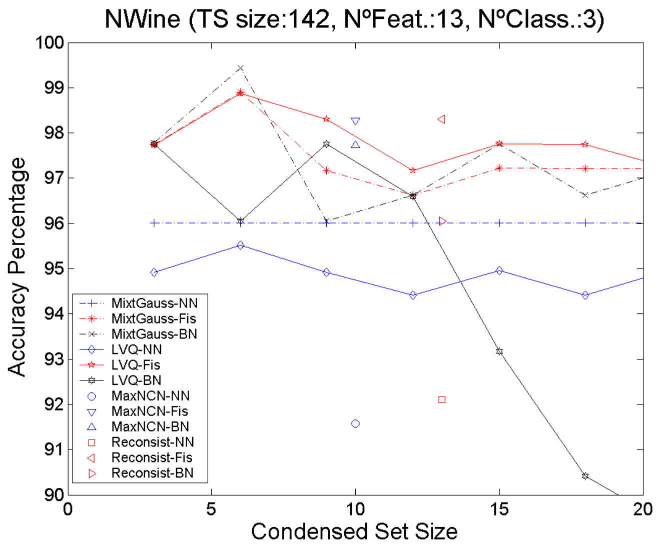
Figure 6.10, represents the results for the *Phoneme* database. It shows as the best results, in general, those for the combinations of LVQ. In Figure 6.11, the results for the Satimage data set are shown. The most reduced sets lead to a better performance by using the *MixtGauss*-NQC, the LVQ-NQC, the *MixtGauss*-FLD and the LVQ-FLD methods.

The results for the data set Texture are shown in Figure 6.12. The best ones are obtained by the following combinations: the LVQ-NQC, the *MixtGauss*-NQC, the LVQ-FLD and the *MixtGauss*-FLD. Meanwhile the later shows the best behaviour between the sizes of 44 and 300. The larger plots for the Satimage (Figure 6.11 (a)), as well as for the Texture (Figure 6.12(a)) data sets show that the classification accuracy also decreases around the size of 300 representatives for the *MixtGauss*-FLD and the LVQ-FLD. It is due to similar reasons to those mentioned for the Vowel data set (Figure 6.8).

In general, the best trade-off results (between the accuracy and the condensed set size) are the ones for the FLD. The results for the NQC are similar, and the ones for the 1-NN rule are significantly worse, in the most of the databases. In relation to the condensing algorithms, the LVQ and the *MixtGauss* lead to very similar results. In general, the points corresponding to the *MaxNCN*-FLD and the *Reconsistent*-FLD results are near the functions representing the *MixtGauss*-FLD and the LVQ-FLD results, similarly as the points representing the *MaxNCN*-NQC and the *Reconsistent*-NQC results are close to the points describing the *MixtGauss*-NQC and the LVQ-NQC obtained data. This observation does not hold for the *MaxNCN*-NN and the *Reconsistent*-NN, neither the *MixtGauss*-NN and the LVQ-NN, as there exists a big difference in favour of the *MixtGauss* and the LVQ for some of the data sets in the experiments.

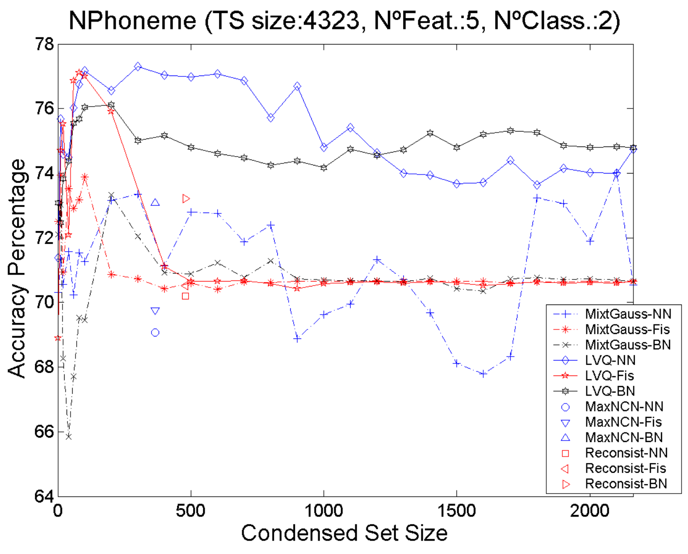


(a)

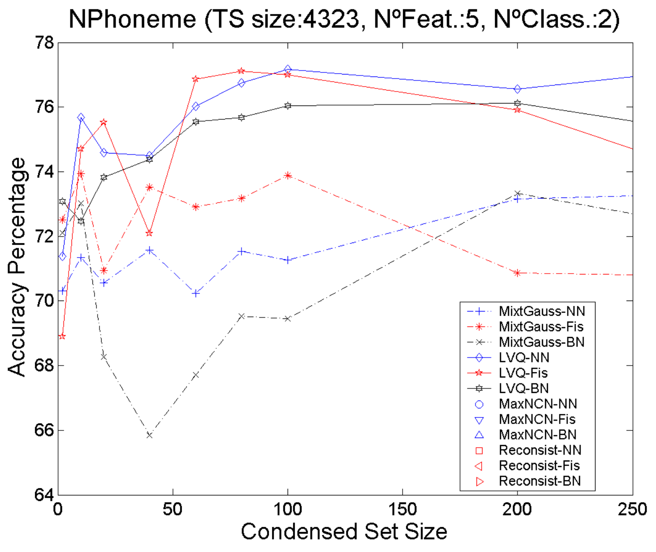


(b)

Figure 6.9: Trade-off between the resulting condensed set sizes and classification accuracy of the first NN rule for the Wine database, starting from one instance per class until half the training size (a), and a zoom at the first cases (b).

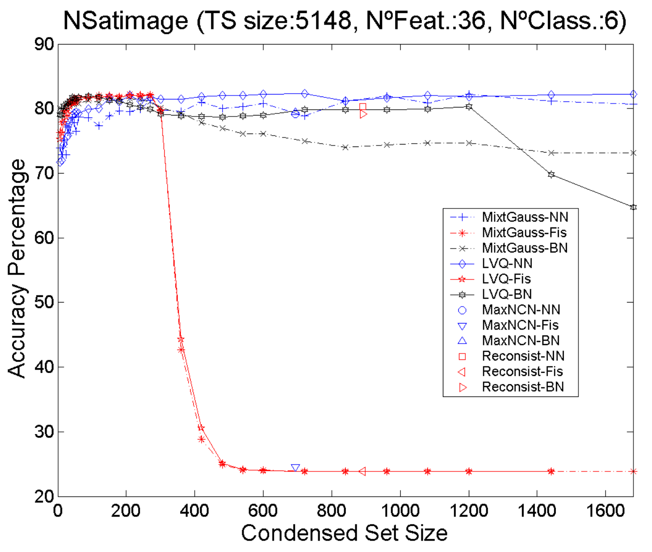


(a)

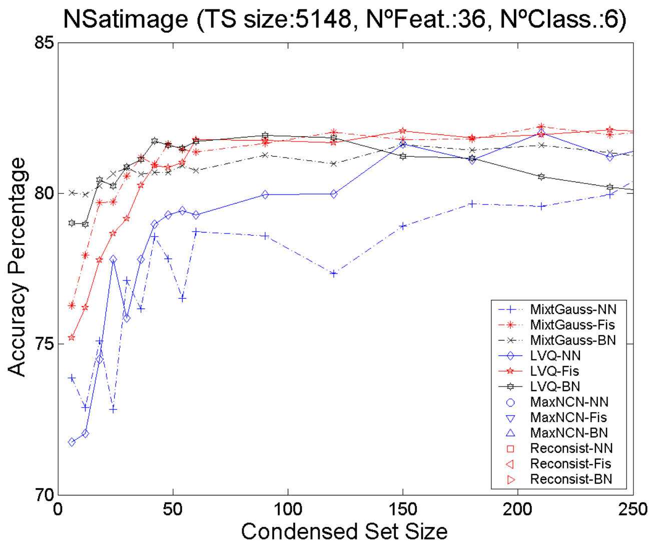


(b)

Figure 6.10: Trade-off between the resulting condensed set sizes and classification accuracy of the first NN rule for the Phoneme database, starting from one instance per class until half the training size (a), and a zoom at the first cases (b).

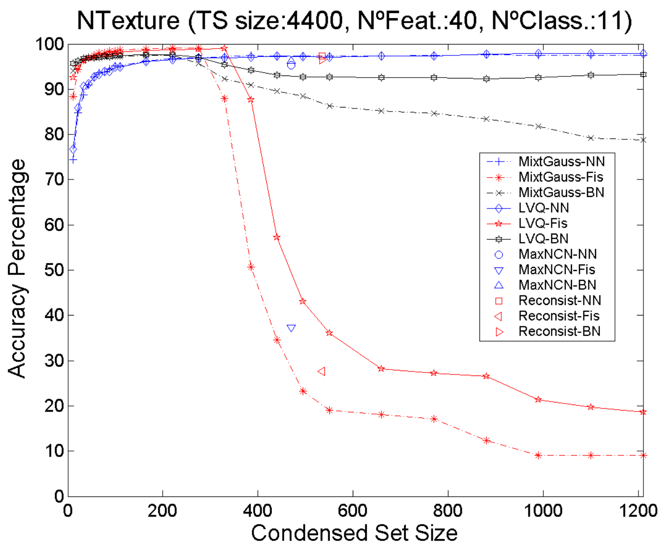


(a)

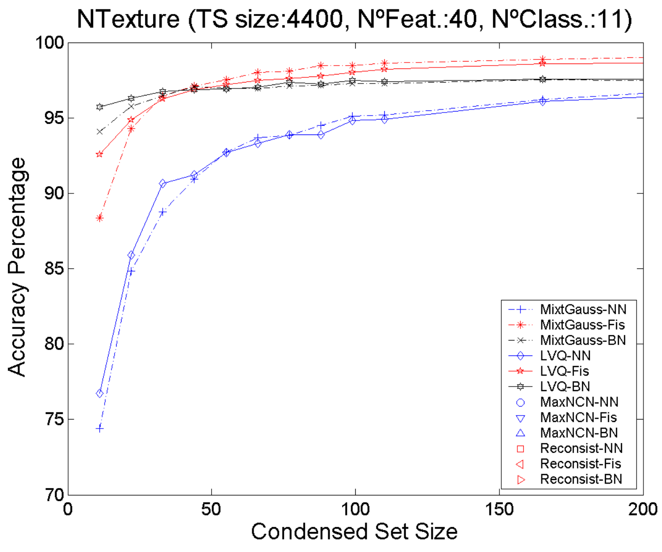


(b)

Figure 6.11: Trade-off between the resulting condensed set sizes and classification accuracy of the first NN rule for the Satimage database, starting from one instance per class until half the training size (a), and a zoom at the first cases (b).



(a)



(b)

Figure 6.12: Trade-off between the resulting condensed set sizes and classification accuracy of the first NN rule for the Texture database, starting from one instance per class until half the training size (a), and a zoom at the first cases (b).

6.4 Conclusions

In this chapter we compare several prototype reduction techniques, including both selecting and generating schemes, in connection with performance of classifiers defined on the corresponding sets of representatives. We use the following classifiers: the 1-NN rule, and linear and quadratic prototype-based classifiers defined in Euclidean-distance dissimilarity spaces. The purpose of this experimental study is to discuss a number of prototype optimisation methods with respect to their ability of maintaining small representative sets and high classification performance.

From our investigations it can be concluded that there is no significant difference between the LVQ and the *MixtGauss* prototype generation techniques in general. However, when comparing them to the *MaxNCN* and *Reconsistent* methods, the first two algorithms tend to give both larger reduction of the training set (i.e. smaller condensed sets) and lead to higher classification accuracy. This means that LVQ and *MixtGauss* are good as condensing algorithms since they yield a good performance and allow one to control the number of prototypes. Additionally, they represent a good trade-off between the classification accuracy and the reduction rates.

In addition, we have observed a general tendency in relation to which data sets obtain better results with one or other approach. Firstly, while LVQ seems to be the best option for data sets of two classes, *MixtGauss* tends to get better results for data sets with more than two classes. Secondly, while LVQ in general for most cases give a better solution for the data sets with a smaller dimensionality, *MixtGauss* offers a better possibility for those with a bigger dimensionality. In fact, the rate between dimensionality and number of classes results in a separation of data sets obtaining better results by the use of LVQ (around four and smaller) or *MixtGauss* (bigger than four); although there are exceptions in all the cases here commented.

In relation to the prototype-based decision rules, in general, the classification accuracy obtained by the FLD tends to be higher than that of the NQC, and both higher than the accuracy of the 1-NN rule, independently of the considered condensing algorithms. This is striking as the condensed sets are often optimised to guarantee that the 1-NN rule performs well. Our explanation is that the FLD and the NQC, although based on the condensed sets, still make use of the entire training set for determining the decision boundary. The 1-NN rule, when applied as a nearest representative rule, discards other training instances.

What is important, is the fact that for the evaluation of new objects the computational complexity of the FLD is $O(Jr)$. This is similar to the computational complexity of the 1-NN rule, $O(r)$, provided that J, the number of classes, is small. For a large number of classes, the computational cost of the FLD increases. The FLD may, therefore, be an alternative to the 1-NN rule for the prototype-based classification in normalised feature vector spaces. The NQC is much more computationally heavy, so it can only be advantageous for small condensed sets and a small number of classes. In fact, the NQC was used here, to show that a more complex classifier than a linear one is not necessarily yielding a better performance in dissimilarity spaces resulting from normalised vectorial data. The effect of feature normalisation plays a role as otherwise the NQC might have been preferred to the FLD. In general, the good performance of the FLD in dissimilarity spaces can be explained by the use of Euclidean distance relying on *standardised* features (hence approximate normal distributions for the resulting distances) and non-complicated class structures for which this distance measure is discriminative. The later means that the classes do not differ much in the range of their average within-class distances.

Our final conclusion is that the adaptive techniques, the LVQ and the *MixtGauss*, combined with the FLD, offer the best trade-off between the reduction rate, computational cost and the classification performance in normalised vector spaces for the considered problems.

In perspective, our study opens a new possibility for using linear (or more complex) classifiers in dissimilarity spaces derived from normalised feature vector spaces instead of the 1-NN rule, both defined by the same optimised condensed sets. Since a linear function in such a dissimilarity space is a non-linear function in the original vector space, the distance measure or its nonlinear (sigmoidal, logarithmic) transformation can be a way to incorporate the nonlinearity aspects of the data. To our knowledge, these aspects have not been considered among the researchers studying the condensing techniques, yet. Although the FLD was chosen here, other linear functions can be studied, such as a logistic classifier or a hyperplane defined by a linear, sparse or non-sparse, programming procedure. Additionally, a linear (or polynomial) support vector machine (SVM) [Cristianini & Shawe-Taylor(2000)] can be trained in dissimilarity spaces. Note, however, that in such a case the SVM will rely on support vectors which are objects found in a complete dissimilarity space, hence they rely on the distances to *all* original instances. So, the SVM will not work as a prototype selection. If however good representatives are found by other techniques, the linear SVM will determine the optimal large-margin hyperplane in the corresponding dissimilarity space. Future research may focus on comparing a (non-)linear SVM built in the original feature vector space and the SVM in a dissimilarity space defined by a small set of optimised representatives.

Chapter 7

Comparative Analysis of Condensing Techniques with Nearest Neighbour Efficient Search Techniques

Contents

7.1 Introduction

The NN rule is used in many tasks due to its simplicity and efficiency. The most simple algorithm for implementing this rule is the *exhaustive search*: it calculates each distance from the sample to classify, to the instances in the training set (TS). In order to reduce the time needed by this process, several algorithms use prototype selection techniques in order to reduce the quantity of instances of each class in the training set. Nevertheless, it exists the possibility that those methods provoke important classification errors, due to overlapping among classes, or incorrectly eliminated instances. That is the reason why the *efficient algorithms* are used: they

Table 7.1: Classification accuracy rate of nearest neighbour efficient search techniques. Averages over all data sets are also given.

	k-d tree	kvp-tree	kFukNar	LAESA	KAESA	k-GNAT
Cancer	95.17	95.32	95.17	95.17	95.17	95.17
Diabetes	70.06	69.93	70.06	70.06	70.06	70.06
Glass	69.66	70.16	69.66	69.66	69.66	69.66
Heart	76.28	76.28	76.28	76.28	76.28	76.28
Liver	62.59	62.30	62.59	62.59	62.59	62.59
Vehicle	69.40	69.63	69.40	69.40	69.40	69.40
Vowel	98.12	98.12	98.12	98.12	98.12	98.12
Wine	94.41	94.41	94.41	94.41	94.41	94.41
Phoneme	71.00	70.87	71.00	71.00	71.00	71.00
Satimage	82.96	82.90	82.96	82.96	82.96	82.96
Texture	98.89	98.86	98.89	98.89	98.89	98.89
Average	80.78	**80.80**	80.78	80.78	80.78	80.78

reduce the computing time, in comparison to the *exhaustive search*, over the original data sets.

The point in this chapter is to compare both kind of techniques used to reduce the time to apply the NN rule (prototype selection and *efficient search*), not only on accuracy terms, but also in computing time terms.

7.2 Experimental Setup

During many years different efficient algorithms to find the nearest neighbour have been developed in order to avoid the exhaustive search. Here we compare the use of the reduction techniques in conjunction to the application of the traditional search, versus the use of several nearest neighbour efficient search techniques over original training sets. So, this chapter presents a comparative analysis of those two ways of application of the traditional 1-NN rule to sample classification.

The focus of these experiments is on comparing the efficient search algorithms introduced in Chapter 2 (that is, k-dimensional tree, vantage point-tree, the approach of Fukunaga and Narendra, LAESA, KAESA and GNAT) to the employment of traditional search techniques after applying some data reduction scheme. Thus, the efficient algorithms have been used over the original training sets (without any preprocessing), while the traditional search has been applied to the previously edited and/or condensed training set.

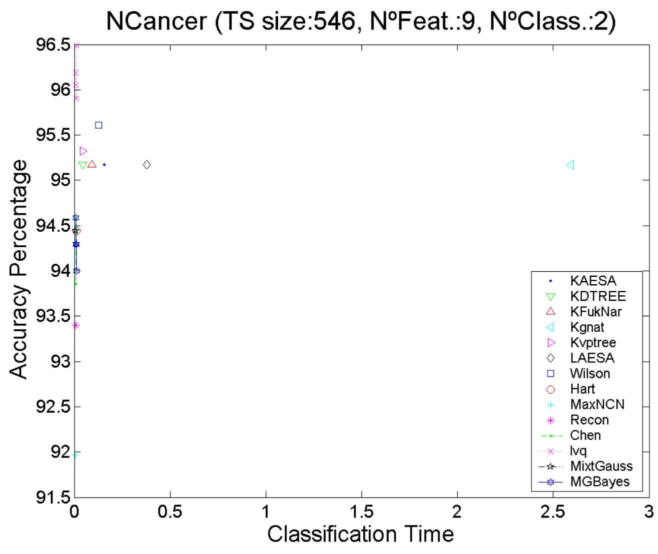
7.3 Quantitative Results

Table 7.1 presents the classification accuracy results for the nearest neighbour efficient search techniques tested in this chapter. Only accuracy rate is presented as the showing of size reduction rate makes no-sense as it is always 0 (as explained, the goal of these techniques is not to reduce the training set, but to process it efficiently). There, the average result for kvp-tree is highlighted in bold, as it is the highest, and the only one different, among all the nearest neighbour efficient search techniques tested. Although it is the highest result, it is very similar to the accuracy of the other efficient algorithms. Maybe due to the relatively small size of the databases in which this thesis is based on.

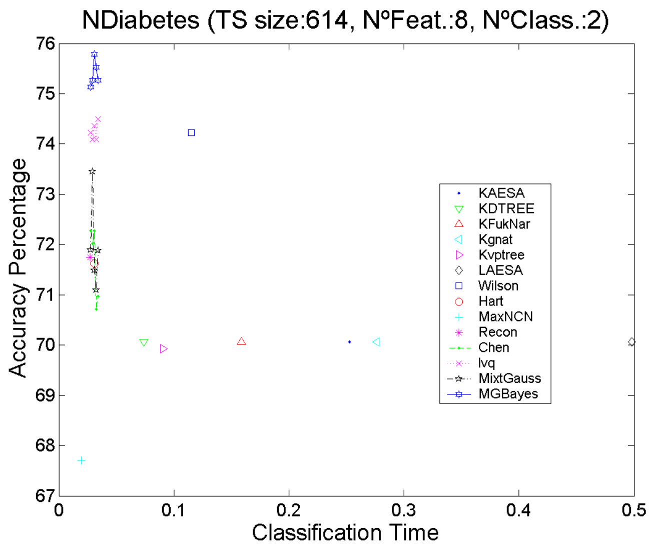
Figures 7.1-7.6 present the average computing times (that is, the time required to classify the whole test set) against the classification accuracy for the different algorithms over each database. Size reduction is not represented as other algorithms have already been compared, and nearest neighbour efficient search techniques do not apply any kind of reduction. However, it is to keep in mind that the "best" results for reduction techniques are not those which achieve the highest accuracy, but those that reach a good balance between accuracy and reduction. So, the highest results in accuracy are not directly representatives of the best reduction algorithms.

Figure 7.1 represents time and accuracy for the Cancer and Diabetes databases. In both we can observe that the time needed to process the test sets is not important, as it is very small in all the cases, specially with the LVQ, Hart, *MixtGauss*, Chen and *Reconsistent* algorithms. Having in mind that the scale in the classification accuracy axis is very small (from 91.5 to 96.5 for Cancer, and from 67 to 76 for Diabetes), we can say that among all the algorithms represented, the higher accuracy belongs to the LVQ algorithm, in the case of Cancer and to the *MGBayes* version, in the case of Diabetes. Time scale is even smaller using the Glass and Heart databases (Figure 7.2), where the best accuracy, and also the minor time, is obtained by the LVQ approach for both data sets.

Figure 7.3 represents results for the Liver and Vehicle databases. The higher accuracy is for the Wilson's editing using Liver and for the LVQ technique using Vehicle. In Figure 7.3 (b) we can observe that for the Vehicle database the time of the Fukunaga and Narendra's approach and the k-d tree and k-GNAT algorithms, is quite similar to the time needed using the reduced sets. Specially, it is to note that the minor time is needed for the use of kvp-tree, which obtains the highest accuracy in comparison to other effective search algorithms.

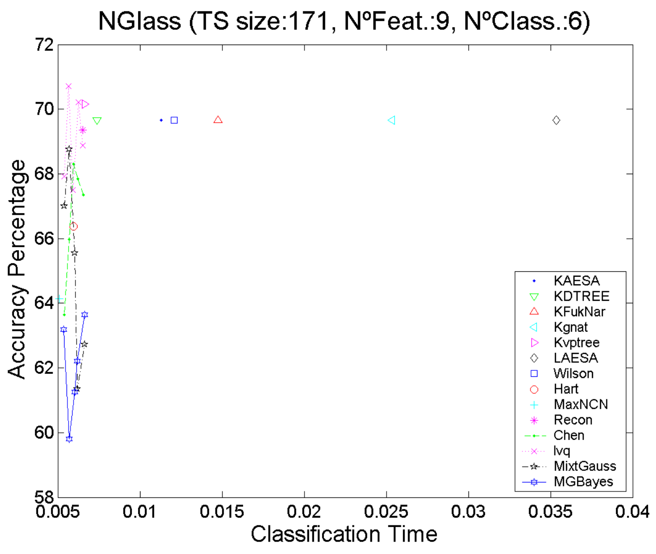


(a)

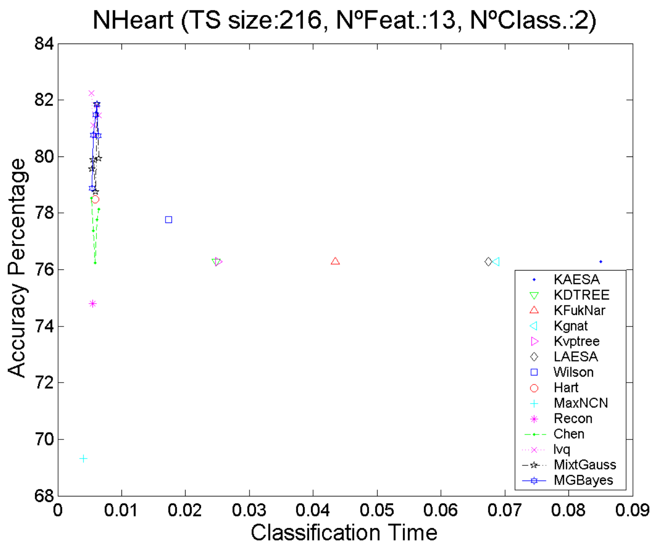


(b)

Figure 7.1: Classification accuracy (in %) and computing time (in secs.) for the Cancer (a), and Diabetes (b) databases.

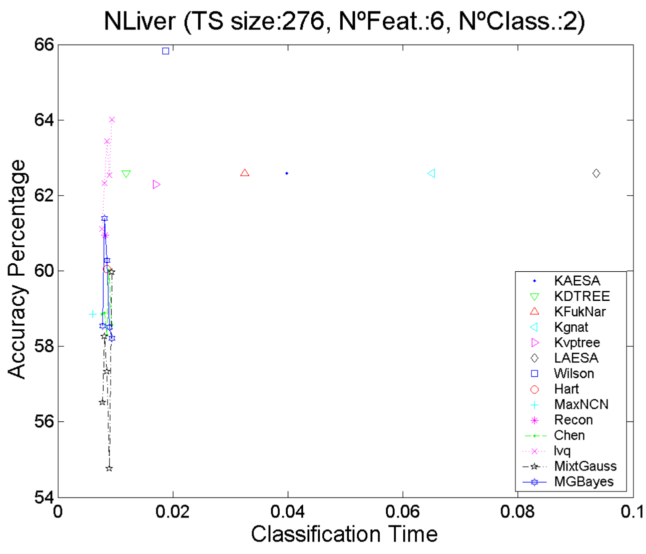


(a)

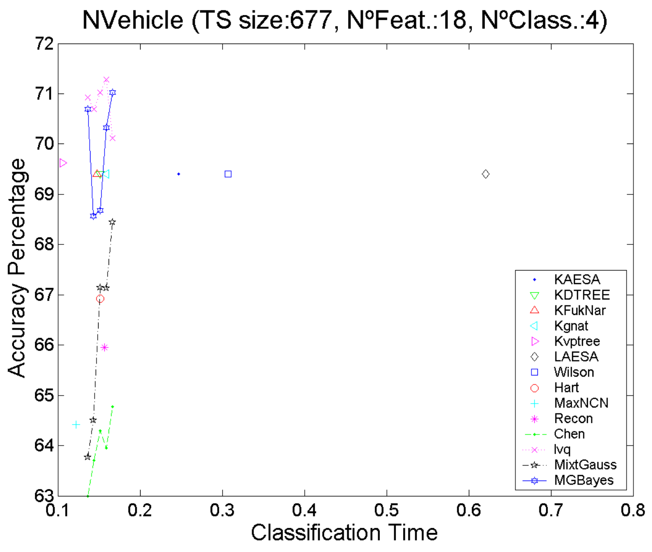


(b)

Figure 7.2: Classification accuracy (in %) and computing time (in secs.) for the Glass (a), and Heart (b) databases.

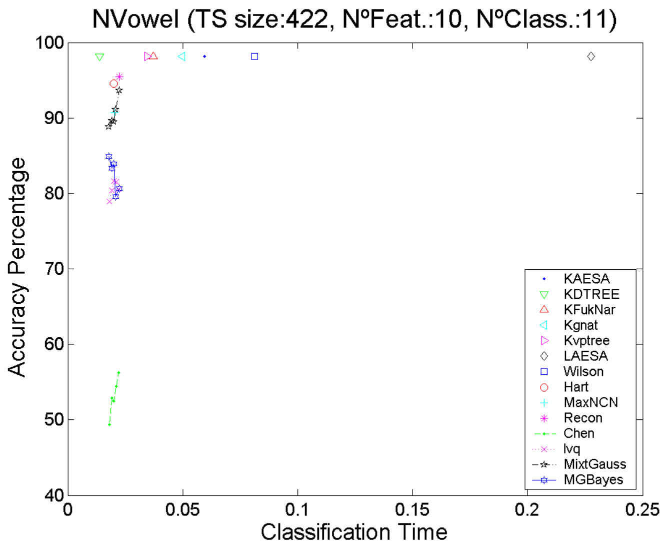


(a)

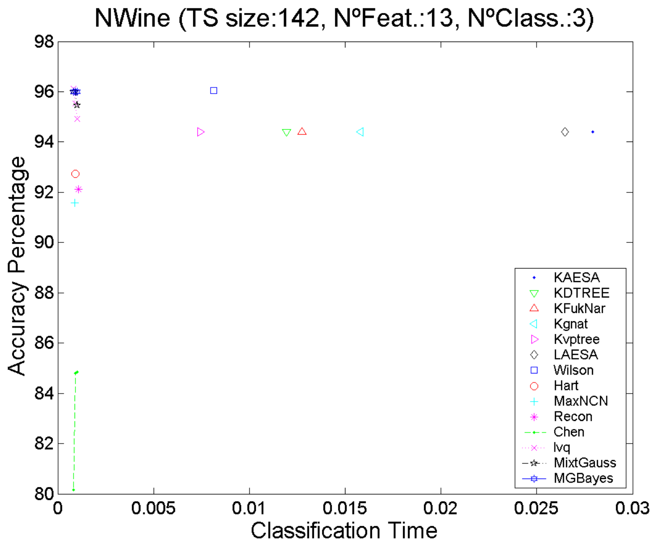


(b)

Figure 7.3: Classification accuracy (in %) and computing time (in secs.) for the Liver (a), and Vehicle (b) databases.



(a)



(b)

Figure 7.4: Classification accuracy (in %) and computing time (in secs.) for the Vowel (a), and Wine (b) databases.

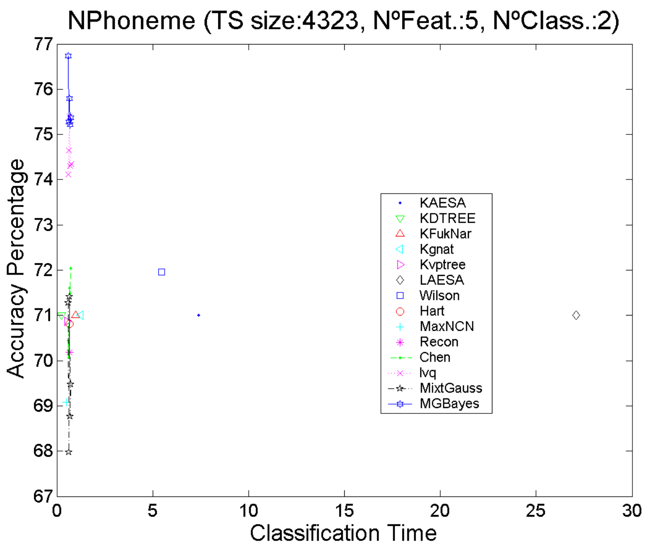
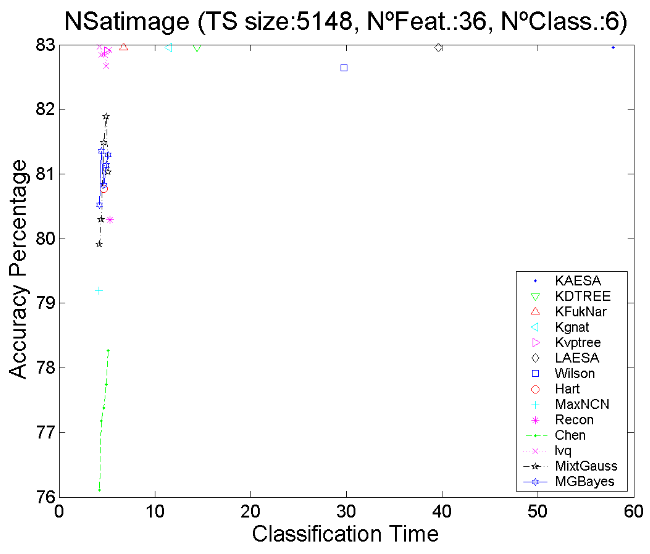


Figure 7.5: Classification accuracy (in %) and computing time (in secs.) for the *Phoneme* database.

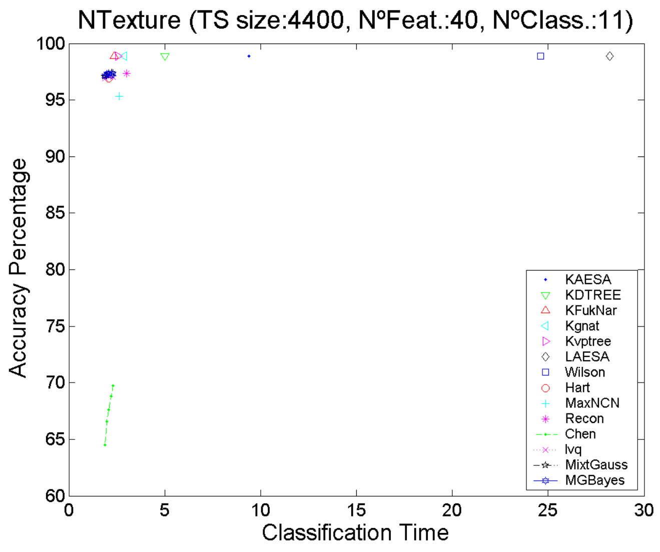
In Figure 7.4 (a), where results obtained using the Vowel database are shown, the highest accuracy is reached by efficient algorithms. In Figure 7.4 (b), where results for the Wine database are represented, the highest accuracy is obtained by the LVQ and *MGBayes* algorithms.

In Figure 7.5, the highest accuracy rate is reached by the *MGBayes* algorithm, while the point to highlight here is that, as the Phoneme database is higher in size than other databases shown before, the time needs have already changed. LAESA is the technique which needs more time. The next are KAESA andIn t Wilson's editing, as only the editing process has been applied to this reduced set, and the exhaustive search is used on it. Using Satimage and Texture (Figure 7.6), the time range increases, as those (including Phoneme) are the ones with the highest size among all the databases used for the experiments. KAESA, LAESA and Wilson's are the algorithms which need more time, in this order, using Satimage (Figure 7.6 (a)), meanwhile using Texture (Figure 7.6 (b)), the order changes to LAESA, Wilson's and KAESA.

In our opinion, the differences in relation to the classification time are not quite important, as the represented time is the total time (in secs.), and the maximum difference reached (Satimage) is less than a minute. It is to note that testing these algorithms with bigger test sets will produce a difference in the resulting time ob-



(a)



(b)

Figure 7.6: Classification accuracy (in %) and computing time (in secs.) for the Satimage (a), and *Texture* (b) databases.

tained by the use of each approach. Anyway, it is also to note that in general, less process time is needed when the *MaxNCN*, *Reconsistent* and *MixtGauss* algorithms are used. So, at least with a limited size database, algorithms introduced in this thesis perform better than the nearest neighbour efficient search techniques here tested.

In relation to accuracy, better results than expected are obtained for algorithms presented in this thesis. It is enough to compare figures in Table 7.1 to accuracy results in Table 6.1 to realise that the *MixtGauss* algorithm reaches a higher accuracy than any nearest neighbour efficient search Technique. *Reconsistent* and *MaxNCN* accuracy results in comparison to the best efficient result show a negative percent difference of only 1.6, and 4, respectively. And this accuracy results are obtained by using three different condensed sets, with a reduction rate of 89.1 for *MixtGauss*, 76.4 for *Reconsistent* and 81.5 for *MaxNCN*. This reduction percentages are important, as they mean a notable reduction in memory needs.

In relation to nearest neighbour efficient search techniques, the procedures that, in general, take more time are LAESA and KAESA. Results for the k-d tree, Fukunaga and Narendra's, k-gnat and kvp-tree approaches are, in general, quite similar in time. So, as the accuracy in average is slightly higher for kvp-tree, it can be named the best among the efficient searches, and for the databases, here tested.

7.4 Conclusions

This chapter presented a comparative analysis of two ways of application of the traditional 1-NN rule to object classification. On the one hand, exhaustive search over reduced sets. On the other hand, nearest neighbour effective search over original training sets.

Both of them present a good behaviour: low classification time, and high accuracy. However, in addition, the use of the reduction algorithms presented in this thesis implies less memory needs. Specially the *MixtGauss* algorithm, which obtains a slightly higher classification accuracy than the higher result for efficient searches, with an important size reduction. So, we conclude that the use of size reduction algorithms, versus efficient techniques over original sets, are preferable in data sets of these (small) sizes. The main reasons are that efficient algorithms do not add an important improvement in classification time nor accuracy (because of the use of these small databases) and, in addition, the memory needs are reduced when using prototype selection techniques, without an important lose in classification accuracy (even improving it sometimes).

We think a good idea would be to mix the use of reduction approaches and nearest neighbour efficient search techniques, as both of them perform well. This idea has not been applied to databases here used as it is to be applied for really bigger databases. There, we think the use of efficient search over reduced sets would offer lower memory and time needs, without a big reduction in classification accuracy. This is the reason why we will take this idea for future research.

Chapter 8

Final Conclusions

Contents

8.1 Contributions

The main goal of this thesis has been, on the one hand, the analysis of a set of techniques used in pattern recognition: classification and nearest neighbour search, editing and, especially, condensing techniques. This study has been reported in Part I. On the other hand, some alternative methods for condensing have been proposed in Part II, using the ideas of surrounding neighbourhood and Gaussian clustering. Besides, some classification and search techniques have been compared, and some discussion has been done about them in Part III.

At last but not least, it is important to say that every algorithm proposed in this thesis has been empirically evaluated by the use of some real problems represented in feature spaces (see Section 1.2), comparing their results with those obtained for the algorithms reported in Part I. This analysis permits us to realise the good performance of the algorithms presented in this thesis, and that they represent a good trade-off between the classification accuracy and reduction rates.

Main chapters reporting contributions are Chapter 3, devoted to new non-adaptive algorithms, and Chapter 4, that introduce new adaptive schemes. Particular contributions on these chapters are better explained in next subsections.

8.1.1 Non-Adaptive Contributions

In Chapter 3 some non-adaptive condensing algorithms have been presented. The main idea introduced is the use of surrounding neighbourhoods to cover the different areas and reduce the training set size as much as possible without an important decrease in classification accuracy. The Nearest Centroid Neighbour (NCN) rule had been used in the past in order to classify unlabelled objects. On the contrary, in the algorithms presented in Chapter 3 (and Chapter 4) it has been used to select (and replace), some representative instances from the training set, covering the new prototypes a bigger region than using the NN rule.

Later, in Chapter 5, results for non-adaptive algorithms have been compared in reduction rates and classification accuracy with those introduced in Part I. Taking into account these results, we can emphasise two of the algorithms introduced here, as having better results than others: *MaxNCN* (the best one) and *Reconsistent* (the third one, being Hart the second one). They are the bests as they obtain a good balance between set size reduction and classification accuracy, in addition to being the nearest ones to the ideal point (100, 100).

8.1.2 Adaptive Contributions

In Chapter 4, some different new adaptive condensing algorithms have been presented. The goal is to improve our results by locating prototypes in the exact point that makes the most accurate learning algorithm. On the one hand, some of the algorithms are based on the idea of surrounding neighbourhood. However, results for these adaptive schemes do not offer a strong improvement (in comparison to those from algorithms in Chapter 3), as it has been shown in Chapter 5, in spite of increasing the computational effort.

On the other hand, the main contribution in Chapter 4 is a generative approach which mainly uses mixtures of Gaussians. Some different initialisations and stopping criterions have been tested. Among all of them, *MixtGauss* can be highlighted as the best combination, in relation to the results obtained, and the computational effort needed. In Chapter 6, the results for this principal adaptive algorithm have been evaluated by classifiers such as the NN rule, constructed in a feature space, and linear (FLD) and quadratic (NQC) classifiers constructed in a dissimilarity space. These results were compared to those of other algorithms studied in Part I (Chapter 2: LVQ and Chen) and others presented in Part II (Chapter 3: *MaxNCN* and *Reconsistent*). On the one hand, from the experiments carried out, it seems that *MixtGauss* provides good results, i.e. higher accuracy rate than those obtained by other adaptive algorithms such as Chen's, while keeping the same size reduction. Moreover, it is to highlight that the differences between *MixtGauss* and Chen are more significant when a higher reduction in size is applied. On the other hand, these results show that *MixtGauss* and LVQ have a similar behaviour, even better than *MaxNCN* and *Reconsistent*. In addition, they allow the user to control the number of prototypes, and they represent a good trade-off between classification accuracy and size reduction rates. A general tendency in relation to which approach, LVQ or *MixtGauss*, obtain better results with a specific database, have been observed. Firstly, while LVQ seems to be the best option for data sets of two classes, *MixtGauss* tends to get better results for data sets with more than two classes. Secondly, while LVQ in general provides a better solution for the data sets with a smaller dimensionality, *MixtGauss* offers better results for those with a higher dimensionality, although there are exceptions in all the cases here commented.

8.1.3 Classification Rules Comparison

Additionally, the experiments in Chapter 6 compare classifiers such as the NN rule, constructed in a feature space, and FLD and NQC classifiers constructed in a dissimilarity space. In general, the best trade-off results (between accuracy and condensed set size) are the ones for the FLD. The results for the NQC are similar. That shows that a more complex classifier than a linear one is not necessarily yielding a better performance in dissimilarity spaces resulting from normalised vectorial data.

And finally, the results for the 1-NN rule are significantly worse, in the most of the databases. So, these results show that classification rules other from the 1-NN can be used, that lead to better results, independently of the considered condensing algorithms. Especially, the linear and quadratic normal density based classifiers built in dissimilarity spaces, are advised for problems with a small number of classes.

8.1.4 Nearest Neighbour Efficient Search Techniques Comparison

In Chapter 2, the bases of the nearest neighbour efficient search techniques have been reported. In Chapter 7 their computing time and classification accuracy have been presented in a comparative analysis. On it, two ways of applying the classical 1-NN rule to object classification are studied: exhaustive search over reduced sets, versus nearest neighbour effective search over original training sets.

Differences in time among the effective and the condensed techniques tested are not really important, as the highest difference observed is lower than a minute. Having this in mind, we can conclude that less computing time is needed when the *MaxNCN*, *Reconsistent* and *MixtGauss* algorithms are used. So, at least with a limited size database, algorithms introduced in this thesis perform more rapidly than the nearest neighbour efficient search techniques here tested.

In relation to accuracy, the *MixtGauss* algorithm reaches a higher accuracy than any nearest neighbour efficient search technique (being at the same time, more rapid). *Reconsistent* and *MaxNCN* obtain a slightly lower accuracy. So, *MixtGauss* is the algorithm which, with lower needs in time and memory, reaches the best accuracy.

In relation to nearest neighbour efficient search techniques, kvp-tree shows the best behaviour among the efficient searches here tested, as it obtains higher accuracy than the rest, and a quite similar computing time.

To conclude, we can affirm that both methodologies, set size reduction and efficient search obtain good results in a good time.

8.2 Conclusions

The main goal of this section is to enumerate the principal conclusions of the research here reported. In brief, they are listed below.

1. **Surrounding Neighbourhood**. Some (selective and generative) algorithms based on the use of the NCN rule have been presented. Among all of them, we can emphasise two, as having better results than others: *MaxNCN* and *Reconsistent*. See Chapters 3, 4 and 5.

2. **Mixtures of Gaussians**. A generative algorithm based on the use of mixtures of Gaussians has been presented too: *MixtGauss*. From the experiments carried out, it seems that *MixtGauss* provides good results (a good trade-off between classification accuracy and size reduction rates), even better than *MaxNCN* and *Reconsistent*. In addition, it allow the user to control the number of prototypes. See Chapters 4 and 6.

3. **Classifiers**. The results for LVQ, Chen, *MaxNCN*, *Reconsistent* and *MixtGauss* (Chapters 2, 3 and 4) have been evaluated by classifiers such as the NN rule, in the feature space, and FLD and NQC, in the dissimilarity space. From this study, in Chapter 6 we report that classification rules other from the 1-NN can be used, that lead to better results. Especially, the linear and quadratic normal density based classifiers built in dissimilarity spaces, are advised for problems with a small number of classes.

4. **Nearest Neighbour Efficient Search**. The use of exhaustive search over reduced sets, versus nearest neighbour efficient search over original training sets have been compared. The *MaxNCN*, *Reconsistent* and *MixtGauss* algorithms

are processed in the minor time, as well as *MixtGauss* reaches the highest accuracy. So, we highlight the latter algorithm as the best. It is to note that kvp-tree shows the best behaviour among the efficient searches here tested, as both methodologies obtain good results in a good time.

8.3 Future Work

The first idea about possible improvements is related to adaptive reduction, concretely to *Mixtures of Gaussians.* In the work presented here, the Gaussians are orthogonal (i.e. their principal and secondary diagonals are parallel to the X and Y axes). As any database is forced to have a determined distribution around parallels to axes X or Y, we think that the *MixtGauss* algorithm would improve its results by permitting the Gaussians in charge of representing every original instance by their means to have any orientation. These would mean a bigger matrix calculation effort that we think could be bearable.

In addition, it would be very convenient to study the relation between the number of classes and the dimensionality of a database, and the best algorithm to use, LVQ or *MixtGauss.* This study could be done over several synthetic databases, as they allow one to control the number of classes and the quantity of dimensions.

In relation to the dissimilarity field, it seems very interesting the application of utilities from the representation space, as well as the application of utilities from the dissimilarity field on the representation space. It seems a very interesting world for study, still new for us in different aspects.

In Chapter 4 we use the local covariance matrix in the *MixtGauss* procedure to find prototypes. After that the NN rule was used based on Euclidean distances, neglecting the local correlation in the data. A more consistent approach would be to replace the Euclidean distance by the Mahalanobis distance based on the covariance matrix used for finding the prototype under consideration. This will increase, however, the computational complexity.

Another point for future research may focus on comparing a (non-)linear SVM built in the original feature vector space and the SVM in a dissimilarity space defined by a small set of optimised prototypes, as in such a case the SVM will rely on support vectors which are objects found in a complete dissimilarity space, hence they rely on the distances to *all* these optimised representatives.

In relation to efficient search, we think a good idea would be to mix the use of reduction approaches and nearest neighbour efficient search techniques, as both of them perform well. This idea has not been applied to databases here used as it is to be applied for heavy databases. There, we think the use of efficient search over reduced sets would offer lower memory and time needs, without a big reduction in classification accuracy.

8.4 Related Publications

Part of the results of this thesis have already been published. The full references are as follows:

International Journals

- **PR39 06**: M. Lozano, J.M. Sotoca, J.S. Sánchez, F. Pla, E. Pękalska, R.P.W. Duin, Experimental Study on Prototype Optimisation Algorithms for Prototype-based Classification in Vector Spaces, Pattern Recognition 39 (2006) 1827-1838.

International Book Chapters and Lecture Notes

- **FAIA113 04**: M. Lozano, J.M. Sotoca, J.S. Sánchez, F. Pla, An Adaptive Condensing Algorithm Based on Mixtures of Gaussians, Setè Congrés Català d'Intel·ligencia Artificial, Barcelona (Spain), Frontiers in Artificial Intelligence and Applications, Vol. 113, IOS Press, J. Vitrià et al. (Eds.), (2004) 225-232.

- **LNAI3040 04**: M. Lozano, J.S. Sánchez, F. Pla, Using the Geometrical Distribution of Prototypes for Training Set Condensing, Current Topics in Artificial Intelligence, Lecture Notes in Artificial Intelligence, Vol. 3040, R. Conejo, M. Urretavizcaya, J.L. Pérez de la Cruz (Eds.), Springer-Verlag, (2004) 618-627.

- **LNCS2652 03**: M. Lozano, J.S. Sánchez, F. Pla, Reducing Training Sets by NCN-based Exploratory Procedures, Lecture Notes in Computer Science, Vol. 2652, F. J. Perales et al (Eds.), Springer-Verlag, (2003) 453-461.

National Conferences

- **CAEPIA 03 (Best Paper Award)**: M. Lozano, J.S. Sánchez, F. Pla, Training Set Size Reduction by Replacing Neighbouring Prototypes, 10th. Conference of the Spanish Association for Artificial Intelligence, Vol. 1, (2003) 37-46, San Sebastián (Spain).

Part IV

Resúmenes Tesis

Chapter 9

Objetivos, Planteamiento y Conclusiones

Contents

Chapter 9 presents some requirements from the PhD regulation for those PhD reported in a different language than the official ones, to be in Spanish or Valencià. Mainly, motivation and general objectives, approach and methodology used, contributions, conclusions, and future work. All of them have already been reported in English, in previous chapters.

Este capítulo aparece para dar respuesta a la propuesta de molificación de la normativa de doctorado, aprobada por el Consejo de Gobierno N^{o} 16 de 14/07/2005, en referencia a las tesis escritas en una lengua diferente a las oficiales (valenciano y castellano). Por tanto, a continuación se desarrollan los puntos solicitados en el Punto 5 del Apartado XIII de la misma. Estos son: objeto y objetivos de la investigación, planteamiento y metodología utilizados, aportaciones, conclusiones, y futuras líneas de investigación. Todos ellos ya han sido desarrollados en ingles, en capítulos anteriores.

9.1 Motivación y Objetivos Generales

El título de esta tesis es *Técnicas de Reducción de Datos en Procesos de Clasificación*. Por tanto, ésta es la temática en la que se encuadra nuestro objetivo principal. Por *Técnicas de Reducción de Datos* entendemos aquellas que se encargan de hacer disminuir la cantidad de información para reducir las necesidades tanto de memoria como de tiempo de ejecución. Tradicionalmente, el concepto de *Reducción de Datos* ha recibido diferentes nombres (editado, condensado, filtrado...) dependiendo del objetivo perseguido. Dependiendo del *objeto* de la reducción, se observan dos posibles acepciones: *selección de características* y *selección de prototipos*. La primera se refiere a la reducción en la cantidad de características, y la segunda se basa en la selección de algunas instancias, entre todas las disponibles. No se hace referencia en esta tesis a la *selección de características*, sino a la *selección de prototipos*. Por tanto, el objetivo principal que se persigue es la introducción, el análisis y la evaluación de diferentes técnicas dedicadas a reducir el número de instancias.

Para continuar con el título, *Procesos de Clasificación* se refiere a aquellas técnicas que se encargan de clasificar o etiquetar una nueva muestra por medio de las instancias o ejemplos presentes en el conjunto de entrenamiento. Y esto es lo que tratamos de reducir: el conjunto de entrenamiento. Por tanto, la idea es representar todas las instancias incluidas en dicho conjunto mediante unos pocos representantes, elegidos de modo que sean lo más efectivos posible. Es decir, que conserven la misma precisión de clasificación, o muy parecida. Por tanto, lo que se pretende principalmente en esta tesis es el diseño de unos algoritmos que permitan reducir los requisitos de tiempo y memoria, a la vez que mantengan la precisión en la clasificación.

Estos objetivos están motivados por el problema que se desarrolla a continuación. Hoy en día, en muchos campos (como por ejemplo en imágenes multiespectrales, clasificación de textos, biométricas y recuperación de bases de datos multimedia, entre otros) el tamaño de los datos es tan extremadamente grande que los sistemas de tiempo real no pueden abordar su procesamiento, debido al tiempo y a la memoria necesarios para ello. Bajo estas condiciones, clasificar, entender o comprimir la información puede convertirse en una tarea verdaderamente problemática. Esto se hace especialmente grave en los casos en los que se utilizan algoritmos de aprendizaje basados en la distancia, como por ejemplo la regla del vecino más cercano (Nearest Neighbour; NN). Brevemente, el algoritmo se encarga de buscar entre todos los ejemplos del conjunto de entrenamiento (grandes requisitos de memoria) para clasificar un nuevo caso (proceso lento de clasificación). Por otra parte, como la regla del vecino más cercano almacena todos los ejemplos en el conjunto de entrenamiento, los ejemplos erróneos también son incluidos. Como consecuencia, estos últimos pueden hacer disminuir la precisión de clasificación.

Entre las muchas propuestas para solucionar este problema, existe un método tradicionalmente utilizado que consiste en eliminar algunos ejemplos del conjunto de entrenamiento (reducción de datos). En la bibliografía de Reconocimiento de Formas estos métodos encargados de reducir el tamaño del conjunto de entrenamiento se conocen como *selección de prototipos* [Devijver(1982)]. Entre los métodos de selección de prototipos se pueden diferenciar dos familias. En primer lugar, aquellos algoritmos que se encargan de eliminar instancias mal etiquetadas del conjunto de entrenamiento, evitando al mismo tiempo posibles superposiciones entre áreas de clases diferentes. Esto se conoce como *editado*. En segundo lugar se encuentran las técnicas de *condensado*. Estas se encargan de seleccionar (o modificar) un pequeño subconjunto de instancias para que representen el conjunto de entrenamiento, sin perder por ello su precisión de clasificación.

Las muchas propuestas existentes en relación con métodos de *condensado* se pueden dividir en dos grupos o familias principales. Por una parte, aquellos algoritmos que seleccionan un subconjunto entre los ejemplos originales del conjunto de entrenamiento [Dasarathy(1994), Aha et al.(1991), Toussaint et al.(1985), Hart(1968), Tomek(1976)] (algoritmos selectivos). Por otra parte, aquellas técnicas que los modifican [Sánchez(2004), Kohonen(1995), Chen & Jozwik(1996), Chang(1974)] (algoritmos adaptativos). En esta tesis se introducen varios algoritmos encargados de solucionar el problema de la reducción del tamaño del conjunto de entrenamiento en la medida apropiada, tanto mediante métodos selectivos como adaptativos. El objetivo principal en ambos casos es obtener una reducción considerable de tamaño, sin disminuir por ello la precisión de clasificación. Es decir, lo que se busca es un equilibrio entre ambas, reducción y precisión.

En este estudio se comparan varios métodos tanto de selección como de generación de instancias del conjunto condensado, en combinación con tres clasificadores basados en la distancia Euclídea. Además, el comportamiento de los algoritmos que aquí se presentan es comparado también con algunas técnicas de búsqueda efectiva.

Los métodos de condensado que se comparan en esta tesis son algunos de los clásicos: Chen, Hart y LVQ, además de la versión RSP3; y algunos de los implementados durante el desarrollo de esta tesis, como son: *MaxNCN*, *Iterative MaxNCN*, *Iterative kNeighbours*, *Consistent*, *Reconsistent*, *Centroide*, *WeightedCentroide* y *MixtGauss*. En cuanto a los métodos de clasificación que se comparan, el primero es la regla del vecino más cercano. Ésta se encarga de asignar a una nueva muestra la etiqueta o clase del ejemplo más cercano del conjunto de entrenamiento (o del representante más cercano, en el caso del conjunto condensado). El resto de métodos a comparar son el *Fisher Linear Discriminant* (FLD), y el Quadratic Normal density based Classifier (NQC), ambos entrenados en espacios de disimilaridad. Las técnicas de búsqueda efectiva que se comparan en los experimentos son las siguientes: k-d tree, la propuesta de Fukunaga y Narendra, vp-tree, GNAT, LAESA y KAESA.

9.2 Planteamiento y Metodología Utilizados

Tal y como se ha expresado anteriormente, el objetivo de esta tesis es principalmente la reducción del conjunto de entrenamiento, sin que esto implique una reducción importante en la precisión de clasificación. Se trata de cubrir completamente el área de cada clase con la unión de las regiones de influencia de algunos ejemplos del conjunto de entrenamiento seleccionados con este fin. Esto se basa en la siguiente idea intuitiva: en reducción de datos, la selección de algunos representantes que cubran la zona de influencia de cada clase, favorece la precisión de clasificación para muestras de esas clases. Además, una vez que el área de una clase está completamente cubierta por las regiones de los representantes más efectivos, cuanto menos representantes se conserven en el conjunto condensado, mejor se considera el resultado final, debido a la reducción de datos. Por tanto, en esta tesis se presentan diferentes algoritmos encargados de buscar los representantes más eficientes, y en la menor cantidad posible para las diferentes áreas de cada clase.

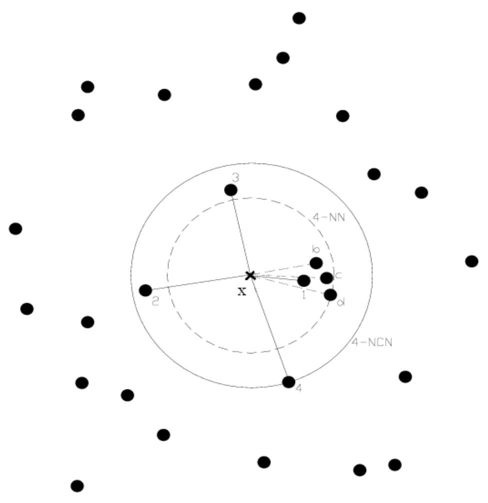
Para aplicar las ideas especificadas anteriormente nos basamos en diferentes métodos. Por una parte, en la regla Nearest Centroide Neighbour (NCN), y por otra parte en Gaussianas.

9.2.1 Sobre la regla NCN

La distribución geométrica de las instancias de un conjunto de entrenamiento puede resultar más importante que simplemente la distancia entre ellas. En este sentido, las reglas basadas en el concepto de vecindad envolvente [Sánchez et al.(1997)B] tratan de obtener información más útil sobre las instancias en el conjunto de entrenamiento, y especialmente información relacionada con aquellos ejemplos cercanos a las fronteras de decisión. Esto se puede conseguir teniendo en cuenta no sólo la proximidad de una muestra dada a las instancias, sino también la distribución de dichas instancias alrededor del objeto. Hasta el momento, la regla NCN ha sido utilizada como clasificador [Sánchez et al.(1997)B], obteniendo mejores resultados que otras reglas utilizadas tradicionalmente, como por ejemplo la regla del vecino más cercano. La novedad en cuanto a la aplicación de la regla NCN reside en que aquí se utiliza como método de condensado.

Chaudhuri [Chaudhuri(1996)] fue quien propuso este nuevo concepto de vecindad, el cual es una realización particular del concepto de vecindad envolvente. Para calcular el vecindario NCN se procede de la siguiente manera. Sea p un punto dado,

Figure 9.1: Representación del concepto de vecindario NCN.



cuyos k vecinos de tipo NCN se deben encontrar en un conjunto de entrenamiento como el siguiente $X = \{x_1, \ldots, x_n\}$. Esos k vecinos se buscan mediante un procedimiento iterativo que sigue los pasos que se indican a continuación:

1. Seleccionar el primer vecino NN de p, q_1, que será a su vez su primer vecino NCN.

2. Definir el i-ésimo vecino NCN, q_i, $i \geq 2$ como aquel que haga que el centroide de los vecinos NCN seleccionados, incluyéndose a él mismo, $q_1, \ldots, q_i$ sea el más cercano a p.

El vecindario que se obtine mediante este algoritmo satisface algunas propiedades interesantes que serán utilizadas para reducir el tamaño de conjuntos de entrenamiento, tanto mediante la selección de ejemplos como mediante la generación de nuevos prototipos. En particular, merece la pena destacar que la regla NCN es incremental y que las instancias alrededor de una muestra dada, p, representan una distribución que tiende a envolver a p, compensando la distribución de instancias a su alrededor. También es importante resaltar que en general, el área de influencia de la regla NCN es mayor que la región determinada por la regla NN. Lo cual puede verse en la Figura 9.1, donde se representan las regiones definidas por 4 vecinos NCN, y por 4 vecinos NN.

Algorithm 9.1: Pseudocódigo del algoritmo *MaxNCN*.

```
**for** $i = each\_instance(TS)$ **do**
  $neighbours\_number[i] = 0$
  $neighbour = next\_NCN(i)$
  **while** $neighbour.class == i.class$ **do**
    $neighbours\_vector[i] = Id(neighbour)$
    $neighbours\_number[i] + +$
    $neighbour = next\_NCN(i)$
  **end while**
**end for**
**while** $Max\_neighbours() > 0$ **do**
  $EliminateNeighbours(id\_Max\_neighbours)$
**end while**
```

A continuación se presentan los algoritmos más representativos de entre todos los introducidos que utilizan esta metodología: *MaxNCN* y *Reconsistent*.

9.2.2 MaxNCN

El algoritmo *MaxNCN* está basado en la regla NCN, y utiliza la técnica de búsqueda presentada anteriormente. Un subconjunto de instancias es seleccionado de entre todas las que forman el conjunto de entrenamiento, garantizando la óptima distribución de las mismas con respecto a sus vecinos NCN. El uso del vecindario NCN de una muestra dada puede facilitar información local sobre la forma de la *distribución de probabilidad de la clase*, la cual depende de la naturaleza y de la clase de sus vecinos NCN, es decir, de las instancias en su área envolvente.

La idea en la que se basa este algoritmo es que las instancias que pertenecen a la misma clase están localizadas en un área cercana, y que por tanto pueden ser sustituidas por un único representante, sin afectar significativamente a las fronteras de decisión originales.

En una primera inicialización, todas las instancias del conjunto de entrenamiento son consideradas como representantes. El algoritmo trata de reemplazar un grupo de ejemplos vecinos de la misma clase, por uno de ellos. Para decidir que subconjunto de instancias va a ser sustituido, la idea de vecindario NCN que aquí se utiliza, es introducida ahora.

Definición El vecindario NCN de una instancia p es el conjunto de vecinos de p, calculados mediante la regla NCN hasta que se alcanza un vecino que pertenece a una clase diferente a la de p.

Las instancias que pertenecen al vecindario NCN de p son consideradas sus vecinos

(vecinos NCN).

Por tanto, para cada ejemplo p del conjunto de entrenamiento, se calcula su vecindario NCN. La instancia con mayor número de vecinos NCN es declarada representante de su grupo. Dada el área de influencia definida por la distribución del vecindario NCN, todos los miembros del subconjunto, a excepción del representante, pueden ser eliminados del conjunto de entrenamiento.

Después, y teniendo en cuenta las instancias remanentes, el algoritmo se encarga de actualizar el número de vecinos NCN que conserva cada uno, dado que algunos han podido ser eliminados como vecinos de un representante ya seleccionado. Esto se repite hasta que no queda ningún vecindario que pueda ser reemplazado. Este esquema básico recibe el nombre de *MaxNCN*, y se presenta en el Algoritmo 9.1.

9.2.3 Reconsistent

El objetivo tenido en cuenta al introducir la familia de algoritmos consistentes es precisamente su consistencia, o casi consistencia en el caso del algoritmo que se presenta en esta subsección.

Definición Se dice que un conjunto condensado, R, es *consistente* con respecto a un conjunto de entrenamiento X, si R clasifica correctamente todas las instancias de X, mediante la regla 1-NN.

Es decir, si el error de clasificación estimado mediante la asignación de todas las instancias del conjunto de entrenamiento a la clase de su vecino más cercano en el conjunto condensado es 0; véase [Dasarathy(1991), Hart(1968)]. Por tanto, dado un conjunto de instancias encargadas de representar la distribución de la clase, el porcentaje de clasificación del conjunto de entrenamiento puede ser utilizado para medir la consistencia de dicho conjunto. Mientras la condición de consistencia se mantiene, el conjunto condensado obtenido es considerado mejor cuanto menor sea el número de representantes.

Los algoritmos presentados bajo esta idea son importantes modificaciones sobre el algoritmo básico, *MaxNCN*, en aras de conseguir una mayor consistencia del conjunto condensado [Lozano(2004)A]. La idea principal aquí desarrollada es que la consistencia de un conjunto condensado respecto al conjunto de entrenamiento debe incidir en una mayor precisión en la clasificación. Mediante la aplicación del algoritmo *MaxNCN*, algunas instancias cercanas a las fronteras de decisión pueden ser eliminadas debido al orden en el que las instancias son tomadas durante el proceso de condensado. Varios algoritmos han sido implementados con el objetivo de solucionar este problema, entre ellos el que en esta subsección se presenta.

Algorithm 9.2: Pseudocódigo del algoritmo *Reconsistent*.

```
for $i = each\_instance(TS)$ do
   $neighbours\_number[i] = 0$
   $neighbour = next\_NCN(i)$
   while $neighbour.class == i.class$ do
      $neighbours\_vector[i] = Id(neighbour)$
      $neighbours\_number[i] + +$
      $neighbour = next\_NCN(i)$
   end while
end for
while $Max\_neighbours() > 0$ do
   $EliminateNeighbours(id\_Max\_neighbours)$
end while
$count = 0$
for $i = each\_instance(TS)$ do
   if $Classify(i)! = i.class$ then
      $incorrect\_class[count + +] = i$
   end if
end for
for $i = each\_instance(incorrect\_class[])$ do
   $neighbours\_number\_inc[i] = 0$
   $neighbour\_inc = next\_NCN\_inc(i)$
   while $neighbour\_inc.class == i.class$ do
      $neighbours\_vector\_inc[i] = Id(neighbour\_inc)$
      $neighbours\_number\_inc[i] + +$
      $neighbour\_inc = next\_NCN\_inc(i)$
   end while
end for
while $Max\_neighbours\_inc() > 0$ do
   $EliminateNeighbours\_inc(id\_Max\_neighbours\_inc)$
end while
$AddCondensedIncToCondensedTS()$
```

Reconsistent

El procedimiento comienza con la aplicación de los pasos del algoritmo *MaxNCN* sobre un conjunto de entrenamiento, para obtener un conjunto reducido, R. Entonces, cada instancia del conjunto de entrenamiento es testeado mediante la regla 1-NN, con respecto al conjunto R. Todas las instancias mal clasificadas en este test forman un nuevo grupo, el cual es condensado utilizando R como conjunto de referencia. Finalmente, este nuevo conjunto condensado de instancias mal clasificadas es añadido a R, con lo que se obtiene el conjunto condensado final. Éste, se dice que casi es consistente dado que su consistencia no vuelve a ser comprobada una segunda vez. El procedimiento *Reconsistent* se presenta en el Algoritmo 9.2.

9.2.4 Sobre Gaussianas

En general en un problema de reconocimiento de formas, tenemos un conjunto de entrenamiento con n instancias, donde cada instancia x es un punto, en un espacio de características d-dimensional, $x = [x_1, \dots, x_d] \in \Re^d$. Entonces, para una muestra dada, el objetivo consiste en asignarle la clase correcta donde $\Theta = \{c_1, \dots, c_J\}$ es el conjunto finito de las J clases del conjunto de entrenamiento. Los diferentes puntos del conjunto de entrenamiento respetan una distribución de clase espacial, según su verdadera función de densidad de probabilidad condicional de clase $P(x|c_j)$ y la respectiva probabilidad *a priori* $P(c_j)$, $c_j \in \Theta$. De este modo, podemos decir que un vector x puede ser óptimamente clasificado utilizando la regla de Bayes o de máxima probabilidad *a posteriori* basada en el conocimiento de las componentes $P(c_j)P(x|c_j)$ para cada clase c_j.

En la práctica, esta función de densidad de probabilidad condicional de clase no tiene asumida ninguna estructura subyacente ni se necesita un conocimiento a priori sobre las formas de esos *pdf*s para resolver el problema. Así, un modo de aproximarlo es mediante la estimación de una función de densidad de la distribución.

Una forma natural de lidiar con un estimador de densidad es considerando una mezcla (mixtura) de densidades de modos. Una aproximación para mantener la capacidad de $P(x|c_j)$ es mediante la reflexión de la estructura local de la distribución por medio de mezclas de modos $P_m(x|c_j)$. Este estimador de densidad es utilizado en la literatura [Novovicova et al.(1996)]. Nosotros lo utilizamos asumiendo la independencia estadística de las características. De hecho, calculamos los estimadores de un modo paramétrico. Consecuentemente, se puede utilizar la regla de Bayes, y cada modo es estimado mediante el producto de probabilidades en cada característica:

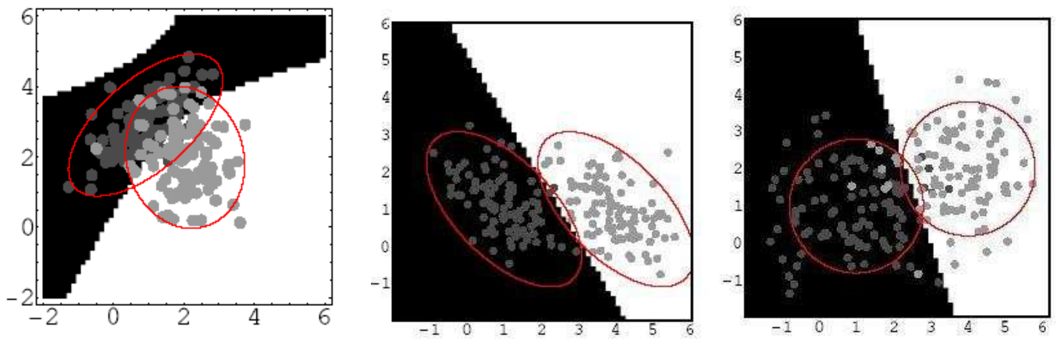
$$P(x|c_j) = \sum_{m=1}^{M} \alpha_{m|j} P_m(x|c_j) = \sum_{m=1}^{M} \alpha_{m|j} \prod_{k=1}^{d} N(x_k; \mu_{m|kj}, \sigma_{m|kj}) \tag{9.1}$$

donde M es el número de modos y $\alpha_{m|j}$ es una probabilidad *a priori* del modo m-ésimo en la clase c_j. Por tanto, el clasificador de Bayes es:

$$C_{Bayes} = \max_{j=1..J} P(c_j)P(x|c_j) \tag{9.2}$$

donde $P(c_j)$ es la probabilidad de clase *a priori*.

Por tanto, la caracterización de la distribución la podemos considerar como un problema de aprendizaje paramétrico no supervisado [Duda & Hart(1973)] uti-



(a) (b) (c)

Figure 9.2: Gaussianas elípticas determinando límites cuadráticos (a) y límites lineales (b), y Gaussianas esféricas determinando límites lineales.

lizando una mezcla de modos de una distribución normal multivariante. Representamos las Gaussianas multidimensionales como un producto de distribuciones normales univariantes, con $\mu_{m|kj}$ y $\sigma_{m|kj}$ como media y desviación típica, respectivamente. El algoritmo EM [Dempster(1977)] se utilizará para estimar estos modos desconocidos.

9.2.5 MixtGauss

El objetivo último del procedimiento de condensado propuesto consiste en obtener una distribución de probabilidad que represente una clase, para cada clase del conjunto de entrenamiento. Cuando cada clase se describe mediante una mezcla de distribuciones Gaussianas multivariante, se mantienen tanto las formas generales de las clases, como las fronteras de decisión. Si además asumimos la independencia estadística de las características, la componente m-ésima en un modelado de mezcla de la clase c_j es una distribución Gaussiana multivariante expresada mediante un producto de distribuciones normales univariantes, $N(\mu_{m|kj}, \sigma_{m|kj})$. Es conveniente destacar que esto es equivalente a una distribución Gaussiana multivariante de forma elíptica $N(\mu_{m|j}, diag(\sigma_{m|j}))$, paralela a los ejes de $\Re^d$.

La densidad Gaussiana multidimensional es un modelo típico utilizado para describir una densidad de probabilidad. Los únicos parámetros a estimar son las medias y las matrices de covarianza. Mediante la introducción de modelos estimados en el test de Bayes, se obtienen límites cuadráticos (véase la Figura 9.2 (a)) relacionada con la distancia de Mahalanobis a cada clase. Además, si las matrices de covarianza de las clases son iguales, los términos cuadráticos se anulan, y los límites se convierten en lineales (hiperplanos; véase la Figura 9.2 (b)).

Si los agrupamientos (clusters) tuviesen formas esféricas del mismo tamaño, y las clases fuesen equiprobables, el clasificador óptimo sería el vecino más cercano a la media de cada clase (véase la Figura 9.2 (c)); el llamado clasificador de la media más próxima. Por tanto, si el algoritmo que aquí se presenta fuese el encargado de cubrir el área de cada clase equiprobable con Gaussianas esféricas del mismo tamaño, los centros de influencia serían exactamente las medias de las Gaussianas.

De todos modos, dado que las simplificaciones supuestas nunca tendrán lugar en la práctica, intentamos entrenar a nuestro clasificador para datos reales utilizando Gaussianas de diferentes tamaños y formas.

Además, se aplicará un criterio de consistencia sobre los prototipos que representen a cada clase. Cuanto más Gaussianas se incluyan en la distribución, más exacta será la representación de las fronteras de decisión, y consecuentemente, más precisa será la clasificación. Por ejemplo, en el algoritmo de Hart, el criterio de consistencia se basa en la correcta clasificación 1-NN de todas las instancias en el conjunto de entrenamiento utilizando el conjunto de condensado.

Sin embargo, en el método aquí presentado se utiliza un criterio de consistencia diferente, parecido al de la Definición 9.2.3.

Definición Un conjunto condensado R, se dice que es *quasi-consistente* con respecto a un conjunto de entrenamiento X, si el error de clasificación es (suficientemente) pequeño, cuando la clase para cada elemento en X es estimada por la regla NN con respecto a R.

Dado un conjunto de prototipos que representan la distribución de una clase, podemos utilizar el porcentaje de clasificación del conjunto de entrenamiento para valorar cuan consistente (según la definición anterior) es.

Mientras que la condición de *quasi-consistencia* se mantenga, cuanto menor sea el número de prototipos utilizados para representar una clase en el conjunto condensado, mejor se considera el resultado. Por otra parte, para algunas aplicaciones es útil determinar anticipadamente el tamaño del conjunto condensado.

Para buscar una distribución condensada de elementos, imaginemos un problema con dos clases de distribución Gaussiana en un espacio Euclídeo. El mejor conjunto condensado será la media de la función de densidad de probabilidad Gaussiana en cada clase. Por tanto, supongamos una distribución de clase que pueda modelarse mediante una mezcla de Gaussianas. Los centros de esas Gaussianas pueden convertirse en el conjunto de prototipos que representen la distribución de clase, es decir, el conjunto condensado. Entonces, si cada clase se cubre mediante una mezcla de Gaussianas, la unión de las áreas de influencia de todas las Gaussianas cubriría completamente el área donde cada clase está definida.

Además, suponiendo prioridades a priori iguales para todas las Gaussianas, si

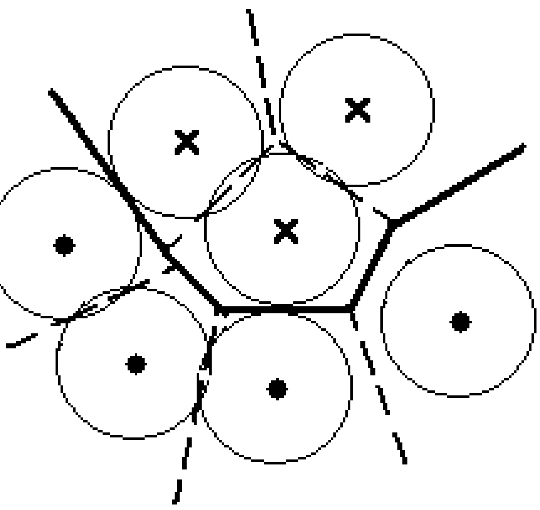


Figure 9.3: Equivalencia entre las fronteras de Voronoi y los límites de las diferentes áreas de influencia de Gaussianas circulares iguales.

fuesen forzadas a tomar la forma hiperesférica (misma matriz diagonal de covarianza con la misma varianza en cada característica), el uso de Gaussianas sería equivalente al clasificador de distancia mínima, 1-NN [Duda et al.(2001)], cuando se asocia cada Gaussiana a una clase. Por tanto, los límites entre áreas de influencia de Gaussianas vecinas se corresponderán con las fronteras de Voronoi (véase la Figura 9.3).

Inicialización de MixtGauss

Obsérvese que a pesar de que el uso del algoritmo EM está muy extendido, hay que ser consciente de sus inconvenientes [Figueiredo & Jain(2002)]. Como método que trabaja en vecindarios locales, es sensible a la inicialización porque la función de probabilidad de un modelo de mezclas no es unimodal. Otro punto a destacar en modelado de mezclas es la selección del número de componentes. Si son demasiadas, la mezcla puede sobreajustar (over-fit) los datos, mientras que una mezcla con componentes insuficientes puede no ser suficientemente flexible para aproximar el verdadero modelo subyacente.

Utilizamos mezclas de Gaussianas para obtener una estimación de la función de densidad de probabilidad. Asumimos que todas las componentes tienen la misma forma funcional. Por ejemplo, que todas son Gaussianas d-variantes, estando cada una completamente caracterizada por el vector paramétrico Θ_m. Y cuanto más componentes Gaussianas se incluyan en la mezcla, más precisa será la representación de las clases y las fronteras de decisión.

Teniendo en cuenta este hecho, debemos facilitar un número de Gaussianas M que representen cada distribución de clase. Este número corresponde a la cantidad de prototipos que se generarán para cada clase en el conjunto condensado final. En el primer paso, cada clase se representa por M Gaussianas. Aquí se presentan tres inicializaciones diferentes.

1. En la primera, el rango de cada parámetro se divide entre $(M - 1)$ (de este modo obtenemos tantos límites de intervalos como Gaussianas necesitamos). A cada paso se representa una Gaussiana con todos los valores de los parámetros para ese paso. Así se obtiene una diagonal d-dimensional, representada por centros de Gaussianas. Para cada Gaussiana, la varianza inicial toma el valor de un décimo del rango de cada dimensión.

2. En la segunda inicialización, cada clase se representa mediante M Gaussianas situadas en la media de las instancias de esa clase. Añadiendo variaciones aleatorias ($\frac{Rg}{500} * aleatorio(0, 1)$, para cada dimensión/característica), se crean diferentes medias, y las Gaussianas se desplazan. Así se obtiene una mezcla de Gaussianas para representar cada distribución de clase.

3. En el tercer caso, los M modos iniciales son elegidos aleatoriamente de entre los ejemplos del conjunto de entrenamiento. Esta idea está basada en la distribución aleatoria de las representaciones.

Entre estas tres inicializaciones, la segunda es la que se ha elegido como la mejor, para comparar los resultados con otros algoritmos. Así, en el algoritmo *MixtGauss* final, los modos iniciales son las M Gaussianas situadas en las medias de las clases, aleatoriamente dispersadas.

Optimización de MixtGauss

En la optimización de *MixtGauss* se estima la densidad de probabilidad independientemente para cada clase, y se representa esta densidad de probabilidad mediante los modos seleccionados. Por lo tanto, el método estándar utilizado para ajustar modelos de mezclas finitos a los datos observados es el conocido algoritmo EM [McLachlan & Basford(1988), McLachlan & Krishnan(1997)], y consiste en estimar los parámetros de las distribuciones de clase. Básicamente es un procedimiento de optimización del tipo máxima probabilidad (*Maximum Likelihood*; ML), para problemas con variables desconocidas o falta de datos [Dempster(1977)]. Así, después de la inicialización tiene lugar un proceso de optimización iterativa basado en el algoritmo EM, para poder determinar la posición óptima de las mezclas de Gaussianas independientemente para cada clase. Este proceso de optimización iterativa converge a una probabilidad máxima (*maximum likelihood*) de los parámetros de la mezcla. Así, para ajustar una mezcla de Gaussianas a cada distribución de clase, iteramos sobre los dos siguiente pasos:

Paso E Calcula las contribuciones de las instancias x^t (donde t es el número de iteración) a las clases c_j, que pertenecen al conjunto $\{x^1, \ldots, x^{N_j}\}$, donde N_j es la cardinalidad de la clase c_j. La función de densidad de probabilidad condicional para el modo m-ésima es:

$$P_m(x^t|c_j) = \frac{\alpha_{m|j} \prod_{k=1}^{d} N(x_k^t; \mu_{m|kj}, \sigma_{m|kj})}{\sum_{l=1}^{M} \alpha_{l|j} \prod_{k=1}^{d} N(x_k^t; \mu_{l|kj}, \sigma_{l|kj})} \tag{9.3}$$

Esta función está normalizada de modo que $\sum_{m=1}^{M} \alpha_{m|j} = 1$. Las Gaussianas multivariantes son representadas como un producto de distribuciones normales multivariantes, con las medias $\mu_{m|kj}$ y las desviaciones típicas $\sigma_{m|kj}$.

Paso M Calcula los parámetros del modo m-ésima para cada valor x^t que existe en la clase c_j:

$$\alpha_{m|j} = \frac{1}{N_j} \sum_{t=1}^{N_j} P_m(x^t|c_j) \qquad \mu_{m|kj} = \frac{\sum_{t=1}^{N_j} P_m(x^t|c_j) x_k^t}{\sum_{t=1}^{N_j} P_m(x^t|c_j)} \tag{9.4}$$

$$\sigma_{m|kj} = \frac{\sum_{t=1}^{N_j} P_m(x^t|c_j)(x_k^t - \mu_{m|kj})^2}{\sum_{t=1}^{N_j} P_m(x^t|c_j)} \tag{9.5}$$

Ambos pasos son iterativamente repetidos para cada clase. Los prototipos finales del conjunto condensado buscado son las medias de las distribuciones Gaussianas determinadas por el algoritmo EM al final del bucle en el que el proceso se detiene.

Control de Velocidad de la Optimización de MixtGauss

En relación con el control de velocidad, se han probado dos implementaciones diferentes.

1. La primera de ellas no tiene en cuenta las diferencias en la *velocidad de optimización* de cada clase. Así el bucle de optimización se repite hasta que se satisface el criterio de parada.

2. En el segundo caso, como nos hemos dado cuenta de que la optimización de cada clase alcanza una velocidad diferente, hemos intentado controlarla, de modo que obligamos a las que alcanzan una velocidad mayor a esperar al resto.

La segunda versión es la que obtiene mejores resultados, y se explica con más detalle a continuación.

Después de cada iteración del proceso de optimización descrito anteriormente, los modos de cada clase son modificados. Alcanzado este punto, sus mejoras en cuanto a la consistencia son comprobadas para saber si alguna de las clases aproxima sus modos con una velocidad mayor al resto. La idea principal consiste en que si los modos de una clase convergen más rápidamente que los modos de una clase vecina, los primeros pueden acaparar las muestras de los bordes. Mientras que dichas muestras pueden pertenecer a las otras clases que todavía se están modificando para acercarse a su mínimo local.

Por tanto, esta descompensación en la velocidad se detecta comprobando la precisión de clasificación de cada clase en este momento del proceso de optimización. Para medir dicha exactitud, se siguen los siguientes pasos:

1. En primer lugar, cada modo del estado actual se toma como un representante del nuevo conjunto reducido.

2. En segundo lugar, se estiman las etiquetas de las instancias del conjunto de entrenamiento original, y se comparan con sus etiquetas verdaderas.

3. Finalmente, se calcula la precisión de clasificación, mediante el uso del ratio entre el número de clasificaciones correctas en cada clase, y el número total de ejemplos en la misma clase. Si este porcentaje empeora para una clase en comparación con la iteración anterior, dicha clase es obligada a esperar a la próxima repetición del bucle para modificar sus modos.

El bucle EM se repite para cada clase, y se comprueba la precisión de clasificación hasta que se alcanza el criterio de parada.

Criterio de Parada de MixtGauss

La optimización iterativa del proceso descrito anteriormente puede provocar una superposición (overlap) inadecuada entre las componentes de clases diferentes, deteriorando por tanto, la precisión de la clasificación final. Así, este proceso debe parar en un momento determinado. Se presentan a continuación tres alternativas.

1. En la primera versión, limitamos las iteraciones después de una cantidad dada de repeticiones, tal y como se hace en el algoritmo LVQ [Kohonen(1996)].

2. En la segunda versión, mediante la aplicación del criterio de *quasi-consistencia* presentado con anterioridad, el bucle de optimización de *MixtGauss* se repite mientras la precisión de clasificación aumenta para alguna clase. Cuando ninguna clase mejora sus resultados, el proceso se detiene. El criterio de parada funciona como se indica a continuación. Después de modificar los modos de todas las clases, se comprueba su *quasi-consistencia*. Para hacer esto, se toma el ratio de precisión calculado del modo que se explica en el apartado *Control de Velocidad de Optimización de MixtGauss*, mediante el uso de la regla 1-NN. Los resultados de estos cálculos se comparan con los de la iteración previa. El proceso se detiene cuando ninguna clase alcanza una mejora en su comportamiento, tratando de obtener el mínimo error de clasificación. Mientras tanto, el movimiento de las Gaussianas continúa hasta que se encuentra una posición de equilibrio.

3. Se propone una tercera versión, en la que se utiliza el clasificador de Bayes como estimador para el criterio de parada, en lugar de la regla 1-NN. Este caso recibe el nombre de *MGBayes*.

Finalmente, la segunda versión del criterio de parada, utilizando el criterio de quasi-consistencia basado en la regla 1-NN, es adoptado por el algoritmo *MixtGauss* final, dado que los resultados son mejores en general (aunque bastante similares a los de la versión del criterio de consistencia basado en el clasificador de Bayes). La técnica completa se resume en el Algoritmo 9.3 que se presenta más adelante.

Algoritmo MixtGauss

Aquí se presenta el algoritmo *MixtGauss* (Algorithm 9.3), basado en el proceso descrito en secciones anteriores. Aquí recordamos no obstante, la opción elegida para cada paso, de modo que el algoritmo quede suficientemente claro.

1. En primer lugar, en la inicialización se colocan M modos en la posición media de cada clase, y se separan entre ellos mediante la suma de cantidades aleatoriamente obtenidas.

2. En segundo lugar, se aplica la optimización EM a cada clase independientemente; véanse la Fórmula 9.3 y la Fórmula 9.4.

3. Cuando una iteración del bucle EM finaliza para todas las clases, se compara el error con el calculado en la iteración anterior para averiguar si unas clases

Algorithm 9.3: Pseudocódigo del algoritmo *MixtGauss*.

```
(* inicialización *)
for $c = 1..number\_of\_classes$ do
   $mean[c] = CalculateMean(c, TS)$
   for $g = 1..number\_of\_Gaussians\_per\_class$ do
      $Gaussians[c, g] = mean[c] + RandomDisturbance()$
   end for
end for
$current\_accuracy = CalculateAccuracy()$
(* optimización *)
repeat
   $previous\_Gaussians[c, g] = Gaussians[c, g]$
   $Gaussians[c, g] = EMstep()$
   $previous\_accuracy = current\_accuracy$
   $current\_accuracy = CalculateAccuracy()$
   $classes\_improve = 0$
   for $c = 1..number\_of\_classes$ do
      if $current\_accuracy[c] > previous\_accuracy[c]$ then
         $classes\_improve = classes\_improve + 1$
      else
         if $current\_accuracy[c] < previous\_accuracy[c]$ then
            for $g = 1..number\_of\_Gaussians\_per\_class$ do
               $Gaussians[c, g] = previous\_Gaussians[c, g]$
            end for
         end if
      end if
   end for
until $classes\_improve == 0$
```

se mueven más rápidamente que otras. Si la precisión ha disminuido para alguna de las clases, ésta o éstas son forzadas a esperar hasta que el resto haya realizado una iteración más del bucle EM.

4. Si el ratio de la bonanza de la clasificación no mejora para ninguna clase, es porque se ha alcanzado la posición de equilibrio y el criterio de parada hace que el proceso se detenga.

El proceso introducido puede ser observado como un esquema de condensado adaptativo, en el cual los prototipos del conjunto resultante se corresponden con los centros de las Gaussianas finales. Algorítmicamente, puede ser representado como se muestra en el Algoritmo 9.3.

Sobre la Convergencia de MixtGauss

Es un hecho reconocido que la aplicación del algoritmo EM sobre una mezcla de Gaussianas que representa una única clase tiene un comportamiento convergente. Basándonos en este hecho, vamos a mostrar que el algoritmo *MixtGauss* también es convergente.

Sea una base de datos con J clases diferentes. Entonces, un proceso EM independiente es ejecutado para cada una de ellas, en el cual los modos de cada clase se mueven independientemente, dado que las iteraciones EM tienen lugar de un modo completamente separado en el tiempo y utilizan información diferente para cada clase. Al final de cada iteración del algoritmo EM para cada una de las clases, una comprobación externa se encarga de calcular las mejoras de la precisión de clasificación para cada clase. Dependiendo de los resultados de dicha comprobación externa, el estado del proceso EM para una clase puede ser detenido (*stand by*) o no. La detención consiste en forzar la espera de una clase, manteniendo su posición y variables, mientras el resto de procesos EM de otras clases continúa iterando.

Por tanto, como la lista de estados de una clase, al igual que el procedimiento EM de cada clase, no cambia en ningún aspecto y actúa independientemente de su comportamiento para el resto de las clases, entonces la convergencia de la técnica EM sobre todas las clases está garantizada por la convergencia del algoritmo EM mismo. Por tanto, *MixtGauss* presenta un comportamiento convergente.

Cuando ninguna de las clases mejora su precisión de clasificación, eso significa que se ha alcanzado la situación de equilibrio. Esto puede suceder por dos razones: una de ellas es que el proceso EM para cada clase finalmente haya convergido; la otra es que se alcance un punto de equilibrio consistente entre las clases. Independientemente de cual de los dos casos se ha satisfecho para encontrar la posición de equilibrio, el proceso se detiene, dado que cada modo está situado en una posición que optimiza la representación de las clases.

9.3 Aportaciones

El objetivo principal de esta tesis es, por un lado, el análisis de un conjunto de técnicas utilizadas en reconocimiento de formas: clasificación y búsqueda del vecino más cercano, edición y, especialmente, condensado. Este estudio ha sido detallado en la Parte I. Por otro lado, algunos métodos alternativos para condensado se han propuesto en la Parte II, utilizando los conceptos de vencindad envolvente y agrupamientos de tipo Gaussiano. Además se comparan algunas técnicas de clasificación y búsqueda, y se comentan en la Parte III.

Finalmente, es importante destacar que cada uno de los algoritmos propuestos en esta tesis ha sido evaluado empíricamente mediante el uso de problemas reales, representados en espacios de características, comparando sus resultados con los obtenidos para los algoritmos estudiados en la Parte I. Este análisis permite darse cuenta del buen comportamiento de los algoritmos presentados en esta tesis, y de que los mismos representan un buen equilibrio entre los ratios de precisión de clasificación y reducción de tamaño.

Los principales apartados dedicados a contribuciones son dos. El primero de ellos se refiere a aquellos algoritmos no adaptativos. El segundo en cambio está dedicado a la introducción de nuevos esquemas adaptativos. Las contribuciones particulares de estos apartados se detallan en las siguientes secciones.

9.3.1 Contribuciones No Adaptativas

En primer lugar se han presentado una serie de algoritmos de condensado no adaptativos. La idea principal que se introduce es la del uso del concepto de vecindad envolvente para cubrir las diferente áreas y reducir el tamaño del conjunto de entrenamiento lo máximo posible, sin una disminución importante en cuanto a lo que a precisión de clasificación se refiere. La regla NCN había sido utilizada hasta el momento para clasificar objetos sin etiquetar. En los algoritmos que aquí se presentan, por el contrario, la regla NCN ha sido utilizada para seleccionar (y reemplazar) algunas instancias representativas del conjunto de entrenamiento, de modo que los nuevos prototipos cubran una mayor superficie, que si se hubiese utilizado la regla NN.

También se presentan los resultados obtenidos con estos algoritmos no adaptativos, comparándolos en términos de porcentaje de reducción y precisión de clasificación, con los ya existentes. Teniendo en cuenta estos resultados, podemos destacar dos de los algoritmos introducidos en esta tesis, como los que obtienen mejores resultados: *MaxNCN* (el mejor) y *Reconsistent* (el tercero, siendo segundo el condensado de Hart). Estos son los algoritmos que obtienen los mejores resultados dado el equilibrio que representan entre reducción de tamaño y precisión de clasificación, además de ocupar la posición más cercana al punto ideal (100,100).

9.3.2 Contribuciones Adaptativas

En cuanto a las contribuciones adaptativas, el objetivo es el de mejorar nuestros propios resultados mediante el posicionamiento de prototipos en el punto exacto que haga el algoritmo de aprendizaje lo más preciso posible. Por una parte, algunos de estos algoritmos se basan en la idea de vecindario envolvente. Sin embargo, los resultados para estos esquemas adaptativos no ofrecen importantes mejoras (en comparación con los algoritmos no adaptativos basados en vecindarios envolventes), además de aumentar el esfuerzo computacional.

Por otro lado, la mayor contribución adaptativa es un algoritmo que utiliza principalmente mezclas de Gaussianas. Se han comparado diferentes inicializaciones y criterios de parada. Entre todos ellos, la mejor combinación, según los resultados obtenidos, y el esfuerzo computacional necesario es la del esquema *MixtGauss*. Los resultados obtenidos con el uso de este algoritmo han sido evaluados mediante clasificadores como la regla NN, construida en un espacio de características, y clasificadores lineales (FLD; Fisher Linear Discriminant) y cuadráticos (NQC; Quadratic Normal density based Classifier) construidos en espacios de disimilaridad. Estos resultados son comparados con los de los algoritmos LVQ, Chen, *MaxNCN* y *Reconsistent*. Por una parte, de los experimentos llevados a cabo parece que *MixtGauss* obtiene buenos resultados, por ejemplo mayor porcentaje de precisión que los obtenidos por otros algoritmos adaptativos como el de Chen, manteniendo el mismo ratio de reducción de tamaño. Además, es destacable que las diferencias entre *MixtGauss* y Chen son más significativas a medida que se aplica una reducción mayor. Por otro lado, estos resultados muestran que *MixtGauss* y LVQ tienen un comportamiento similar, y mejor que *MaxNCN* y *Reconsistent*. Además, ambos esquemas adaptativos permiten que sea el usuario quien controle el número de prototipos, y representan un buen equilibrio entre precisión de clasificación y reducción de tamaño. Se ha observado una tendencia general con relación a que algoritmo, *MixtGauss* o LVQ obtiene mejores resultados con una base de datos concreta. En primer lugar, mientras que LVQ parece ser la mejor opción para bases de datos con dos clases, *MixtGauss* tiende a obtener mejores resultados con bases de datos con más de dos clases. En segundo lugar, mientras LVQ en general obtiene mejores resultados para bases de datos con una dimensionalidad menor, *MixtGauss* representa mejores resultados para esos con una dimensionalidad mayor, a pesar de que hay excepciones en ambos casos.

9.3.3 Comparación de Reglas de Clasificación

Adicionalmente, se comparan los clasificadores como la regla NN, construida en un espacio de características, y los clasificadores FLD y NQC, construidos en espacios de disimilaridad. En general, los resultados más equilibrados son los obtenidos por FLD. Los resultados obtenidos por NQC son similares. Esto muestra que un clasificador con mayor complejidad que la lineal no tiene, necesariamente, que mostrar un comportamiento mejor en espacios de disimilaridad obtenidos mediante datos vectoriales normalizados.

Finalmente, los resultados obtenidos con el uso de la regla 1-NN son significativamente peores en la mayor parte de las bases de datos. Por tanto, estos resultados muestran que se pueden utilizar otras reglas de clasificación, diferentes a 1-NN, que además arrojan mejores resultados, independientemente de los algoritmos de condensado utilizados. Especialmente, los clasificadores FLD y NQC construidos en espacios de disimilaridad, son aconsejables para problemas con un número pequeño de clases.

9.3.4 Comparación con Técnicas Eficientes de Búsqueda del Vecino Más Cercano

Las bases de algunos algoritmos eficientes de búsqueda del vecino más cercano han sido introducidas, y sus tiempos y precisiones han sido analizadas. La comparación ha consistido básicamente en aplicar de dos modos diferentes la regla 1-NN para la clasificación de objetos: búsqueda exhaustiva sobre conjuntos reducidos, versus búsqueda efectiva sobre conjuntos de entrenamiento originales.

Las diferencias en cuanto a tiempo entre la búsqueda efectiva y las técnicas de condensado no son de importancia, dado que la mayor diferencia observada es inferior a un minuto. Teniendo esto en cuenta, podemos concluir que la menor cantidad de tiempo es requerida cuando se utiliza *MaxNCN*, *Reconsistent* o *MixtGauss*. Por tanto, al menos con una base de datos pequeña como las utilizadas en los experimentos, los algoritmos introducidos en esta tesis procuran un funcionamiento más rápido que las búsquedas efectivas aquí comparadas.

En relación con la precisión, el algoritmo *MixtGauss* obtiene una precisión mayor que cualquiera de las técnicas de búsqueda eficiente que aquí se comparan (siendo al mismo tiempo, más rápido). *Reconsistent* y *MaxNCN* obtienen una precisión ligeramente inferior. En consecuencia, *MixtGauss* es el algoritmo que, con menores requerimientos de memoria y tiempo, alcanza la mayor precisión.

En relación con los algoritmos eficientes, kvp-tree muestra el mejor comportamiento entre los que aquí se han comparado, dado que obtiene una precisión más alta que el resto, y requiere para su ejecución de una cantidad de tiempo similar.

Para concluir, podemos afirmar que ambas metodologías, reducción de tamaño del conjunto de entrenamiento, y búsqueda eficiente, obtienen buenos resultados en un buen tiempo.

9.4 Conclusiones

El objetivo de este apartado es el de enumerar las principales conclusiones de la investigación aquí referida. Brevemente se indican a continuación.

1. **Vecindario Envolvente**. Se presentan algunos algoritmos, (selectivos y generativos) basados en el uso de la regla NCN. Entre todos ellos, podemos destacar dos, por sus buenos resultados en comparación con otros: *MaxNCN* y *Reconsistent*. Véanse los Capítulos 3, 4 y 5.

2. **Mezclas de Gaussianas**. Se ha presentado un algoritmo generativo basado en el uso de mezclas de Gaussianas: *MixtGauss*. De los experimentos llevados a cabo se extrae que *MixtGauss* obtiene buenos resultados (un equilibrio adecuado entre los porcentajes de precisión de clasificación y reducción de tamaño), mejorando incluso los resultados obtenidos por *MaxNCN* y *Reconsistent*. Adicionalmente, permite al usuario controlar el número de prototipos. Véanse los Capítulos 4 y 6.

3. **Clasificadores**. Los resultados para LVQ, Chen, *MaxNCN*, *Reconsistent* y *MixtGauss* (Capítulos 2, 3 y 4) han sido evaluados por clasificadores como la regla NN, en un espacio de características, y FLD y NQC, en espacios de disimilaridades. De este estudio, (véase el Capítulo 6) podemos extraer que se pueden utilizar otras reglas de clasificación, y que de hecho obtienen mejores resultados que la regla 1-NN. Especialmente los clasificadores FLD y NQC construidos en espacios de disimilaridad, son aconsejables para problemas con un número de clases pequeño.

4. **Búsqueda Eficiente del Vecino Más Cercano**. Se compara el uso de la búsqueda exhaustiva sobre conjuntos reducidos, con la búsqueda eficiente del vecino más cercano sobre conjuntos de entrenamiento originales. Los algoritmos *MaxNCN*, *Reconsistent* y *MixtGauss* son los que se ejecutan en el menor tiempo, del mismo modo que *MixtGauss* alcanza la mayor precisión. Así, destacamos este algoritmo como el mejor. Merece la pena resaltar que kvp-tree muestra el mejor comportamiento de entre todas las búsquedas eficientes aquí contrastadas. En realidad, ambas metodologías obtienen buenos resultados en un buen tiempo.

9.5 Futuras Líneas de Investigación

La primera idea sobre posibles mejoras está relacionada con la reducción adaptativa, concretamente con mezclas de Gaussianas. En el trabajo que aquí se presenta, las Gaussianas son ortogonales (es decir, sus diagonales principal y secundaria son paralelas a los ejes X e Y. Dado que en este caso las bases de datos están obligadas a definir su distribución alrededor de líneas paralelas a los ejes X o Y, pensamos que el algoritmo MixtGauss mejoraría sus resultados si se permitiese cualquier orientación a las Gaussianas encargadas de representar las instancias originales mediante sus medias. Esto representaría un mayor esfuerzo en cuanto a cálculos matriciales, que creemos sería abordable.

Además, será muy conveniente estudiar la relación entre el número de clases y la dimensionalidad de una base de datos, y el mejor algoritmo a utilizar, LVQ o *MixtGauss*. Este estudio podría realizarse sobre diferentes bases de datos sintéticas, dado que con ellas se permite controlar el número de clases y la cantidad de dimensiones.

Con relación al campo de la disimilaridad, parece muy interesante la aplicación de utilidades del espacio de representación, igualmente que la aplicación de utilidades del campo de la disimilaridad en el espacio de representación. Parece un mundo a estudiar muy interesante todavía nuevo para nosotros en diferentes aspectos.

En el Capítulo 4 utilizamos una matriz de covarianza local en el algoritmo *MixtGauss* para encontrar prototipos. Después de eso, la regla NN se ha utilizado en distancias Euclídeas, dejando de lado la correlación local entre los datos. Una aproximación más consistente se basaría en reemplazar la distancia Euclídea por la distancia de Mahalanobis, basada en la propia matriz de covarianza que se utiliza para encontrar los prototipos a considerar. Sin embargo, somos conscientes de que esto aumentará la complejidad computacional.

Otro punto para la investigación futura puede basarse en comparar una SVM (no) lineal construida en el espacio vectorial de las características originales y la SVM en un espacio de disimilaridad definido por un pequeño conjunto de prototipos optimizados, dado que en ese caso la SVM se basaría en vectores de soporte que son objetos encontrados en un espacio de disimilaridad completo, y por tanto se basan en las distancias a *todos* esos representantes optimizados.

En cuanto a la búsqueda eficiente, creemos que una buena idea podría ser unir el uso de algoritmos de reducción junto con técnicas eficientes de búsqueda, dado que ambos enfoques funcionan bien. Esta idea no ha sido utilizada en las bases de datos que aquí se usan, porque debería probarse sobre bases de datos mayores. Pensamos que el uso de técnicas de búsqueda eficiente sobre conjuntos reducidos tendrá como consecuencia una disminución de los requisitos tanto de memoria como de tiempo, sin implicar una gran reducción de precisión de clasificación.

Bibliography

[Aha et al.(1991)] D.W. Aha and D. Kibler and M.K.Albert, Instance-based Learning Algorithms, *Machine Learning* **6**(1), (1991) 37–66

[Aha(1992)] D.W. Aha, Tolerating Noisy, Irrelevant and Novel Attributes in Instance-based Learning Algorithms, *International Journal of Man-Machine Studies* **36**, (1992) 267–287

[Brin(1995)] S. Brin, Near Neighbour Search in Large Metric Spaces, *Proceedings of the 21st VLDB Conference* (1995) 574–584

[Barandela & Gasca(2000)] R. Barandela and E. Gasca, Decontamination of Training Data for Supervised Pattern Recognition Methods. *Advances in Pattern Recognition, Lecture Notes in Computer Science* **1876**, Springer-Verlag (2000) 621–620.

[Bendtley(1975)] J.L. Bentley, Multidimensional Binary Search Trees Used for Associative Searching. *ACM* **18** (1975) 509–517

[Cameron-Jones(1995)] R.M. Cameron-Jones, Instance Selection by Encoding Lenghth Heuristic with Random Mutation Hill Climbing, *Proceedings of the Eighth Australian Joint Conference on Artificial Intelligence*, (1995) 99–106

[Chang(1974)] C.L. Chang, Finding Prototypes for Nearest Neighbor Classifiers, *IEEE Trans. on Computers*, **23** (1974) 1179–1184

[Charnes(1978)] A. Charnes, W. Cooper and E. Rhodes, Measuring the Efficiency of Decision Making Units, *European Journal of Operational Research*, **2** (1978) 429–444

[Chaudhuri(1996)] B.B. Chaudhuri, A New Definition of Neighbourhood of a Point in Multi-Dimensional Space, *Pattern Recognition Letters*, **17** (1996) 11–17

[Chavez et al.(2001)] E. Chavez, G. Navarro, R.A. Baeza-Yates and J.L. Marroquin, Searching in Metric Spaces, *ACM Computing Surveys*, **33**, 3 (2001) 273–321

[Chen & Jozwik(1996)] C.H. Chen and A. Józwik, A Sample Set Condensation Algorithm for the Class Sensitive Artificial Neural Network, *Pattern Recognition Letters*, **17** (1996) 819–823

[Cover & Hart(1967)] T.M. Cover and P.E. Hart, Nearest Neighbor Pattern Classification, *IEEE Trans. on Information Theory* **13** no. 1, (1967) 21–27

[Cristianini & Shawe-Taylor(2000)] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines* (Cambridge University Press, 2000)

[Dasarathy(1991)] B.V. Dasarathy, Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques, *IEEE Computer Society Press* (Los Alamitos, CA, 1991)

[Dasarathy(1994)] B.V. Dasarathy, Minimal Consistent Subset (MCS) Identification for Optimal Nearest Neighbor Decision Systems Design, *IEEE Trans. on Systems, Man and Cybernetics* **24** (1994) 511–517

[Dempster(1977)] A.P. Dempster, N.M. Laird and D.B. Rubin, Maximum Likelihood from Incomplete Data Via the EM Algorithm, *Journal of the Royal Statistical Society (B).* **39** (1977) 1–38

[Devijver(1982)] P.A. Devijver and J. Kittler, *Pattern Recognition: a Statistical Approach*, (Prentice Hall, Englewood Cliffs, N.J., 1982)

[Devroye(1996)] L. Devroye, L. Györfi and G. Lugosi, *A Probabilistic Theory of Pattern Recognition* (Springer-Verlag, New York, NY, 1996)

[Duda et al.(2001)] R.O. Duda, P.E. Hart and D.G. Stork, *Pattern Classification, 2nd ed.* (John Wiley & Sons, Inc., 2001)

[Duda & Hart(1973)] R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis* (John Wiley & Sons, NY, 1973)

[Duin et al.(2004)] R.P.W. Duin, P.Juszczak, D. de Ridder, P. Paclík, E. Pękalska and D.M.J. Tax, *PR-Tools, a Matlab Toolbox for Pattern Recognition* (2004). http://www.prtools.org

[Figueiredo & Jain(2002)] M.A.T. Figueiredo and A.K. Jain, Unsupervised Learning of Finite Mixture Models, *IEEE Trans. on Pattern Analysis Machine Intelligence* **24** (2002) 381–396

[Friedman(1977)] J.H. Friedman, J.L. Bentley and R.A. Finkel, An Algorithm for Finding Best Matches in Logarithmic Expected Time, *ACM Transactions on Mathematical Software* **3** (1977) 209–226

[Fu(1982)] K.S. Fu, *Syntactic Pattern Recognition and Applications* (Prentice-Hall, Englewood Cliffs, NJ, 1982)

[Fukunaga & Narendra(1975)] K. Fukunaga and M.Narendra, A Branch and Bound Algorithm for Computing k-Nearest Neighbors, *IEEE Transactions on Computing* **24** (1975) 750–753

[Fukunaga(1990)] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press (1990)

[Hart(1968)] P.E. Hart, The Condensed Nearest Neighbor Rule, *IEEE Trans. on Information Theory* **14** no. 5 (1968) 505–516

[Hastie & Tibshirani(1996)] T. Hastie and R. Tibshirani, Discriminant Analysis by Gaussian Mixtures, *Journal of the Royal Statistical Society (B)* **58** (1996) 155–176

[Hattori & Takahashi(2000)] K. Hattori and M. Takahashi, A New Edited k-Nearest Neighbor Rule in the Pattern Classification Problem. *Pattern Recognition* **33** (2000) 521–528.

[Hinton et al.(1997)] G. Hinton and P. Dayan and M. Revow, Modeling the Manifolds of Images of Handwritten Digits, *IEEE Transactions on Neural Networks* **8** (1997) 65–74

[Keung & Lam(2000)] C.-K. Keung and W. Lam, Prototype Generation Based on Instance Filtering and Averaging. *Proceedings of the 4th. Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Springer-Verlag (2000) 142–152.

[Kohonen(1995)] T. Kohonen, *Self-Organizing Maps* (Springer-Verlag, 1995)

[Kohonen(1996)] T. Kohonen, J. Hynninen, J. Kangas, J. Laarksonen and K. Torkkola, *LVQ_PAK: The Learning Vector Quantization Program Package*, Helsinki University of Technology (1996) http://www.cis.hut.fi/research/som-research/nnrc-programs.shtml

[Koplowitz & Brown(1981)] J. Koplowitz and T.A. Brown, On the Relation of Performance to Editing in Nearest Neighbor Rules. *Pattern Recognition Letters* **13** (1981) 251–255.

[Kuncheva(1995)] L.I. Kuncheva, Editing for the k-Nearest Neighbors Rule by a Genetic Algorithm. *Pattern Recognition Letters* **16** (1995) 809–814.

[Lozano(2004)A] M. Lozano, J.S. Sánchez, F. Pla, Using the Geometrical Distribution of Prototypes for Training Set Condensing, *Current Topics in Artificial Intelligence, Lecture Notes in Artificial Intelligence*, Vol. 3040, R. Conejo, M. Urretavizcaya, J.L. Pérez de la Cruz (Eds.), Springer-Verlag, (2004), 618-627.

[Lozano(2004)B] M. Lozano, J.M. Sotoca, J.S. Sánchez, F. Pla, An adaptive Condensing Algorithm Based on Mixtures of Gaussians, *Setè Congrés Català d'Intel·ligència Artificial, Frontiers in Artificial Intelligence and Applications*, Vol. 113, IOS Press, J. Vitrià et al. (Eds.), (2004), 225-232, Barcelona, Spain.

[McLachlan & Basford(1988)] G. McLachlan and K. Basford, *Mixture Models: Inference and Application to Clustering* (Marcel Dekker, New York, 1988)

[McLachlan & Krishnan(1997)] G. McLachlan and T. Krishnan, *The EM Algorithm and Extensions* (John Wiley & Sons, New York, 1997)

[Merz & Murphy(1998)] C.J. Merz and P.M. Murphy, *UCI Repository of Machine Learning Databases*, Deptartment of Information and Computational Science, University of California-Irvine (1998). http://www.ics.uci.edu/˜mlearn/MLRepository.html.

[Micó et al.(1994)] L. Micó, J. Oncina and R.C. Carrasco, A New Version of the Nearest Neighbour Approximating and Eliminating Search Algorithm (AESA) with Linear Preprocessing-Time and Memory Requirements, *Pattern Recognition Letters* **15** (1994) 9–17

[Micó et al.(1996)] L. Micó, J. Oncina and R.C. Carrasco, A Fast Branch & Bound Nearest Neighbour Classifier in Metric Spaces, *Pattern Recognition Letters* **17** no. 3-4 (1996) 731–739

[Mollineda et al.(2002)] R.A. Mollineda, F.J. Ferri and E. Vidal, An Efficient Prototype Merging Strategy for the Condensed 1-NN rule through Class-conditional Hierarchical Clustering, *Pattern Recognition* **35** (2002) 2771–2782

[Moreno(2004)] F. Moreno Seco, *Classificadores Eficaces Basados en Algoritmos Rapidos de Busqueda del Vecino Mas Cercano*, PhD thesis, Dept. de Lenguajes y Sistemas Informaticos, Universidad de Alicante (Spain, February 2004).

[Novovicova et al.(1996)] J. Novovičová and P. Pudil and J. Kittler, Divergence Based Feature Selection for Multimodal Class Densities, *IEEE Trans. on Pattern Analysis and Machine Intelligence.* **18** (1996) 218–223

[Paclik & Duin(2003)A] P. Paclík and R.P.W. Duin, Dissimilarity-based Classification of Spectra: Computational Issues, *Real Time Imaging* **9** no. 4 (2003) 237–244

[Paclik & Duin(2003)B] P. Paclík and R.P.W. Duin, Classifying Spectral Data using Relational Representation, *Procedures of the Spectral Imaging Workshop* (Graz, Austria, 2003)

[Pavlidis(1977)] T. Pavlidis, *Estructural Pattern Recognition* (Springer-Verlag, Berlin, 1977)

[Pękalska & Duin(2002)] E. Pękalska and R.P.W. Duin, Dissimilarity Representations Allow for Building Good Classifiers, *Pattern Recognition Letters* **23** no. 8 (2002) 943–956

[Pękalska et al.(2002)] E. Pękalska, P. Paclík and R.P.W. Duin, A Generalized Kernel Approach to Dissimilarity Based Classification, *Journal of Machine Learning Research*, **2** no. 2 (2002) 175–211

[Pękalska & Duin(2004)] E. Pękalska, R.P.W. Duin, S. Günter and H. Bunke, On Not Making Dissimilarities Euclidean, *Proc. of Structural Syntactic and Statistical Pattern Recognition, Lecture Notes in Computer Science* **3138** (Springer Verlag, Berlin, 2004) 1145–1154

[Pękalska(2005)] E. Pękalska, *Dissimilarity Representations in Pattern Recognition. Concepts, Theory and Applications*, PhD thesis, Delft University of Technology, ASCI Dissertation Series, 109 (The Netherlands, January 2005).

[Pękalska et al.(2006)] E. Pękalska, R.P.W. Duin and P. Paclík, Prototype Selection for Dissimilarity-based Classifiers, *Pattern Recognition* **2** no. 39 (2006) 189–208.

[Ramasubramanian & Paliwal(2000)] V.Ramasubramanian and K.K. Paliwal, Fast Nearest-Neighbour Search Algorithms Based on Approximation-Elimination Search, *Pattern Recognition* **33** no. 9 (2000) 1497–1510

[Sánchez(2004)] J.S. Sánchez, High Training Set Size Reduction by Space Partitioning and Prototype Abstration, *Pattern Recognition* **37** no. 7 (2004) 1561–1564

[Sánchez et al.(1997)A] J.S. Sánchez, F. Pla and F.J. Ferri, Prototype Selection for Nearest-Neighbour Rule Through Proximity Graphs, *Pattern Recognition Letters* **18** no. 6 (1997) 507–513

[Sánchez et al.(1997)B] J.S. Sánchez, F. Pla and F.J. Ferri, On the Use of Neighbourhood-based Non-Parametric Classifiers, *Pattern Recognition Letters* **18** (1997) 1179-1186

[Schölkopf et al.(2000)] B. Schölkopf, A.J. Smola, R.Williamson and P.L. Bartlet, New Support Vector Algorithms, *Neural Computation* **12** (2000) 1207–1245

[Skurichina(2001)] M. Skurichina, *Stabilizing Weak Classifiers* PhD thesis, Delft University of Technology, Delft, The Netherlands (2001)

[Smola et al.(1999)] A.J. Smola, T.T. Friess and B. Schölkopf, Semiparametric Support Vector and Linear Programming Machines, In M.M. Kearns, S.A. Solla and D.A. Cohn editors, *Advances in Neural Information Processings Systems* **11** (1999) 485–591, Cambridge, MA, MIT Press.

[Tomek(1976)] I. Tomek, *Two Modifications of CNN*, *IEEE Trans. on Systems, Man and Cybernetics* **6** (1976) 769-772

[Toussaint et al.(1985)] G.T. Toussaint and B.K. Bhattacharya and R.S. Poulsen, The Application of Voronoi Diagrams to Nonparametric Decision Rules, *Computer Science and Statistics: The Interface* (L. Billard, Elsevier Science, North-Holland, Amsterdam, 1985)

[Vidal(1986)] E. Vidal, An Algorithm for Finding the Nearest Neighbour in (Aproximately) Constant Time, *Pattern Recognition Letters* **4** (1986) 145–457

[Vidal(1994)] E. Vidal, New Formulation and Improvements of the Nearest-Neighbour Approximating and Eliminating Search Algorithm (AESA), *Pattern Recognition Letters* **15** (1994) 1–7

[Wilson(1972)] D.L. Wilson Asymptotic Properties of Nearest Neighbor Rules Using Edited Data Sets, *IEEE Trans. on Systems, Man and Cybernetics* **2** (1972) 408–421

[Wilson & Martinez(1997)] D.R. Wilson and T.R. Martinez, Improved Center Point Selection for Radial Basis Function Networks, *Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA'97)* (1997) 514–517

[Wilson & Martinez(2000)] D.R. Wilson and T.R. Martinez, Reduction Techniques for Instance-based Learning Algorithms, *Machine Learning.* **38** no. 3 (2000) 257–286

[Yianilos(1993)] P.N. Yianilos, Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces, *Proceedings of the ACM-SIAM Simposium on Discrete Algorithms* (1993) 311–321