Rijksuniversiteit Groningen Faculteit der Wiskunde en Natuurwetenschappen

Wiskunde en Informatica

Bieb

Novelty Detection for Neural Pattern Classification

Ing. J.K. Kok

supervisors:
Dr.ir. J.A.G. Nijhuis
Drs. M.H. ter Brugge

January 1998

Rijksuniversiteit Groningen
Bibliotheek Informatica / Rekencentrum
Landleven 5
Postbus 800
9700 AV Groningen

2 MAART 1998

RuG

For the truth is that I already know as much about my fate as I need to know. The day will come when I will die. So the only matter of consequence before me is what I will do with my allotted time. I can remain on shore, paralyzed with fear, or I can raise my sails and dip and soar in the breeze.

– Richard Bode, First you have to row a little boat.

Abstract

Complex forms of pattern recognition is more widely used these days. Complex recognition problems are characterized pattern classes that are hard to separate and high demands on recognition speeds. This is why neural pattern classifiers have become more important. Especially the multi–layer perceptron (MLP) is very suitable for complex classification, since this method combines high classification speeds with high accuracy.

In a safety–critical environment like in medical and industrial applications pattern classifiers, like any other system, must meet high robustness standards. In pattern recognition one aspect of robustness is the reaction of a classifier when a novelty is presented to it. A novelty is an input pattern which differs significantly from the patterns used to develop the classifier. For most statistical and neural pattern classifiers is a novelty at the input easy to detect. However, an MLP based classifier is not capable to recognize an input as novelty and will simply classify the pattern in one of the known classes and the result will be invalid. To guarantee the reliability of the classifier an extension with a novelty detection method is needed. The extension must work in such a way that the strenghts of the MLP classifier remain unaffected.

To avoid interference with the classification accuracy of the MLP is searched for a separate novelty detection system that can work in parallel with the MLP. A system that investigates the input pattern and determines the degree of novelty. Existing forms of these dedicated novelty detectors are either too slow to be used for complex systems or the optimal setting of their parameters are hard to determine

To solve these problems a new novelty detection method is developed. Using this method fast novelty detection systems can be build. The appropriate parameter settings can easily be found using measurable quantities that reflect the quality of the system.

Samenvatting

Complexe vormen van patroonherkenning zoals optische karakterherkenning vinden meer en meer toepassingen. Complexe herkenningsproblemen kenmerken zich door moeilijk separeerbare patroonklassen en hoge snelheidseisen. Onder invloed hiervan zijn neurale classificatoren een belangrijker plaats gaan innemen. Met name de multi–layer perceptron (MLP) is zeer geschikt voor complexe classificatie, omdat deze methode hoge classificatiesnelheden combineert met een hoge nauwkeurigheid.

Voor patroonherkenning in een veiligheidsgevoelige omgeving, zoals in medische en industriële toepassingen, zijn de eisen aan de robuustheid hoog. Een aspect van deze robuustheid is de reactie van een patroon herkenner op een z.g. novelty, een patroon dat sterk afwijkt van de data waarmee de herkenner is ontwikkeld. Bij het gebruik van de meeste statistische en neurale classificatoren is een novelty aan de ingang eenvoudig te detecteren. Een voor classificatie getrainde MLP is echter niet in staat aangeboden novelties als zodanig te herkennen en zal hoe dan ook het patroon classificeren. Het gegeven resultaat is dus niet valide. Om de betrouwbaarheid van een op een MLP gebaseerde patroon herkenner te waarborgen is dus een uitbreiding met een novelty–detectie methode gewenst. De uitbreiding moet van zodanige aard zijn dat de sterke kanten van op de MLP gebaseerde classificatoren behouden blijven.

Om de nauwkeurigheid van de MLP niet aan te tasten is gezocht naar een op zichzelf staand novelty detectie systeem parallel aan de MLP. Een dergelijk systeem onderzoekt het input patroon en doet een uitspraak noviteit er van. In de literatuur bekende vormen van deze systemen zijn of te traag om gebruikt te worden voor complexe problemen of de optimale instelling van systeemparameters is moeilijk te vinden.

Om een oplossing te vinden voor deze problemen is een nieuwe novelty detectie methode ontwikkeld. Met deze methode zijn snelle novelty detectie systemen te bouwen, waarvan de juiste parameterinstelling via meetbare grootheden eenvoudig is te vinden.

Acknowledgements

The author whishes to thank Jos Nijhuis and Mark ter Brugge for their supervision during the research project described in this report. Without the discussions about the subject of novelty detection and their comments on the evolving report–text this thesis would not be what it is.

Lots of thanks go to my partner Bianca, who was a great support during the accomplishment of this thesis. She was a driving force behind completing it. I thank her for that.

Contents

Used Symbols

c	number of classes, number of clusters
d	pattern space dimension, number of inputs
j	class label
C_j	class j, cluster j
$(x)_i$	the i–th element of vector x
N	number of training patterns
T	matrix of N training patterns
$P(.)$	probability
$p(.)$	probability density function
P_j	prior probability of class j
$f_j(.)$	probability density function of class j
$q_j(.)$	posterior probability density of class j

Chapter 1 Introduction

1.1 Classification

Pattern recognition techniques are used in a wide variety of applications, like speech recognition, optical character recognition, medical diagnosis and fault detection in machinery. Basically pattern recognition problems are either waveform classification or classification of geometric figures [13]. For example consider the problem of testing a patient's electrocardiogram (ECG) for abnormalities. The problem here is to discriminate normal ECG–forms of abnormal, which is a typical waveform classification problem. On the other hand the recognition of handwritten characters from a graylevel mesh is a form of classification of geometric figures.

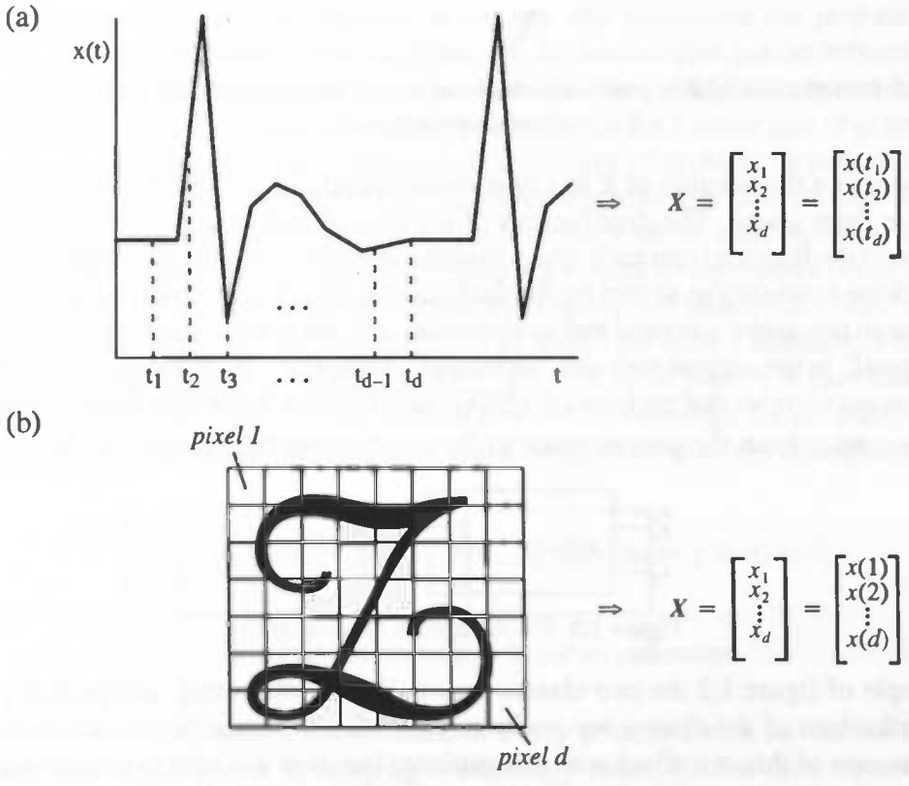

Figure 1.1: *Measuring the pattern vector* $\boldsymbol{X}$ *in (a) waveform classification and (b) classification of geometric figures.*

To perform a classification of any kind we need to measure some observable characteristics. The most primitive way is to measure the time–sample values for the waveform, $x(t_1),...,x(t_d)$, and the gray levels in the mesh for a geometric figure, $x(1),...,x(d)$, as shown by figure 1.1. These d observations can be seen as a vector in a d–dimensional pattern space. In both examples the obtained vector $\boldsymbol{X}$ is a random vector, since the outcome of the measurement will not be the same each time it is performed. Of course there must be sufficient difference between the patterns of distinctive classes to be able to carry out the classification at all, but also a spread among the patterns of one particular class is likely to occur in practical classification problems. In other words, many measurements of the pattern form a distribution of $\boldsymbol{X}$ in the d–dimensional pattern space.

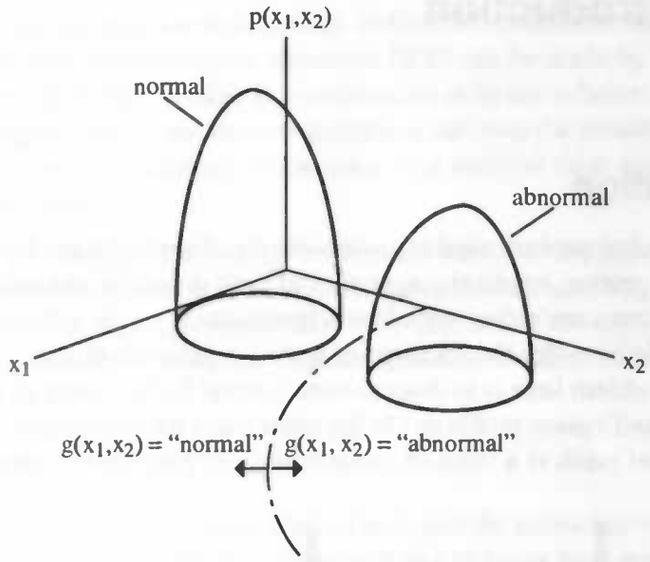

Figure 1.2: *Distribution of $\boldsymbol{X}$ in a two–dimensional two–class problem with no overlap in the two class–distributions.*

Figure 1.2 shows a distribution of $\boldsymbol{X}$ in a two–dimensional classification problem, say the ECG classification from above. The distributions of the normal and the abnormal patterns are positioned on a certain distance from each other, so somewhere between the distributions we can nicely lay a *decision boundary* as shown by the dash–dotted line. The decision boundary divides the pattern space in two areas: a normal and an abnormal area. Next we can define a function g, which results "normal" in the normal area and "abnormal" otherwise. Then g is the so called *discriminant function* or *classifier* that performs the ECG classification. Generally discriminant functions perform a mapping from the pattern space to the set of c possible classes: $g : \mathbb{R}^d \rightarrow \{1,\ldots,c\}$.

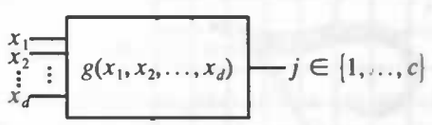

Figure 1.3: *Blockdiagram of a classifier*

In the example of figure 1.2 the two classes "normal" and "abnormal" are perfectly separable, i.e. the distributions of the classes are positioned on some distance of each other and show no overlap. Because of this, a well–chosen discriminant function g is able to classify every pattern drawn from one of the two class distributions to the right class. When two classes do show an overlap as in figure 1.4 it is impossible to define a discriminant function that classifies all patterns

correctly. For all $g : \mathbb{R}^d \rightarrow \{1,\ldots,c\}$ the probability of error is greater than zero. The *probability of error* or the *probability of misclassification* is the probability that a classifier assigns a pattern to a wrong class.

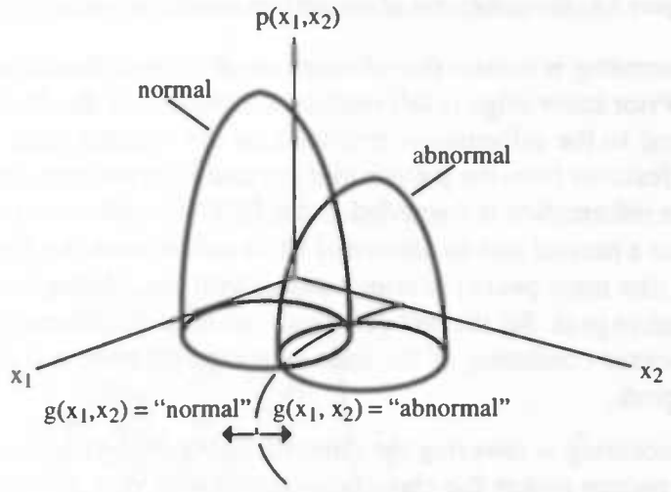

Figure 1.4: *Distribution of X in a two–dimensional two–class problem with an overlap in the two class–distributions.*

In the case of a classification problem with class–overlap the determination of a suitable discriminant function becomes an optimization process, in which the probability of error is used as a performance measure. The optimal classifier is the one that minimizes the probability of error. Theoretically this optimal discriminant function, the *Bayes classifier*, can be defined as discussed in detail in paragraph 2.1. Problem is, however, that the Bayes classifier requires the probability density function of each distinctive class to be known, while for a major part of practical classification problems these functions are unknown. For this type of problems a pattern classification method is needed that uses some kind of estimation technique to approximate the Bayes classifier or the decision boundary it generates. For this estimation a set of measurements of the pattern is needed. This set, called the *data set* or the *training set*, must be representative of the underlaying distributions to make an accurate estimation. Most estimation based classifiers have the internal structure shown by figure 1.5. The estimator produces per class a measure of the likelihood that the input pattern belongs to that particular class. The input pattern is then classified to the *winning class*, that is the class with the highest likelihood. Although the resulting class label is the classifier output, the set of likelihood measures is referred to as the *classification outputs*.

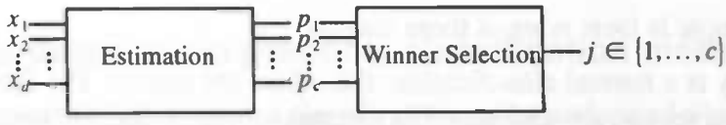

Figure 1.5: *Classification based on estimation.*

In the two examples the patterns are obtained in a very primitive way, so that the the amount of information one measurement is carrying is relatively small. Rather than represent the entire mapping from the input pattern $x_1,\ldots,x_d$ to the output j at once, it can be beneficial to split the mapping into an initial *pre–processing* stage and a classification stage, as shown by figure 1.6.

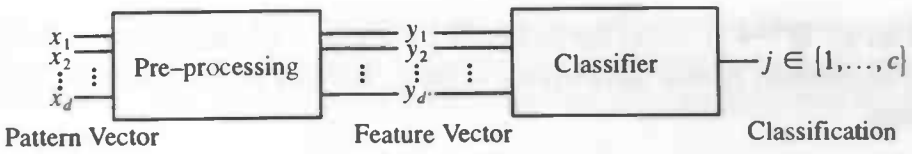

Figure 1.6: *Blockdiagram of the pattern classification process*

The aim of the pre–processing is to raise the information–density of the classifier input by applying *prior knowledge*. Prior knowledge is information we possess of the desired form of the solution which is additional to the information provided by the training data. When we use prior knowledge to extract features from the pattern that are usable to perform the classification task, the unusable part of the information is discarded. In the ECG classification problem, for example, the distinction between a normal and an abnormal ECG can be made by looking at the time between two heartbeats (the main peaks) in combination with the relative heights of a main peak and the following negative peak. So, the pre–processor can map the pattern of d samples to a two–dimensional feature vector consisting of the inter beat interval time and the ratio between the main and the second peak.

Second aim of pre–processing is lowering the dimensionality of the input pattern. High–dimensionality of the input pattern makes the classification problem very difficult and the necessary size of the training set increases exponentially with the size of the input pattern for equivalent performances. This problem is known as the *curse of dimensionality* or the *empty space phenomenon* [4], [6].

1.2 Rejection

In a lot practical applications we are not satisfied with just the answer in which class the classifier thinks a presented input pattern belongs. We would like to know how sure the classifier is of it's answer. Because then we can use another (more expensive) classification method in case of reasonable doubt. We can leave the classification to a human or a slower but more powerful method. Also we would like to know if the presented pattern is *novel*, e.g. lies outside the trained domain of the input space. We expect a well designed and tested classifier to give reliable results when the input data are similar to the training data, but when to such a network a novelty is presented, the reliability of the result may decrease significantly.

It can be concluded that a well–designed classifier should be able to give these three results on a presented example [35]:

- 'this example is from class j'
- 'this example is too hard for me
- 'this example is from none of these classes

The first category is a normal classification, the others are *rejects*. The first type of rejects are *'doubt' reports*, the second are *novelties*. The patterns subject to doubt belong to one of the known classes, but in the input space they are positioned too near to a decision boundary to make a clear decision to which class the pattern belongs. Novelties are defined as input data which differ significantly from the data used to train the network [5]. So novelties can be seen as patterns appearing in regions of the input space where the measured class densities are near to zero. So, either those patterns belong to a new class apparently unknown prior to the design process or either during training data collection it was for some reason not possible to collect data for that particular

region. The main property of novelties is that characteristics of this 'data class' are unknown. Figure 1.7 shows an example of both a novelty and a pattern subject to doubt.

To evaluate the outcome of a classification in order to accept or reject the result, some reliability measure is needed. Mostly such a reliability measure is referred to as *confidence value* which can be defined as the probability that a classification outcome is correct.

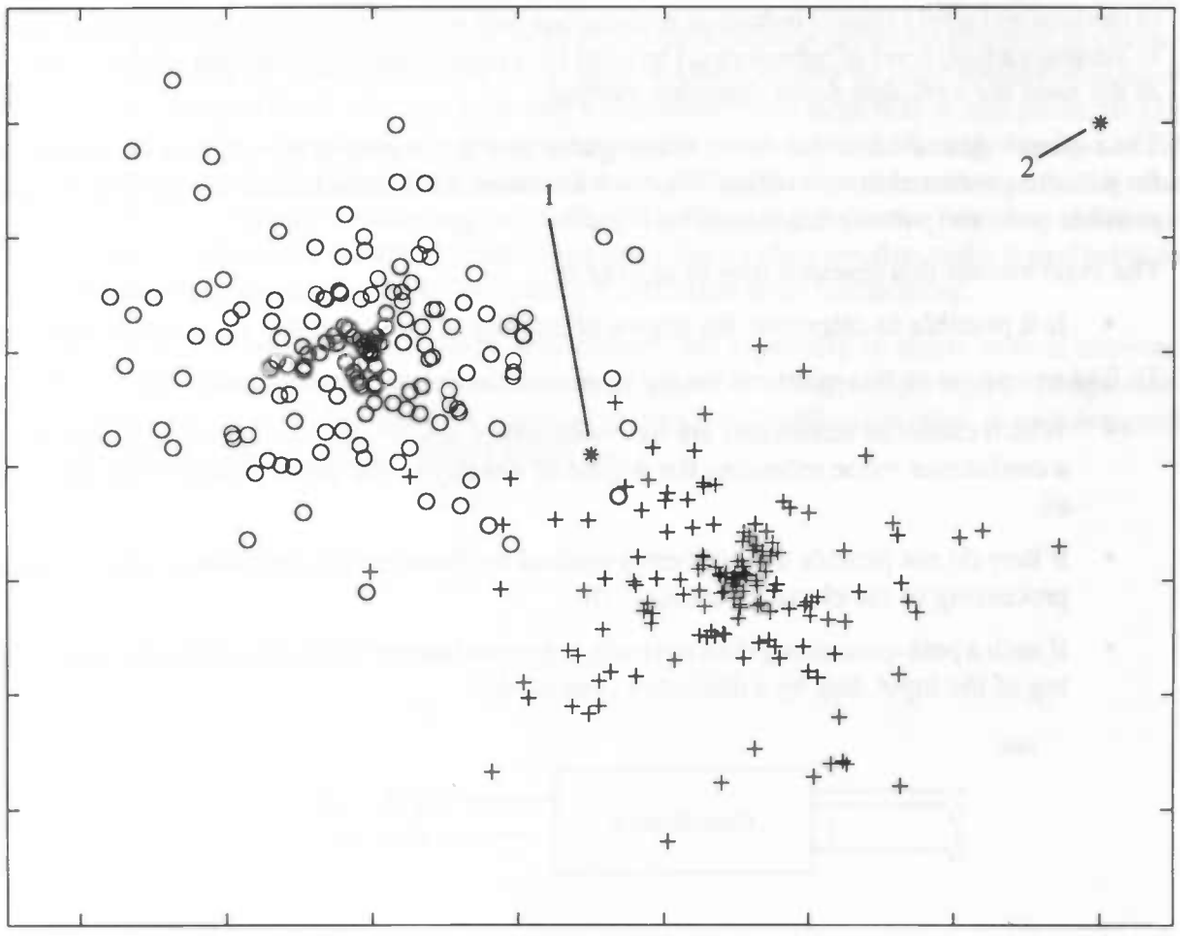

Figure 1.7: *Two examples of patterns of which the class is unclear: (1) a pattern in a doubt region and (2) a novelty.*

1.3 Problem

1.3.1 Novelty detection in practical classification methods

As described in paragraph 1.1 an optimal classification can be performed when the exact number of classes and their exact distributions are known. In that case novelties do not exist and a low confidence can be given to those patterns lying in a region with a relatively small difference in the two highest class densities. When the exact class densities are unknown, the classifier must use an estimation technique based on a training set. Depending on the estimation technique the resulting model may be tailored to perform the classification task and thus may lose it's validity for the determination of a confidence value. Therefore, without knowledge of the insights of the

used technique, the validity of the model for the purpose of rejection becomes unknown. Further is the existence of novelties ignored by classifier techniques that assume the representativity of the used training set. This implicates that such a classifier gives an undefined result when a novelty is presented to it.

Present research on confidence values for classifiers emphasizes mostly on the doubt detection part of the problem. Relative few work is done on the detection of novelties. For classification in medical and safety critical industrial applications both parts of the confidence figure are needed to maintain a high level of robustness. The need for a reliable novelty detection method is as high as the need for a reliable doubt detection method.

The research described in this report investigates how the degree of novelty can be determined for patterns presented to a classifier. What work is done, what is the lacuna in what is known and possible and can (part of) this lacuna be filled?

The main answer this research tries to answer is:

- Is it possible to determine the degree of novelty of input patterns of a pattern classifier?

To find an answer to this question, we try to answer the following sub–questions:

- Which classifier techniques are by construction able to provide their classification with a confidence value reflecting the degree of novelty of the input pattern? (See figure 1.8 a).
- If they do not provide this, is there a method to obtain such a confidence value via post–processing of the classifier outputs? (b).
- If such a post–processing does not exist, is the confidence value obtainable by the processing of the input data by a dedicated system? (c).

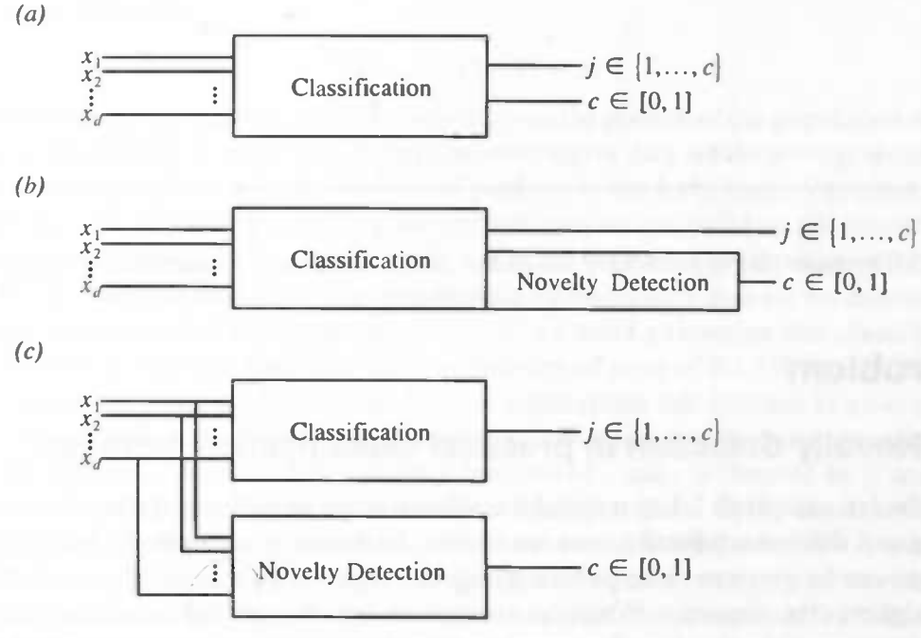

Figure 1.8: *Three structures of classification with confidence assessment: confidence assessment as part of the classification task, (b) confidence assessment as post–processing and (c) dedicated novelty detector in parallel with the classification task*

The emphasis of the research is on the novelty part of the confidence problem, but the problem of doubt detection is not ignored. Although both problems have a different background, the possibility of a method yielding one confidence value reflecting both doubt and novelty must not be closed out at forehand.

1.3.2 Framework

The emphasis of this report is on complex real–world classification problems like optical character recognition (OCR). These problems are in general characterized by large dimensional input spaces. A property of OCR, which is generally a characteristic of large real–world problems [3], is that the classes are approximately (linearly) separable up to a certain limit (generally around a 70% success rate [26]). After that the classes overlap and envelop one another so thickly that the costs of improvement increase exponentially [39]. These high costs for relatively small improvements together with the always remaining error due to class overlap make it profitable to use rejection techniques in order to lower the classification error furthermore.

In virtually all real–world classification applications, but especially in safety critical environments, the cost of a misclassification is significantly higher than the cost of a reject. Using a classifier for purposes it is not designed for will lead to high misclassification rates. A well designed novelty detector can prevent this from happening.

Chapter 2 Theory

2.1 Bayes' classifier

2.1.1 Bayes' theorem

Finding the optimal discriminant function for a given classification problem is a question of minimizing the probability of error. The theory of this optimization process is rooted in Bayesian statistics.

Let g be a classifier that maps a d–dimensional pattern or feature vector to a class $j \in \{1, .., c\}$, where c is the number of possible classes. Then this classifier can be regarded a function $g : \mathbb{R}^d \rightarrow \{1, ..., c\}$ that classifies a given pattern x to the class $g(x)$. Stochastically, a pattern and its associated class is modelled by the random pair $(\mathbf{X}, J)$. In terms of this random pair the error probability is defined by:

$$p_{error} = P(g(X) \neq J) \tag{2.1}$$

The *a priori* or *prior* probability of class j, which gives the portion of the population that belongs to class j, is denoted by $P_j = P(J = j)$. From a representative data set the prior probability can be calculated with n_j/n where n_j is the number of patterns in the set belonging to class j and n the size of the data set. Regarding only the prior probabilities we can build simple classifiers. Think of a peculiar character recognition problem in which the universe only consists of the characters 'a' and 'b', of which we measure d features. From a representative data set we determine P_a and P_b and find that one out of ten characters is a 'b'. If we build a classifier that classifies all input patterns to class a, then this classifier has a probability of error of 0.1 (10%). Not bad for such a simple classifier! But it is obvious that for real applications this solution is a too simple one. To lower the classification error we must take the probability density functions of the distinct classes into account. The probability density function of class j is denoted by f_j and gives the distribution of the patterns belonging to class j in the input space. In the character recognition example f_a and f_b are the density functions of the two classes a and b. Note that f_a and f_b are true density functions, so their integrals over all input space equal to unity. Since we must take the balance between the occurrence of the two characters into account, we multiply the density functions with the respective prior probabilities. When we normalize this result we get the *a posteriori* or *class conditional* probability density functions of class a and b. The general formula for the posterior probability function is:

$$q_j(x) = \frac{P_j f_j(x)}{f(x)} \tag{2.2}$$

which is known as *Bayes' theorem*. The normalization quotient $f(x)$ is the (mixed) probability density function of the total pattern vector space:

$$f(x) = \sum_{j=1}^{c} P_j f_j(x) \tag{2.3}$$

Note that because of the normalization quotient $f(x)$ in (2.2) the ensemble of all c posterior probability functions at x add to unity:

$$\sum_{j=1}^{c} q_j(x) = \sum_{j=1}^{c} \frac{P_j f_j(x)}{f(x)} = \frac{\sum_{j=1}^{c} P_j f_j(x)}{\sum_{i=1}^{c} P_i f_i(x)} = 1 \tag{2.4}$$

In terms of the random pair (X, J), the posterior probability of class j is denoted by $q_j(x) = P(J = j \mid X = x)$. In other words: $q_j(x)$ gives the probability that pattern x belongs to class j.

2.1.2 The optimal classifier

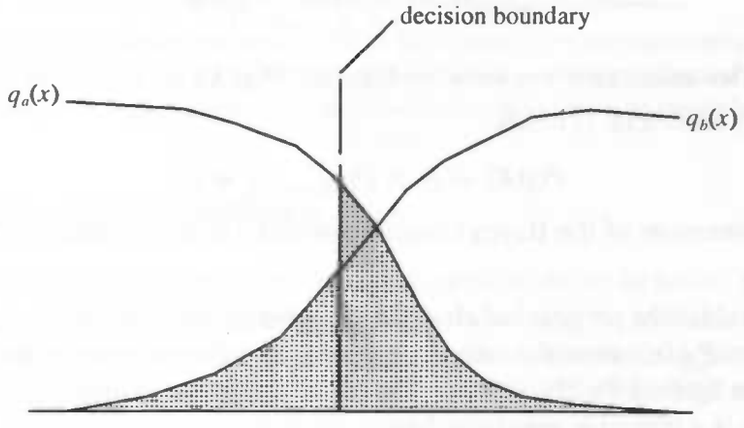

Figure 2.1: *One–dimensional classification problem with non–optimal decision boundary, q_a and q_b are the posterior probabilities of class a and b, the gray area is the probability of error.*

Suppose we know the posterior probabilities of the classes in a classification problem, then how do we chose the decision boundary to minimize the classification probability of error? Figure 2.1 shows a one–dimensional two–class classification problem with known posterior probabilities $q_a(x)$ and $q_b(x)$. The gray area shows the classification error produced by the dashed decision boundary. The gray area left of the decision boundary is the portion of the classification error that is introduced by patterns 'b' being classified as 'a', while patterns 'a' being classified as 'b' are responsible for the gray area to the right of the decision boundary. It is clear that the classification error is minimal when the boundary is positioned on the intersection of the two posterior probabilities (see figure 2.2). This optimal boundary is called the *Bayes boundary* and the minimal error probability produced by this boundary is referred to as the *Bayes error* or the *Bayes risk*.

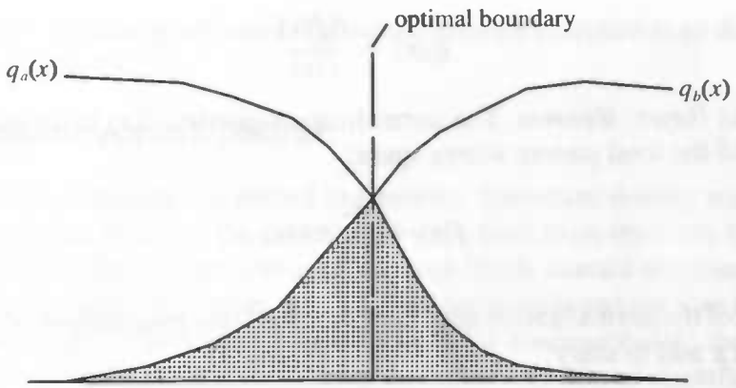

Figure 2.2: *One–dimensional classification problem with the optimal Bayes decision boundary, q_a and q_b are the posterior probabilities of class a and b, the gray area is the Bayes error*

So the optimal classifier, the Bayes classifier is defined by:

$$g_{bayes}(\boldsymbol{x}) = \underset{j = 1,\ldots,c}{argmax}\ q_j(\boldsymbol{x}) \tag{2.5}$$

Where the "argmax" operator gives the maximizing argument: the pattern $\boldsymbol{x}$ is classified to the class that maximizes $q_j(\boldsymbol{x})$. Because of (2.2) an equivalent way to define the Bayes classifier is to consider the product $P_j f_j(\boldsymbol{x})$:

$$g_{bayes}(\boldsymbol{x}) = \underset{j = 1,\ldots,c}{argmax}\ P_j f_j(\boldsymbol{x}) \tag{2.6}$$

The Bayes classifier minimizes the error probability $P(g(\boldsymbol{X}) \neq J)$. So for every classificator $g : \mathbb{R}^d \rightarrow \{1, \ldots, c\}$ lemma (2.7) holds.

$$P(g(\boldsymbol{X}) \neq J) \geq P(g_{bayes}(\boldsymbol{X}) \neq J) \tag{2.7}$$

A more formal discussion of the Bayes classificator and a proof of lemma (2.7) can be found in [10].

As described above, for the purposes of classification there is no difference in using the posterior $q_j(\boldsymbol{x})$ or the product $P_j f_j(\boldsymbol{x})$, since the only difference is that the posterior is normalized to add to unity over all j (see figure 2.3). Throughout this text no difference is made between the two notations, unless there is a plausible reason to do so.

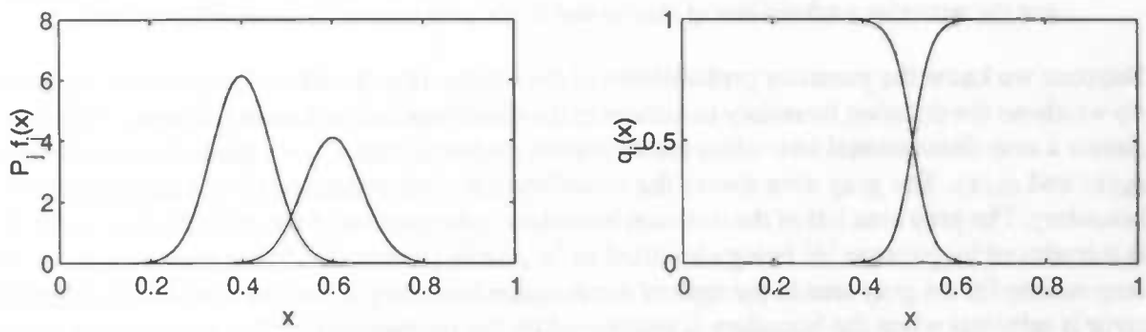

Figure 2.3: *Two weighted class density functions (left) and the corresponding posterior density functions.*

2.1.3 How to approximate the Bayes classifier?

To build the Bayes classifier for a given classification problem, we need to know the class conditional density functions $q_j(x)$ for all classes j. However, in most real world pattern recognition problems these functions are unknown. Therefore we need to make some kind of approximation to get as close as possible to the Bayes classifier. Most common classification techniques estimate the posterior densities from training data. The probability of misclassification of the resulting classificator depends very much on the accuracy of the estimations in the area near the Bayes boundary. Estimation errors in this area introduce an added error on top of the Bayes error, as shown in figure 2.4. The Bayes error is also referred to as the *intrinsic classification error*, while the error added by the estimation process is referred to as the *extrinsic classification error.*

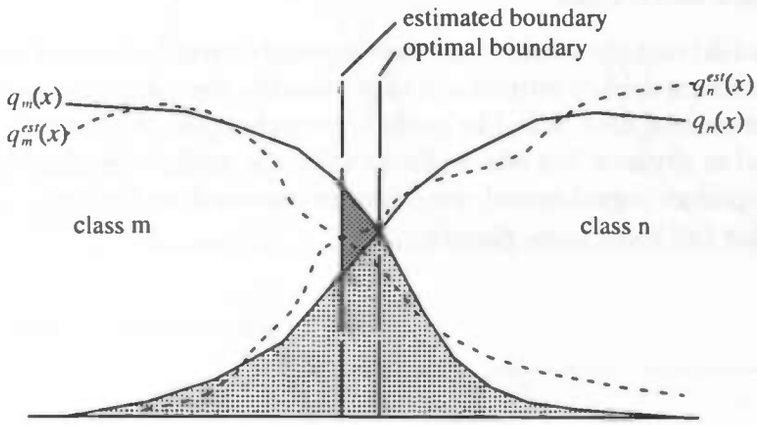

Figure 2.4: *Decision boundaries and error regions associated with approximating posterior probabilities [43]. The solid lines are the true densities, the dotted lines are the estimated densities. The Bayes error and the added estimation error are respectively shown by the light shaded and the dark shaded area.*

2.2 Classifying classifiers

In the literature different taxonomies of classification methods can be found. Most of those taxonomies classify classifiers according to the paradigm underlying the method. In the next two paragraphs two of such a taxonomies are described. In the third paragraph we propose a new criterion by which different classification methods can be subdivided, namely if the method estimates the class densities, the Bayes boundaries or the class boundaries.

2.2.1 parametric versus model–free

A common subdivision of classifiers separates classifiers the parametric and non–parametric techniques. parametric approaches assume a particular functional form for the density functions. This function has a number of adjustable parameters through which the function is fitted to the data. In most cases multivariate normals are used. Drawback of this model is, that the chosen functional form must be capable of approximating the density function accurately. The non–parametric approach does not assume a particular functional from, but lets the form of the density depend on the data alone. Some non–parametric methods suffer from the problem that the number of free parameters in the model grows with the size of the data. These methods yield a slow model in case of a big training set, i.e. evaluating the estimated function for a given pattern x can become very slow. Note that the name non–parametric is not well chosen, since models that are said to

be non–parametric do have parameters. Therefore in some literature is spoken of "model–free" instead of non–parametric.

2.2.2 Statistical versus neural

Historically pattern recognition is rooted in statistics. Statistical density estimation techniques developed in the fifties [12] and the sixties [30] were used from the early beginning of pattern recognition. Statistical density estimation techniques like k–nearest neighbor and kernel/parzen estimation are multi–purpose because they are non–parametric and are able to yield accurate estimations. But these techniques suffer from the curse of dimensionality: the model complexity rises with the size of the training set. For complex high–dimensional classification problems the models tend become unworkable.

This problem was a driving force behind the development of artificial neural networks. The model complexity of a neural density estimator is determined by the complexity of the network rather than the size of the training data. Still, like model–free techniques, the functional form of the density is not assumed in advance, but one is able to tailor the model–complexity to the underlying problem. Some literature regard neural classifiers an intermediate form of parametric and non–parametric and thus call them semi–parametric.

2.2.3 Density, Bayes boundary or class boundary estimation

Regarding the focus of the estimation, classifiers can be subdivided into three groups: classifiers that use density estimation, classifiers that estimate the Bayes boundary and those that estimate the class boundary.

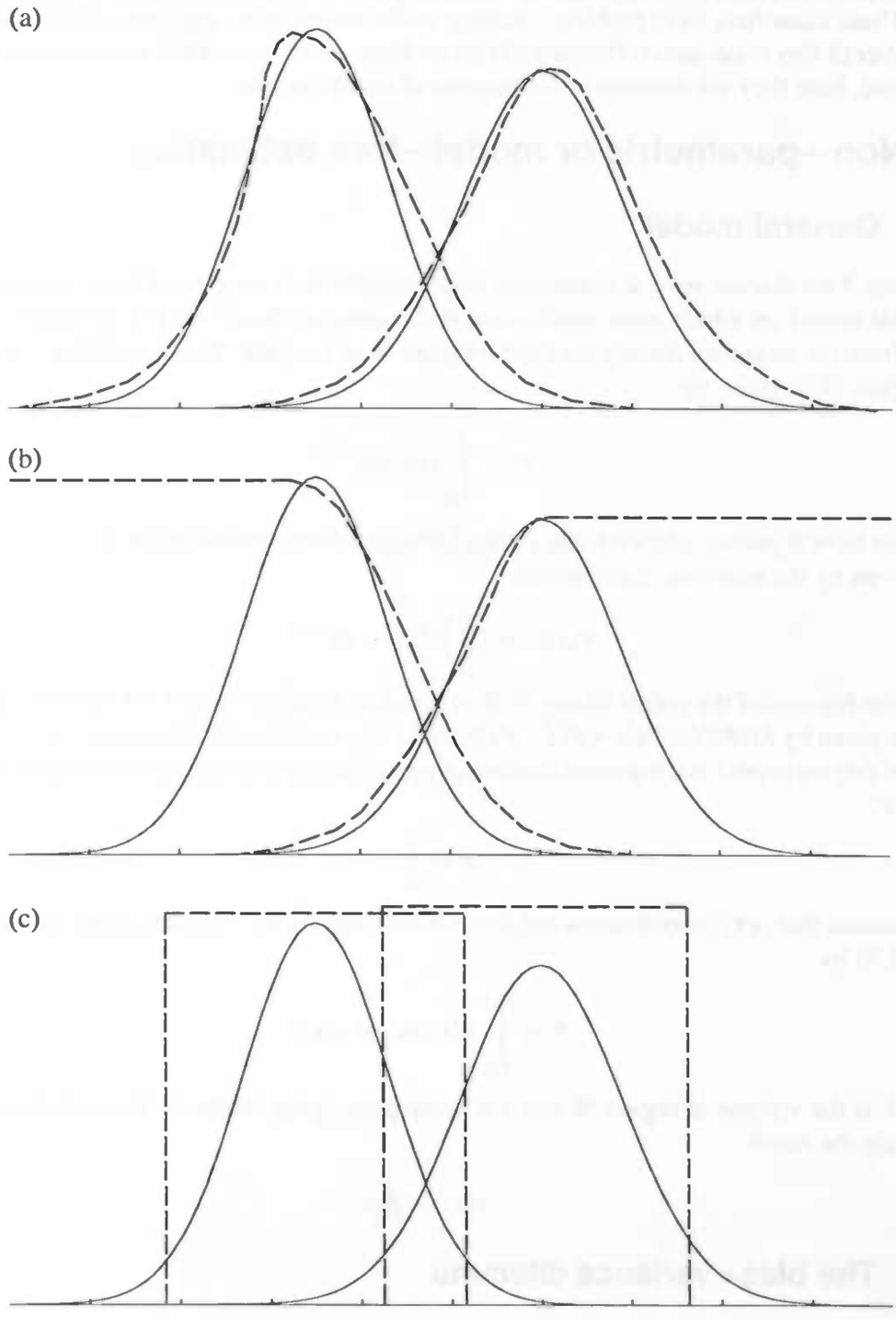

Figure 2.5: *The three basic estimation types used for classification: (a) density estimation, (b) Bayes boundary estimation and (c) class boundary estimation.*

A classifier belonging to the first group (density estimation) has a full model of the posterior density of each distinctive class as shown in figure 2.5 (a). But, for the sole purpose of classification it is only necessary to estimate the posterior densities near the Bayes boundaries. Classifiers that estimate the Bayes boundary (b) make use of this and waste no model parameters on modelling non–interesting areas of the pattern space. The third group consists of classifiers that estimate the boundaries of the classes (c). In pattern space the areas belonging to the particular classes are indicated. These classifiers have problems dealing with class overlap, yielding a large 'doubt'–area or a scatter of tiny class–areas. Because of this problem, these classifiers are not commonly used in practise, here they are mentioned for reasons of completeness.

2.3 Non–parametric or model–free estimation

2.3.1 General model

In chapter 3 we discuss several commonly used classifiers. In support of that overview we give a general model on which most model–free techniques are based on [4]. Consider a vector $\boldsymbol{x}$ drawn from the unknown density $p(\boldsymbol{x})$ and a region $\Re$ of x–space. The probability that $\boldsymbol{x}$ will fall into region $\Re$ is given by:

$$P = \int_{\Re} p(\boldsymbol{x}')d\boldsymbol{x}' \quad . \tag{2.8}$$

When we have N points independently drawn from $p(x)$, the probability that K of them fall within $\Re$ is given by the binomial distribution:

$$P_N(K) = \binom{N}{K} P^K (1 - P)^{N-K} \quad . \tag{2.9}$$

The mean fraction of the points falling in $\Re$ is given by $E(K/N) = P$ and the variance around this mean is given by $E((K/N - P)^2) = P(1 - P)/N$. So if $N \rightarrow \infty$ the distribution is a sharp peak. Because of this we expect that the mean fraction of points falling in $\Re$ is a good estimate of the probability P:

$$P \simeq \frac{K}{N} \quad . \tag{2.10}$$

If we assume that $p(\boldsymbol{x})$ is continuous and doesn't vary much over region $\Re$, then we can approximate (2.8) by

$$P = \int_{\Re} p(\boldsymbol{x}')d\boldsymbol{x}' \simeq p(\boldsymbol{x})V \tag{2.11}$$

where V is the volume of region $\Re$ and $\boldsymbol{x}$ is some point lying inside $\Re$. From (2.10) and (2.11) we obtain the result:

$$p(x) \simeq \frac{K}{NV} \tag{2.12}$$

2.3.2 The bias–variance dilemma

In the derivation of (2.12) we make two assumptions whose validity depends on the choice of the size of the region $\Re$. The approximation (2.10) is most accurate when the region $\Re$ is large,

because then P is large and the binomial distribution (2.9) is sharply peaked. The approximation (2.11) is most accurate if $\mathcal{R}$ is relatively small, because then $p(x)$ is approximately constant over the integration region $\mathcal{R}$. As a result of this there will be an optimum size of $\mathcal{R}$ that gives the best approximation of $p(x)$ for a given dataset. Note that this is a bias–variance trade–off. If $\mathcal{R}$ is large the bias gets large, resulting in an inflexible, over–smoothed model. If $\mathcal{R}$ is small the variance gets large and the model becomes very sensitive to the individual data point, resulting in a noisy model [4], [13], [14].

Chapter 3 Classification Methods

3.1 Introduction

This chapter describes a number of classification methods. Most described methods are commonly used, others are adopted of reasons of completeness. Of each technique at least a description of the model and it's parameters is given. Further is stated in which of the subclasses of paragraph 2.2.3 (density, Bayes boundary or class boundary estimation) the technique falls.

3.2 Parametric density estimation

3.2.1 Model

Parametric approaches using multivariate normals are Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA) [11], [25]. These methods assume that the underlying distribution of the data set follows a Gaussian distribution. So these classifiers are optimal when the actual distribution is a Gaussian one. Since in complex problems the distributions are rarely Gaussian, the parametric techniques are omitted in the rest of the text.

3.2.2 Parameters

Estimation parameters are the covariance matrix and the mean vector for each class. LDA uses equal class covariances while QDA uses a different covariance matrix for each class. As a result of this the logarithm of $P_j f_j(x)$ is a linear function in the first case and a quadratic function in the latter.

3.3 Kernel/Parzen estimation

3.3.1 Model

A classical model–free form of density estimation is the Kernel or Parzen estimation [11], [30]. This method uses a kernel function or Parzen window $H(u)$ that satisfies

$$H(u) \geq 0 \tag{3.1}$$

and

$$\int H(\boldsymbol{u})d\boldsymbol{u} = 1 \ . \tag{3.2}$$

Consider a kernel function corresponding with a unit hypercube centered at the origin defined by:

$$H(\boldsymbol{u}) = \begin{cases} 1 & |u_i| < \frac{1}{2} \\ 0 & otherwise \end{cases} \tag{3.3}$$

with $i = 1,...,d$. For all data points $\boldsymbol{y}$ the quantity $H((\boldsymbol{x} - \boldsymbol{y})/h)$ is equal to unity if the point $\boldsymbol{y}$ falls inside a hypercube with sides of length h, centered at $\boldsymbol{x}$ and is zero otherwise. The total number of data points from a database $\boldsymbol{D}$ falling inside the h–sided hypercube centered at $\boldsymbol{x}$ is:

$$K = \sum_{n=1}^{N} H\left(\frac{\boldsymbol{x} - \boldsymbol{D}_n}{h}\right) \ . \tag{3.4}$$

The volume of the h–sided hypercube is given by:

$$V = h^d \ . \tag{3.5}$$

If we substitute (3.4) and (3.5) in the general formula for the density estimation (2.12) we obtain:

$$p_K(\boldsymbol{x}) = \frac{1}{Nh^d} \sum_{n=1}^{N} H\left(\frac{\boldsymbol{x} - \boldsymbol{D}_n}{h}\right) \tag{3.6}$$

which is an estimation for the density at $\boldsymbol{x}$. The model density $p_K(\mathbf{x})$ can be seen as the superposition of N hypercubes each centered at a data point. The kernel function is here chosen to be a hypercube for reasons of clarity, but is not very usable in practice, since it results in a discontinue model density function. Commonly used kernel functions are Gaussian, Cauchy and Hermite functions [10]. When a multivariate Gaussian kernel function is used the density approximation becomes:

$$\tilde{p}(\boldsymbol{x}) = \frac{1}{N(2\pi h^2)^{d/2}} \sum_{n=1}^{N} \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{D}_n\|^2}{2h^2}\right) \ . \tag{3.7}$$

3.3.2 Parameter

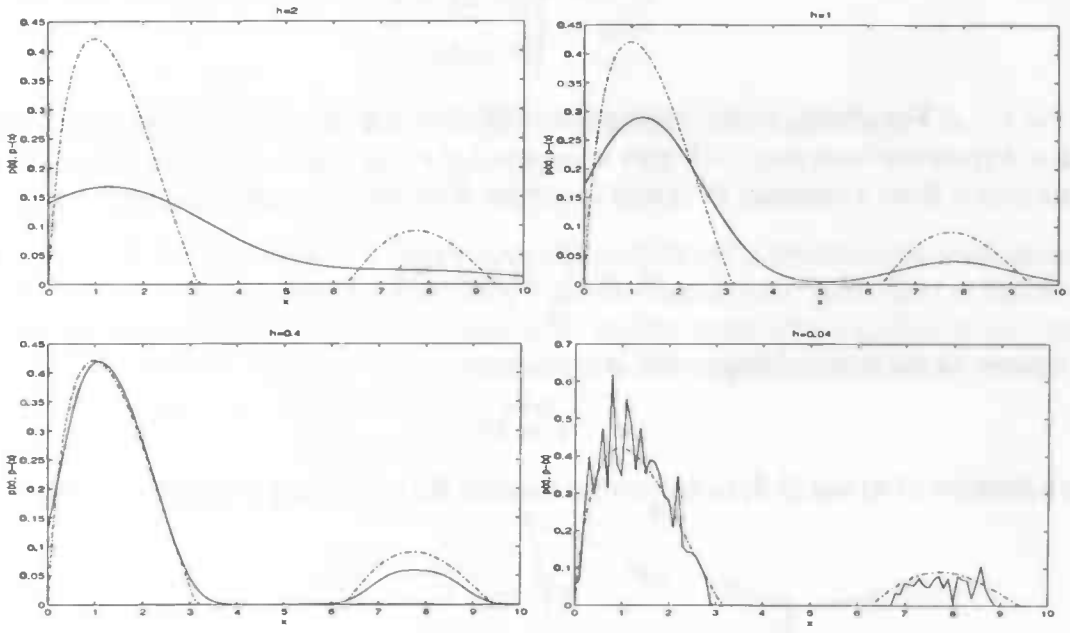

Figure 3.1: *Kernel estimation using a Gaussian kernel function, with different values for h. The solid line is the estimation from data drawn from the true density showed by the dash–dotted line.*

The kernel function width h acts like a smoothing parameter. For a particular dataset drawn from a true density function an appropriate choice for h has to be made in order to obtain a good approximation of that density (Figure 3.1). If h is chosen too large the estimation is over–smoothed. If, on the other hand h is too small, particular properties of the data as supposed to the properties of the true density gain influence and the estimation becomes noisy.

3.3.3 Classification

In a classification problem a model of the data distribution of each class is obtained using kernel density estimation. These models are used for the Bayesian decision making. Since a full model of each class density function is created this classification method belongs to the subclass of density estimating classifiers.

3.4 k–Nearest Neighbor estimation

3.4.1 Model

Another model–free form of density estimation is the k–Nearest Neighbor (k–NN) technique [12]. This technique is like kernel estimation based upon the general formula for the density estimation $p(x) \simeq K/NV$ (2.12). While for kernel estimation the V is fixed and the K varies, for k–NN a constant K is chosen and V is varied. Consider the smallest hypersphere centered at x with exactly K data points inside it. The density $p(x)$ is given by (2.12), where V is the volume of the

hypersphere. A disadvantage of the *k*–NN technique is that the resulting estimate is not a true probability density since its integral over all x–space doesn't equal to unity under all circumstances.

3.4.2 Parameter

Figure 3.2 shows an example of a *k*–NN approximation of a true density $p(x)$, using data drawn from the density. Again there is a smoothing parameter (K) by which the bias/variance trade–off can be optimized. Small values of K yield a noisy density estimation due to a great influence of the individual data point. Density estimations with large values of K suffer from over–smoothing.

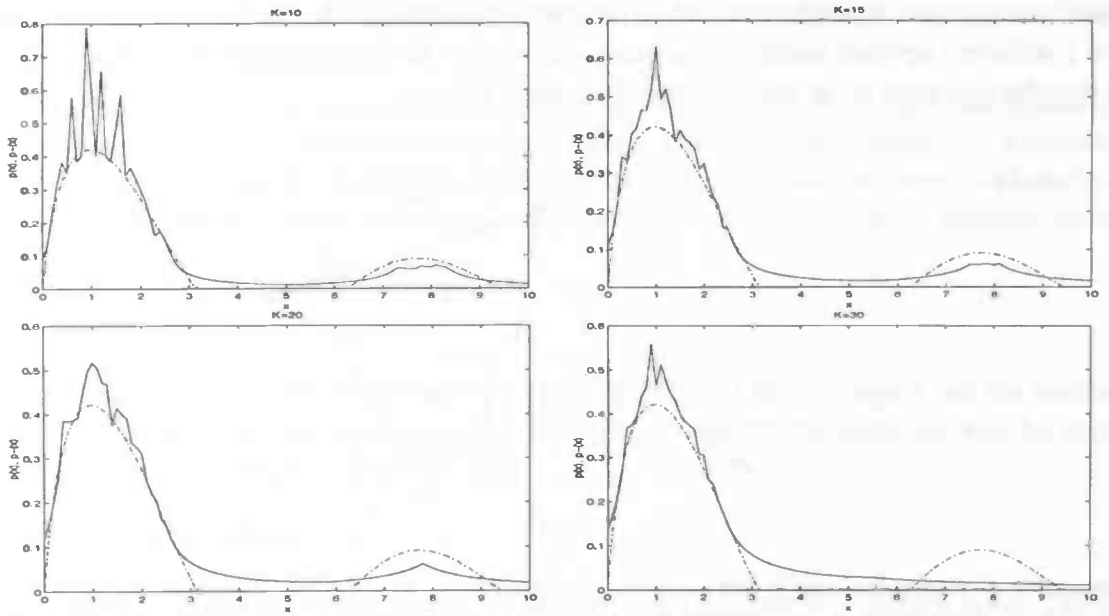

Figure 3.2: *k–Nearest Neighbor estimation with different values for K. The solid line is the estimation from data drawn from the true density showed by the dash–dotted line.*

3.4.3 Classification

In a Bayesian classification problem, classifying $\boldsymbol{x}$, the values of the posterior density functions $f_j(\boldsymbol{x})$, $j \in \{1,\ldots,c\}$ need to be estimated. For each class j an $\boldsymbol{x}$–centered hypersphere enveloping K data points of class j is determined to estimate $f_j(\boldsymbol{x})$. The pattern $\boldsymbol{x}$ is classified to the class j that maximizes $P_j f_j(\boldsymbol{x})$, where P_j is the prior probability of class j. This approach is called the *grouped* form of the *k*–NN technique.

A slight variant is the *pooled* form, which has a lower time–complexity than the grouped form, since it estimates all posterior probabilities $q_j(\boldsymbol{x})$ at once, using one sphere V with K data points inside it. Suppose we have a dataset of N pattern vectors of which N_j vectors belong to class C_j, so that $\sum_j N_j = N$. Then the according to (2.12) the class–conditional density of class j can be estimated from

$$f_j(x) = \frac{K_j}{N_j V} \quad . \tag{3.8}$$

The density of the entire data at x can be estimated from

$$f(x) = \frac{K}{NV} \tag{3.9}$$

while the prior probability of class j can be estimated as

$$P_j = \frac{N_j}{N} \quad . \tag{3.10}$$

If we substitute the last three equations in Bayes' theorem (2.2) we have

$$q_j(x) = \frac{P_j f_j(x)}{f(x)} = \frac{K_j}{K} \tag{3.11}$$

which is pooled k–NN estimation of the class–conditional density. Thus we minimize the classification error if we assign pattern x to the class with the highest ratio K_j/K. This is known as the *k–nearest neighbor classification rule*. The grouped and the pooled form applied to the same data have a different optimal value for the parameter K, but for each optimized mode similar results for the classification error are obtained ([6], page 37).

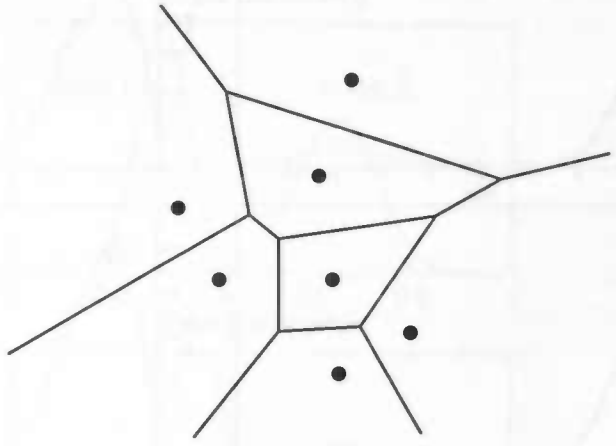

Figure 3.3: *Two–dimensional k–NN classification with K=1. At every point the decision is the class of the closest data point. A set of points that have the same closest data point is called a Voronoi cell. All Voronoi cells from a Voronoi partition.*

Since a k–NN classifier is an approximation to the Bayes classifier its error is greater than the minimal possible Bayes error. However it can be shown [11] that, with an infinitive number of data points, the error is never worse than twice the Bayes error. Under the assumption of an infinitive dataset the difference between the error of the k–NN rule and the Bayes error decreases when the number of neighbors K increases. In real world applications (with a finite number of data points) the error will depend on "accidental characteristics" of the data and, as a consequence, the parameter K must be adapted for each case.

It may be clear that classifiers using k–NN belong to the subclass of density estimating classifiers.

3.5 Multilayer Perceptron

3.5.1 Model

The Multilayer Perceptron is the most important class of neural networks. Typically the network consists of a set of sensory units referred to as the *input layer*, one or more *hidden layers* of com-

putational nodes and an *output layer* of computational nodes. The input signal propagates through the network in a forward direction, on a layer–by–layer basis. MLPs can be trained in a supervised manner with the *error back–propagation algorithm*.

An MLP has three distinctive characteristics [16]:

- The model of each neuron in the network includes a *nonlinearity* at the output end. The nonlinearity must be smooth (i.e., differentiable everywhere). A commonly used form of nonlinearity that satisfies this requirement is a *sigmoidal nonlinearity* defined by the *logistic function*: $y_j = 1/(1 + exp(-v_j))$, where v_j is the net internal activity level of neuron j and y_j is the output of the neuron.
- The network contains one or more layers of hidden neurons that are not part of the input or output of the network. These hidden neurons enable the network to learn complex tasks by extracting more meaningful features from the input patterns.
- The network exhibits a high degree of *connectivity*, determined by the synapses of the network.

Through the combination of these characteristics together with the ability to learn from experience by training the MLP gains it's computing power. However, the presence of a distributed form of non–linearity and the high connectivity of the network make the theoretical analysis of a multilayer perceptron difficult to undertake. Further makes the use of hidden neurons the learning process harder to visualize.

3.5.2 Parameters

The MLP has parameters for the network structure: number of hidden layers and the number of hidden neurons per layer, and learning process parameters like the learning rate and the learning momentum. A better insight is given in [16].

3.5.3 Classification

For classification tasks the output coding of the MLP can be done in different ways. The most common is to use c output neurons, one for each class. The target output for class j is coded as a vector with all c components set to 0, except the j^{th} set to 1.

The hidden unit representations depend on weighted linear summations of the inputs, transformed by monotonic activation functions. Thus the activation of a hidden unit in a multilayer perceptron is constant on surfaces which consist of parallel (d–1)–dimensional hyperplanes in d–dimensional input space [4]. An output unit of an MLP combines several of these hyperplanes to a decision boundary, as is shown in figure 3.4 for a 2D problem where the decision boundary is formed by a combination of two lines.

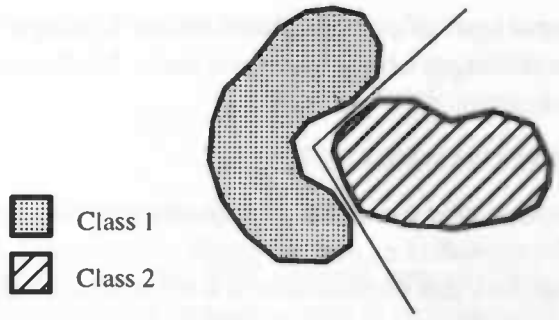

Figure 3.4: *Typical decision boundary of MLP–classification.*

It my be clear that an MLP partitions the whole pattern space into class areas like Bayes boundary estimators do. The output activations approximate the posterior densities [34], [38], as shown for an 1D case in figure 3.5. This implies that the MLP belongs to the Bayes boundary estimating classifiers.

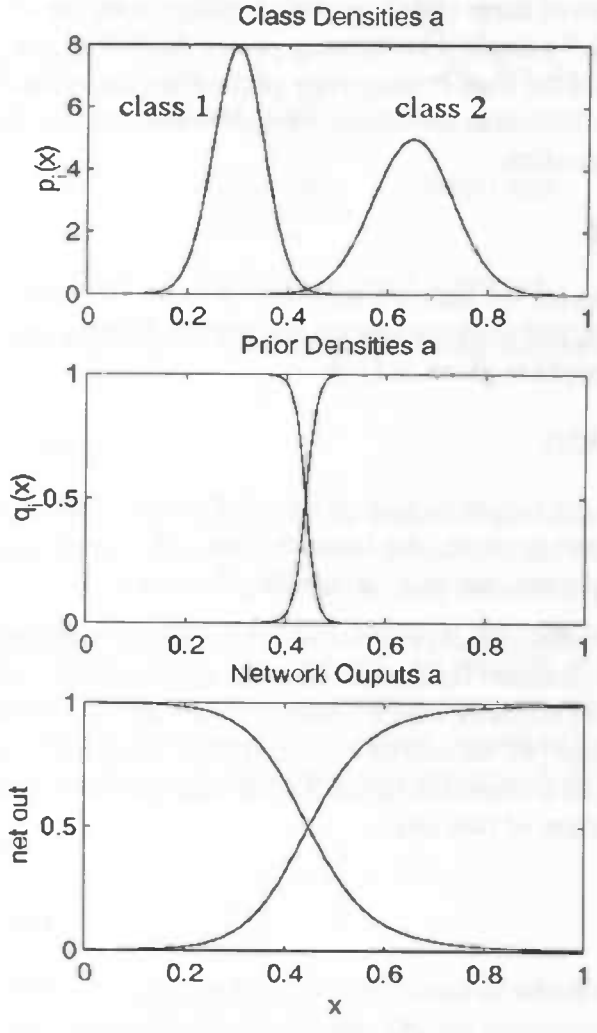

Figure 3.5: *The outputs of an MLP based classifier approximates the posterior class densities.*

3.6 Radial Basis Function Network

3.6.1 Model

The Radial Basis Function Network is a neural network which is closely related to kernel density estimation. Major difference is that the model complexity of kernel estimation is determined by the size of the training data, while the complexity of a RBF network is determined by the size of the network and can thus be tuned to the complexity of the mapping problem. Both kernel estimation and RBF networks can be regarded as variants of a technique for exact interpolation. This technique, the radial basis function interpolation [32], generates an interpolant that passes exactly through the data points $\boldsymbol{D}_n$ by using a set of N *basis functions*, one for each data point. These basis functions have the form $\phi(\|\boldsymbol{x} - \boldsymbol{D}_n\|)$, where ϕ is a non–linear function, which depends on the distance between $\boldsymbol{x}$ and $\boldsymbol{D}_n$. The interpolant is a linear combination of the N basis functions:

$$h(\boldsymbol{x}) = \sum_{n=1}^{N} w_n \phi(\|\boldsymbol{x} - \boldsymbol{D}_n\|) \ . \tag{3.12}$$

For a large class of functions the weights w_n can be determined using matrix inversion techniques. The most common basis function is the Gaussian

$$\phi(x) = \exp\left(-\frac{x^2}{2\sigma^2}\right) \tag{3.13}$$

where the basis function width σ acts as a smoothing parameter for the interpolant. The Gaussian function is a localized function, i.e.: if $|x| \rightarrow \infty$, then $\phi(x) \rightarrow 0$. For the use of radial basis functions in exact interpolation non–localized functions can be used, but in the scope of this text we only consider localized basis functions.

The radial basis function network model is obtained by applying some changes to the procedure for exact interpolation using basis functions [7], [29]:

- The number M of basis functions is not equal to the number N of data points, but reflects the complexity of the mapping function. M is typically much less than N.
- The centers of the basis functions are not determined by the data points, but are determined during training.
- The width parameter σ is not fixed and the same for every basis function, but the appropriate width for each basis function is determined during training.
- For the difference between the average value of the basis function activations over the whole data set and the average value of the targets is compensated by bias parameters.

The mapping function of the RBF network can be written as:

$$y_k(\boldsymbol{x}) = \sum_{j=1}^{M} w_{kj} \phi_j(\boldsymbol{x}) + w_{k0} \ . \tag{3.14}$$

Without loss of generality we can put the bias term w_j inside the equation by introducing a basis function ϕ_0, whose activation equals to unity:

$$y_k(\boldsymbol{x}) = \sum_{j=0}^{M} w_{kj} \phi_j(\boldsymbol{x}) \ . \tag{3.15}$$

For the RBF network the Gaussian basis function is

$$\phi_j(x) = \exp\left[-\frac{\|x - \mu_j\|^2}{2\sigma_j^2}\right] \tag{3.16}$$

where μ_j is a d–dimensional vector to the center of basis function j.

The network structure (figure 3.6) of the RBF network consists three layers, an input layer, a hidden layer of radial basis functions and an output layer of neurons with a linear activation function. The input layer is fully connected to the hidden layer, the weights of these connections are all set to unity. The hidden layer is fully connected to output layer with weighted connections.

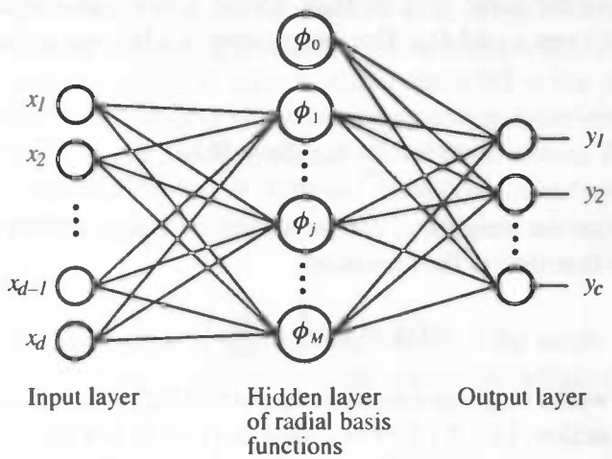

Figure 3.6: *Radial basis function network.*

A deeper view into the theoretical aspects of radial basis function networks can be found in [4] and [16].

3.6.2 Network training

The different roles that the non–linear hidden and the linear output layer of the RBF network play, are reflecting in different learning strategies for the second and the third layer . The centers of the basis functions can either be at fixed randomly chosen positions or positioned in a self–organized or a supervised learning strategy. Since the output of the neurons in the output layer is a linear combination of the outputs of the hidden layer, the weights of the connections between these layers can be calculated using the same matrix techniques used for the exact interpolation problem [7]. Another approach to determine the weights of the output layer is supervised learning in an error–correcting fashion. See [16] for a more comprehensive description.

3.6.3 Parameters

3.6.4 Classification

The basis functions of a RBF network are localized functions of which the centers and the width/height–ratio are determined during the learning stage. During evaluation the hidden units use the distance to a prototype vector followed by a transformation with a localized function. The activation of a basis function is therefore constant on concentric (d–1)–dimensional hyperellipsoids in

d–dimensional input space [4]. The output of the network is a linear combination of these hyperellipsoids as shown in figure 3.7. So, the output activation for class j on a particular pattern x is the sum of the kernel basis functions of j at x. This activation can be considered an approximation of the posterior probability of class j at x [23], [44]. Difference with the MLP is that for the RBF network all the output activations become zero at areas in the input space in which no training data was present. The localized basis functions make the RBF network a density estimating classifier. Note that the training process aims to optimize the classification. As a consequence, basis functions lying further away from a decision boundary will not represent the underlying distribution as accurate as those nearer to a boundary.

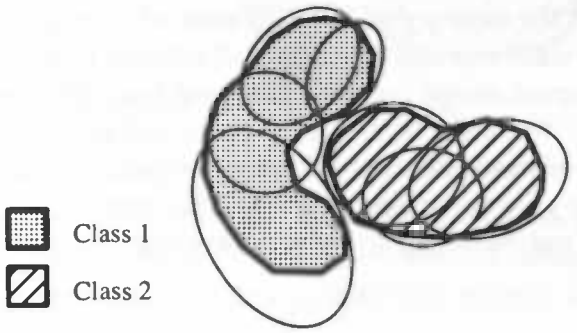

Figure 3.7: *Typical class boundary of RBF–classification.*

3.7 Reduced Coulomb Energy

3.7.1 Model

Reduced or Restricted Coulomb Energy [33] is a neural classification method, based on decision prototypes that are characterized by their influence regions. These regions are represented by hyperspheres around the prototype. The input space is divided in class–zones, each one consisting of hyperspheres round the different prototypes of that particular class.

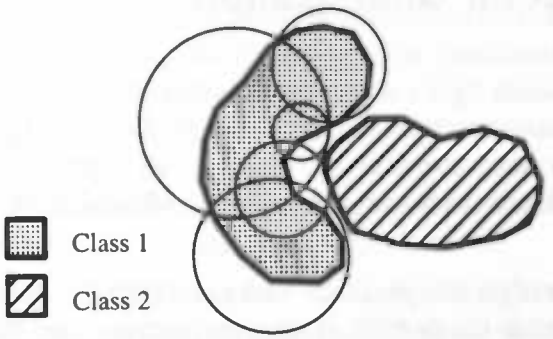

Figure 3.8: *RCE classification. The prototype regions of class 1 are shown.*

RCE is an incremental neural network, i.e. the number of classes and the number of prototypes is not reflected in the model prior to the learning phase. The learning algorithm is supervised. If a training pattern falls outside the influence regions of it's associated class, a new prototype is created with an initial chosen radius. If the pattern falls inside one or more influence regions of a wrong class, the radii of these regions are reduced.

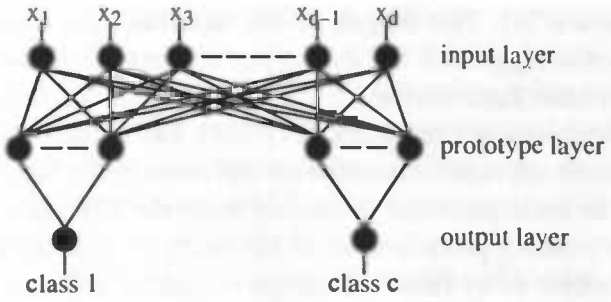

Figure 3.9: *The structure of the RCE network*

As described in figure 3.9 the structure of the RCE network consists of three layers. The layers are connected in a feedforward way and the number of neurons in the hidden prototype layer and in the output layer are adapted during training. The input layer and the prototype layer are fully connected, while a unit in the output layer is only connected to the prototypes of it's class. The connection weights between the input layer and the prototype layer are fixed for each unit created in the prototype layer. They are never modified after their creation. The connection weights between the prototype layer and the output layer are always one. The prototype units have each their local parameter "radius".

3.7.2 Parameters

The initial radius of the prototypes is the main parameter of the model. The value of this parameter in not critical as long as it is not chosen too small, since the radius of a prototype only decreases during learning.

3.7.3 Classification

RCE is a form of class boundary estimation. The performance of the RCE network is not good in case of overlapping classes. In the overlapping area a large number of prototypes with very small influence regions emerge. This has a bad influence on the classification error as well as on the memory and computational requirements.

3.8 Learning Vector Quantization

3.8.1 Model

The Learning Vector Quantization model, proposed by Kohonen [20], [22], is a simple adaptive method of vector quantization capable of learning in an supervised manner. LVQ is a popular method because of its effectiveness as a classifier combined with relatively short training and evaluation times.

LVQ uses a finite number of prototypes, each with a class label. In the input space these prototypes are randomly placed inside the domain of their respective class. During the learning phase the positions of the prototype vectors are changed. Consider a training set $\boldsymbol{T}$ of N pairs (x,j), $x \in \mathbb{R}^d$, $j \in \{1,\ldots,c\}$ and a set $\boldsymbol{\Theta}$ of K random vectors $\theta_k \in \mathbb{R}^d$. Each vector θ_k has a label L_k which associates it with a class j. During the learning phase for each member $(\boldsymbol{x}, j)$ of the training set the closest prototype θ_c is considered. The prototype is adjusted according to:

$$\theta_c = \begin{cases} \theta_c + \alpha(x - \theta_c), & \text{if } L_c = j \\ \theta_c - \alpha(x - \theta_c), & \text{if } L_c \neq j \end{cases} \tag{3.17}$$

with $0 \leq \alpha \leq 1$. The other prototypes remain the same. This update has the effect that if data point x and prototype θ_c have the same classlabel, the prototype is moved towards the data point. If the classes differ the prototype is moved in the opposite direction. After several passes through the training set, the prototype vectors converge [2] and training is complete. During the test phase a data point is classified to the class associated with the nearest prototype.

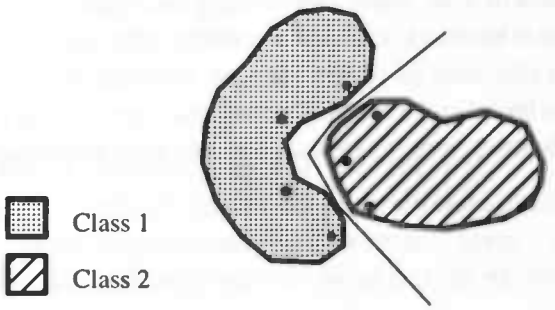

Figure 3.10: *LVQ classification. Only the prototypes that influence the class boundary are shown.*

In the literature different names for the prototype vectors occur. In the original publications by Kohonen the complete set of prototypes is called the *codebook*, while to the individual prototype vector is referred by *codebook vector*. Other authors refer to them as *Voronoi vectors*.

3.8.2 Parameters

The LVQ model has two parameters: the gain α, $0 \leq \alpha \leq 1$ and the number of prototypes. The gain α may be constant or decrease monotonically with the time. The Elena benchmark study [6] shows that the influence of α on the classification error is negligible if α is not chosen over 0.5. The influence of the number of prototypes and the relative number of prototypes per class is more important. For an upper bound for the number of prototypes, 10% to 20% of the size of the learning database can be used. Although choosing the same number of prototypes of each class can be a good strategy even if the prior probabilities of the classes differ greatly, balancing according to the prior probabilities can be beneficial. More sophisticated balancing techniques, using *a priori* knowledge of the training data or properties of the codebook during learning can be applied sucessfully [21].

Another important issue is the initialization of the prototype vectors. Since after learning all prototypes must be surrounded by training data points of their associated class, a common method is initialize prototypes at area's of the pattern space with a high density of training data.

3.8.3 Classification

After training is the distribution of the prototypes roughly the same as the training data distribution therefore LVQ can be seen as a density estimating classifier.

The boundaries between classes established by LVQ do not approximate the optimal Bayes boundary accurately. Therefore improved versions (LVQ2 and LVQ3) are proposed. LVQ2 shifts the decision borders differentially towards the Bayes boundary, but the process is not robust in the sense that prototypes may not converge. This last problem has been solved in LVQ3. In [21] a description of the three LVQ–algorithms is given.

3.9 Discussion

Using kernel estimation or k–Nearest Neighbors estimation accurate models of the distinct class distributions can be obtained. Problem, however, is that these models become too complex when

the complexity of the underlying problem gets high and large training sets are needed. Evaluation times and memory requirements tend to rise to unacceptable heights. Neural approaches lack these problems, since the complexity of the network is supposed to reflex the complexity of the problem. The price one pays for this advantage in computational and memory requirements is the longer learning time of the neural approaches. The 'training' of a statistical model consists of storing the training data in a database and finding the right value for one (smoothing) parameter. The training of a neural network will take far more time and appropriate values must be found for far more parameters (number of hidden layers, number of hidden units per layer, etc.). One of the reasons that neural techniques are used more and more is that the benefits of a computational faster model weight heavier than the extra time it takes to find that model.

Of the four described neural classification techniques Reduced Coulomb Energy has clearly disadvantages. RCE yields a great scatter of tiny class areas in the regions with class overlap. This makes the model complexity far too large and the classification performance becomes poor.

The comparison of the MLP and the LVQ in the Elena benchmark study [6] shows on the average a small advantage of the MLP, although the differences are small. Both the MLP and the RBF network are universal approximators, thus there always exists an RBF network capable of accurately mimicking a specified MLP, or vice versa [16]. So, the classification strenghts of the MLP, the RBF network and the LVQ are approximately equal. In spite of this, the MLP is used far more often than any other neural pattern recognition technique. The straight–on learning procedure of the MLP is one of the reasons for this popularity. Further needs the MLP typically less hidden nodes than the LVQ or the RBF network, since the latter two suffer more from the curse of dimensionality [16].

Classifier	Type	Estimation	Learning Time	Complexity	Accuracy	Relative Usability
Kernel	statistic	density	very short	data size	++	–
k–NN	statistic	density	very short	data size	++	–
MLP	neural	Bayes boundary	long	model size	+	++
RBF network	neural	density	long	model size	+	+
RCE network	neural	class boundary	short	model size	–	–
LVQ	neural	density	long	model size	+	+

Table 3.1: *Properties of the described classifiers. The relative usability reflects the model complexity relative to the problem complexity. Used symbols: ++ very good; + good; – bad.*

3.10 Conclusions

From this chapter the following conclusions can be drawn:

- Classifiers based on kernel or *k*–NN estimation are the best approximators of the Bayes classifier, but for the highly complex problems addressed in this report their usability is low due to their high computational and memory burden.
- Both the MLP, the LVQ and the RBF network have a high accuracy relative to their usability for complex problems. The MLP scores best because the MLP suffers less from the curse of dimensionality than the other two.

Chapter 4 Novelty Detection Theory

4.1 Introduction

Classifiers based on density estimation in combination with plain Bayesian decision making, give a firm answer on the patterns presented to them. These classifiers have only knowledge of the c possible classes they are designed for and every received input is assigned to one of these classes. As discussed in paragraph 1.2, for a great part of pattern recognition appliances we would like the classifier to asses the confidence in it's own answer. Novelty detection is an important part of the confidence figure. This chapter starts with reporting how the Bayes classifier can be extended to detect novelties. After that we discuss if the extended theory can be used in existing techniques.

4.2 Bayes classifier detecting novelties

In order to detect novelties the Bayes classifier can simply be extended with an absolute threshold level t_a. Classifying a pattern x, the posterior probability of the winning class i must exceed this threshold $(q_i(x) > t_a)$, otherwise the result is rejected as a novelty (figure 4.2). Note that this can only be done for the form of the Bayes classifier that uses the product of the posterior P_j and the class density $f_j(x)$ (equation (2.6)). The posterior density $q_j(x)$ sums per definition to unity over all j. From figure 4.1 (repeated from chapter 2) it is clear that using the posterior densities novelties cannot be detected.

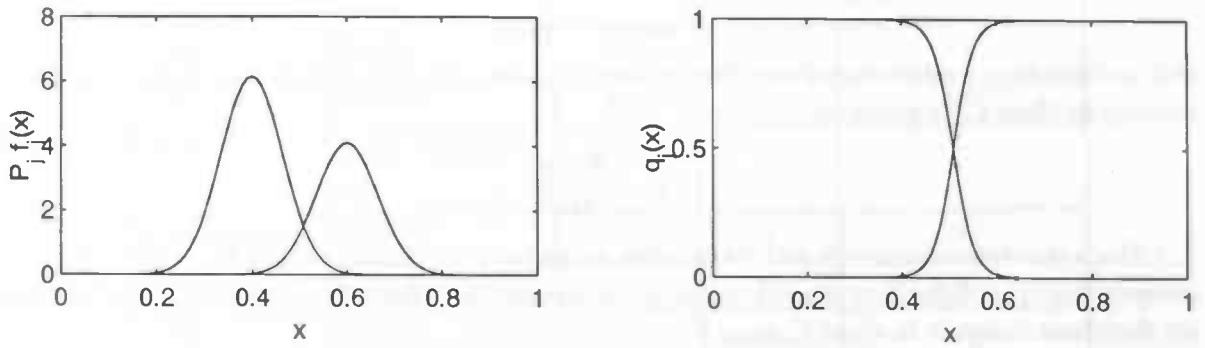

Figure 4.1: *Two weighted class density functions (left) and the corresponding posterior density functions.*

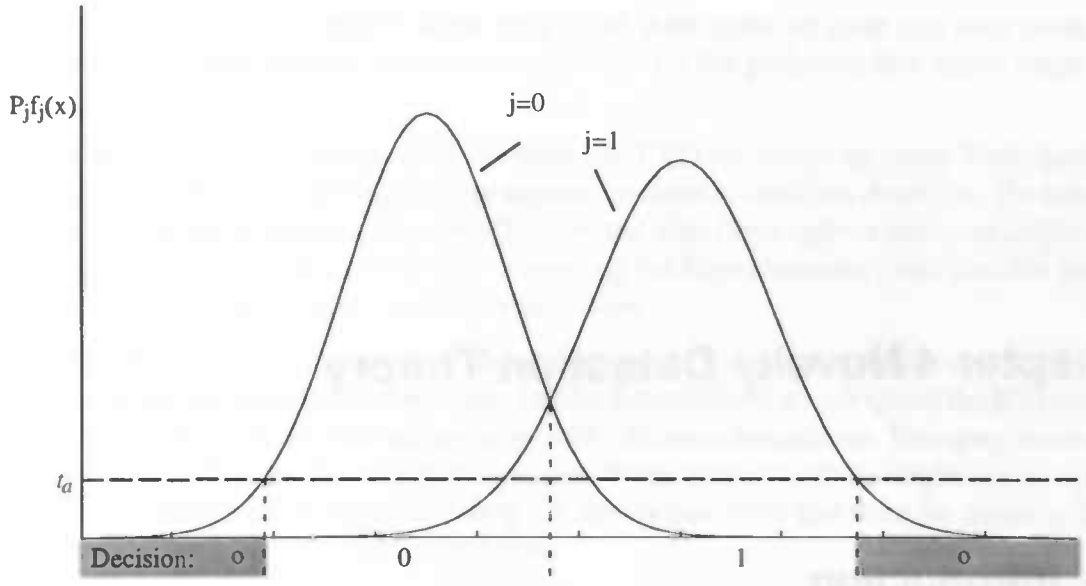

Figure 4.2: *Two class Bayesian classification with rejection of novelties using an absolute threshold level*

Note that the boundary between the classes is not affected when the threshold level is smaller than the posterior class densities at that boundary. The choice of an appropriate threshold level is not unambiguous, since there is no available information about the class of novel data. Heuristically the threshold level can be set in such a way that the major part (like 98%) of the training data is accepted.

4.3 The novelty threshold

Bishop [5] regards the threshold level as the posterior density of the class of novel data: "In applying a threshold we are implicitly classifying all new input vectors into one of two classes: those which are similar the training data, which we denote by class C_{normal}, and those which are novel, which we denote by class C_{novel}. The training data are assumed to be drawn entirely from C_{normal}, while new data could arise from either class, with prior probabilities P_{normal} and P_{novel} respectively, where $P_{normal} + P_{novel} = 1$. Given a new input vector $\boldsymbol{x}$ we wish to assign it to one of the two classes in such a way as to minimize the probability of misclassification. This is achieved when the vector is assigned to the class having the larger posterior probability, so that the vector is assigned to class C_{normal} if

$$q_{normal} > q_{novel} \tag{4.1}$$

and to class C_{novel} otherwise. From Bayes theorem, the posterior probability of the vector $\boldsymbol{x}$ belonging to class C_i is given by

$$q_i = \frac{P_j f_i(\boldsymbol{x})}{f(\boldsymbol{x})} \quad . \tag{4.2}$$

(...) Since the denominator in NO TAG is the same for both classes, it can be omitted from the comparison of probabilities needed to assign $\boldsymbol{x}$ to a particular class. Using NO TAG and NO TAG, we therefore assign $\boldsymbol{x}$ to class C_{normal} if

$$q_{norm} > \frac{P_{novel}\, f_{novel}(\boldsymbol{x})}{P_{norm}} \tag{4.3}$$

and otherwise to class C_{novel}. The quantity q_{norm} is the probability density from which the training data were drawn. The density $f_{novel}(x)$ represents the distribution of novel data, and so by definition we can have little idea of what form it should take. The simplest assumption is to take it to be constant over some large region of input space, falling to zero outside this region to ensure that the density function can be normalized. (In many applications the range of possible input values will in fact be bounded). In this case the condition NO TAG is then equivalent to at threshold on an estimation of the unconditional density of the training data." Note that in spite of this theoretical basis there is still the big assumption about the form of the distribution of the novel data, since nothing is known about this class.

4.4 Rejection in different classification approaches

In paragraph 2.2.3 we described a subdivision of classifying methods according to whether the methods model the class density functions, the Bayes boundaries or the class boundaries (see figure 2.5). The usability of the extensions to the Bayes classifier in order to reject samples on grounds of novelty depends on the type of model that a classifier has of the underlying problem.

4.4.1 Density estimation

Classifiers based on density estimation hold a full model of the class densities, hence these classifiers are able to detect novelties.

4.4.2 Bayes boundary estimation

Classifiers estimating the Bayes boundaries only hold a model of the class densities near the decision boundaries. Since this type of classifiers partition the whole input space into class areas, these classifiers are unable to detect novelties.

4.4.3 Class boundary estimation

Class boundary estimating classifiers are by definition able to reject novelties.

4.5 Discussion

Table 4.1 shows a part of table 3.1 extended with the classificator's ability to reject input patterns using the Bayesian techniques described in this chapter.

Classifier	Estimation	Accuracy	Relative Usability	Bayesian novelty detection
Kernel	density	++	–	++
k–NN	density	++	–	++
MLP	Bayes boundary	+	++	–
RBF network	density	+	+	+
RCE network	class boundary	–	–	+
LVQ	density	+	+	+

Table 4.1: *Rejection properties. The relative usability reflects the model complexity relative to the problem complexity. Used symbols: ++ very good; + good; – bad.*

Again, the non–parametric methods score very good. And again we state that their models are accurate, but their computational burden is unrealistic for the problems this report emphasizes on.

On novelty detection score both the RBF network and LVQ the predicate good. Their output activations round the decision boundaries are approximations of the class densities. However, the training process aims to learning the classification and thus these approximations might not be optimal. The models are optimized for approximating the Bayes boundary and thus the question can be raised if they can be used for novelty detection.

It is clear that the MLP is not capable of novelty detection. The transfer functions of the hidden nodes do not allow the modelling of anything but the Bayes boundary. LVQ and the RBF network do model the class boundaries that face away from the decision boundaries. However, the training process aims to learning the classification and thus these approximations might not be optimal. The models are optimized for approximating the Bayes boundary and thus the question can be raised if they can be used for novelty detection.

4.6 Conclusions

This chapter shows that the theory of the Bayes classifier can easily be extended with a reliability measure to detect novelties. On basis of this extended theory the following conclusions can be drawn:

- Classifiers based on estimation of the class density functions are by construction able to reject input patterns on grounds of novelty.
- The MLP is by construction not able to reject patterns on basis of novelty.
- The RBF network is by construction able to reject input patterns on grounds novelty.
- LVQ is by construction able to reject input patterns on grounds of novelty.

For the last two conclusions it can be questioned how accurate the rejection mechanism is. The training process of the neural classifiers is very much aimed on the accurate modelling of the Bayes boundary. As a result the full model of the class densities may become too much deformed to be useful for other purposes than classification.

Chapter 5 Novelty Rejection Techniques

5.1 Introduction

In chapter 4 we concluded for various reasons that dedicated novelty detection methods are needed. The few novelty detecting methods described in the literature are based on density estimation. These methods form a model $\hat{p}(\boldsymbol{x})$ of the unconditional density function (i.e. the density of the training data regardless of the classes) and use a threshold to decide if patterns are accepted or rejected.

5.2 Non–parametric estimation

The most simple approach of the problem is the use of non–parametric estimation techniques like k–NN or kernel estimation. Bishop uses in [5] a kernel estimator for an off–line test to show that a simple threshold level on an estimated density can be used successfully in practice. Of course for most complex on–line applications a non–parametrical method will appear to be too slow.

The kernel method is better suited for novelty detection than k–NN [41]. The k–NN method only takes into account the distance to the k–th nearest data point as shown by figure 5.1. The distribution of the k–1 nearest points is neglected, while the kernel method takes the distance to all data points into account.

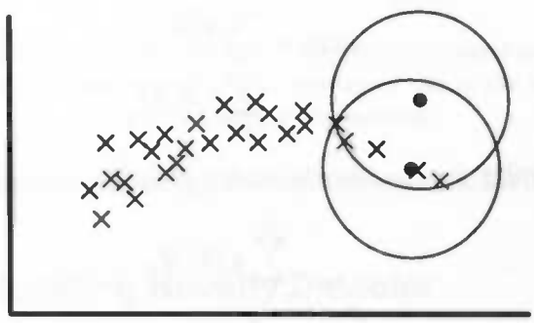

Figure 5.1: *novelty detection using k–NN with k=5. The same density is assigned to the outlying and to the inlying object.*

5.3 Gaussian mixture based novelty detection

5.3.1 Gaussian mixture estimation

In this paragraph we introduce the theory of the Gaussian mixture model which is a semi–parametric density estimating method that is optimized by an iterative algorithm. Since the derivation of this algorithm can be found on several places in the literature ([4], [9]) only the results are quoted here.

The Gaussian mixture model takes the form of

$$\hat{p}(\boldsymbol{x}) = \sum_{i=1}^{M} \alpha_i \phi_i(\boldsymbol{x}) \tag{5.1}$$

where α_i are the *mixing coefficients* that satisfy:

$$\sum_{i=1}^{M} \alpha_i = 1, \quad 0 \leq \alpha_i \leq 1 \tag{5.2}$$

and $\phi_i(\boldsymbol{x})$ is a d–dimensional Gaussian function:

$$\phi_i(\boldsymbol{x}) = \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_i)^T \Sigma_i^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_i) \right) . \tag{5.3}$$

The covariance matrix Σ_i, the center $\boldsymbol{\mu}_i$ and the coefficients α_i are the adaptive parameters of the model. Note that the model is a linear combination of Gaussian basis functions very much like an RBF network with Gaussian basis functions. In contrast to the RBF network are the values of the adaptive parameters determined by optimizing the log likelihood of the model over the training data:

$$log\mathcal{L} = \sum_{n=1}^{N} log\, \hat{p}(\boldsymbol{T}_n) \tag{5.4}$$

where $\boldsymbol{T}$ is the training data set consisting of N patterns.

The optimal forms of the free parameters are analytically determined by differentiation of the log likelihood. The obtained results, which are quoted here use the posterior probability of $\phi_i()$, which is defined using the Bayes theorem as:

$$q_i(\boldsymbol{x}) = \frac{\alpha_i \phi_i(\boldsymbol{x})}{\sum_{j=1}^{M} \alpha_i \phi_i(\boldsymbol{x})} \tag{5.5}$$

The parameters that optimize the log likelihood are given by

$$\hat{\boldsymbol{\mu}}_i = \frac{\sum_{n=1}^{N} q_i(\boldsymbol{T}_n)\boldsymbol{T}_n}{\sum_{n=1}^{N} q_i(\boldsymbol{T}_n)} \tag{5.6}$$

and

$$\hat{\Sigma}_i = \frac{\sum_{n=1}^{N} q_i(\boldsymbol{T}_n)(\boldsymbol{T}_n - \mu_j)(\boldsymbol{T}_n - \mu_i)^T}{\sum_{n=1}^{N} q_i(\boldsymbol{T}_n)} \tag{5.7}$$

while the mixing coefficients are given by

$$\hat{\alpha}_i = \frac{1}{N} \sum_{n=1}^{N} q_i(\boldsymbol{T}_n) \tag{5.8}$$

These optimization values can be found by using the *expectation–maximization (EM) algorithm* [9] or variants based on reinforcement learning [24], [42]. The optimal number of Gauss functions M is subject to cross validation.

5.3.2 Combining Gaussian mixture with RBF classification

By using the basis functions $\phi_i(x)$ obtained from the mixture optimization as the hidden units in an RBF network, a network is obtained that combines the classification task and novelty detection in a natural way [5]. The network has an output for every class plus one extra for the likelihood as shown by figure 5.2. The synaptic weights between the hidden layer and the likelihood output are given by α_i, while the weights to the class outputs are obtained by the usual linear techniques for finding the third layer weights for RBF networks.

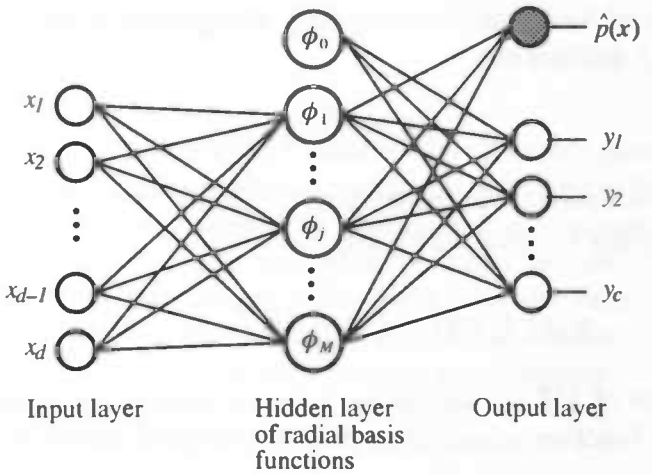

Figure 5.2: *A radial basis function network in which the basis functions are used both in the evaluation of the network outputs $y_j(x)$ and to provide an estimate $p\hat{}(x)$ of the likelihood of input data for the validation of the outputs.*

Note that the set of basis functions obtained from the mixture model is not necessary optimal for the classification.

5.3.3 Resource Allocating Novelty Detector

Roberts and Tarassenko present in [36] a variant of the method of above where the appropriate threshold level arises naturally from the training process. This is achieved by growing a Gaussian

mixture model during training. Once the network is fully trained, new data can be tested for novelty using the same threshold as was used to determine network growth during training. The network model is based on a Gaussian mixture model and uses reinforcement learning to solve the equations (5.6) and (5.7). The paper does not give the network a distinct name, here the network is named Resource Allocating Novelty Detector (RAND).

The network starts with a small number of basis functions, modelling the statistics of the training set $\boldsymbol{T}$ coarsely. As the learning procedure progresses the number of basis functions increases and the precision of the fit becomes better. The iterative equations are as follows:

$$\hat{\mu}_{i,t+1} = \frac{\hat{\mu}_{i,t} + \eta_t\left[q_i(\boldsymbol{T}_t)\boldsymbol{T}_t - \hat{\mu}_{i,t}\right]}{(1 - \eta_t) + \eta_t q_i(\boldsymbol{T}_t)} \tag{5.9}$$

$$\hat{\Sigma}_{i,t+1} = \frac{\hat{\Sigma}_{i,t} + \eta_t\left[q_i(\boldsymbol{T}_t)(\boldsymbol{T}_t - \hat{\mu}_{i,t})(\boldsymbol{T}_t - \hat{\mu}_{i,t})^T - \hat{\Sigma}_{i,t}\right]}{(1 - \eta_t) + \eta_t q_i(\boldsymbol{T}_t)} \tag{5.10}$$

where $\boldsymbol{T}_t$ is a randomly chosen data vector selected from the database $\boldsymbol{T}$ at the t–th iteration and η_t, $(0 \leq \eta_t \leq 1)$ is a learning parameter which is slowly reduced during training. The equations converge to equations (5.6) and (5.7) as $\eta_t \rightarrow 0$, which Roberts and Tarassenko show in their paper. While the objective of the network is the detection of novelty, assumptions about the distribution of the data are avoided by taking all the mixing coefficients α_i to be equal. After adding a basis function the coefficients are updated according to

$$\hat{\alpha}_{i,t} = \frac{1}{M_t} \tag{5.11}$$

where M_t is the number of basis functions at time t, The growth of the network is governed by the test parameter $\lambda(\boldsymbol{T}_t)$, defined as

$$\lambda(\boldsymbol{T}_t) = \max_{i=1}^{M_t}(\psi_i(\boldsymbol{T}_t)) \tag{5.12}$$

where $\boldsymbol{T}_t$ is the input vector presented at time t during training, M is the number of basis functions and $\psi_i()$ is a Gaussian basis function which satisfies $\psi_i(\hat{\mu}_i) = 1$:

$$\psi_i(\boldsymbol{x}) = \exp\left(-\frac{1}{2}(\boldsymbol{x} - \hat{\mu}_i)^T\hat{\Sigma}_i^{-1}(\boldsymbol{x} - \hat{\mu}_i)\right) \tag{5.13}$$

During training the value of $\lambda(\boldsymbol{T}_t)$ is used in the decision whether the network should grow with a single Gaussian basis function according to the following criterion:

$$\lambda(\boldsymbol{T}_t) \begin{cases} \leq \epsilon_t \rightarrow \textit{growth} \\ > \epsilon_t \rightarrow \textit{no growth} \end{cases} \tag{5.14}$$

where ϵ_t, $0 \leq \epsilon_t \leq 1$ is the growth threshold. Taking natural logarithms of equation (5.13) leads to a reformulated growth criterion of

$$\min\left((\boldsymbol{T}_t - \hat{\mu}_i)^T\hat{\Sigma}^{-1}(\boldsymbol{T}_t - \hat{\mu}_i)\right) \geq Q_t \tag{5.15}$$

where $Q_t = 2ln(1/\epsilon_t)$. The growth decision is thus based on monitoring the smallest Mahalanobis distance between $\boldsymbol{T}_t$ and each basis function center μ within the network.

After initialization of the network there are no basis functions present. The first presentation of a training vector causes the addition of a single function centered at this first vector. The training set is presented in random order and the first Gaussian function is adapted according to equations (5.9) and (5.10) until some $\lambda(\boldsymbol{T}_t)$ becomes equal to, or falls below, the ϵ_t threshold. At this point a new kernel function is generated.

The growth threshold is initially set as $\epsilon_0 = 0$, so growth will occur if some $\boldsymbol{T}_t$ has a 0% chance of having been generated by the network at that time. As the magnitude of ϵ_t increases linearly with time according to

$$\epsilon_t = \min\left(\epsilon_{\max}, \epsilon_{\max}\frac{t}{\tau}\right) \tag{5.16}$$

the fit of the model to the training data becomes progressively better.

A newly generated basis function, say with index n, is initialized to be centered at the training vector $\boldsymbol{T}_t$ that causes the growth, while the covariance matrix Σ_n is taken to be uniform and symmetric with each component Σ_{ii} being defined as

$$(\Sigma_n)_{ii} = \frac{1}{d}(\boldsymbol{C})_i \tag{5.17}$$

where $\boldsymbol{C} = (\mu_n - \mu_l)(\mu_n - \mu_l)^T$ in which l is the index of the basis function that, prior to growth, had the largest posterior $q_l(\boldsymbol{T}_t)$.

In their paper Roberts and Tarassenko also present two validation studies using 10 dimensional EEG data. In the first study a set of EEG data of a sleeping person was presented to the network. After training the network was capable to recognize EEG patterns of the person when awake. The network of the second study was trained using normal EEG patterns and was able to detect abnormal signal features caused by epileptics.

5.4 Discussion

Apart from methods for detecting novelties in data sets with known parametric distributions (like binominal and Poisson distributions) were the methods described in this chapter the only dedicated novelty detection methods found in the literature. In this paragraph the properties of the described methods are compared. The method using kernel estimation has for obvious reasons a computational complexity that is some orders higher than the other two. For this reason this method is omitted from this paragraph and only the Gaussian mixture in combination with a RBF network and the resource allocating novelty detector are compared.

5.4.1 Computational complexity

The Gaussian mixture model estimates the unconditional data density and labels a newly presented data vector to be novel when the density at that point is below a certain threshold value. In contrast, the resource allocating method, although based on Gaussian mixture estimation, identifies the borders beyond which vectors are labeled as novelties. Intuitively is an algorithm that identifies borders less computational complex than one estimating a density. In spite of this, the strength of the mixture method is that the basis functions of the Gaussian mixture are shared with the RBF classification task. The basis functions are evaluated for the classification task anyhow, so the extra load of the novelty detection is only the evaluation of one extra output node.

It may be clear that for classification problems that can be solved using a RBF network, the best way to detect novelty is the integration of a Gaussian mixture density estimator. However, the

computational complexity of the RBF network rises faster with the problem complexity than that of the MLP. For problems that for this (or any other) reason are solved using an MLP a separate novelty detector method is needed. The RAND is then better suitable, because it has a lower computational complexity than the Gaussian mixture estimator when not used integrated with an RBF network.

Using the resource allocating novelty detector the computational requirements can be reduced when for each distinctive class a novelty detector is created. After the classification task the novelty detector of the winning class determines the degree of novelty of the classified pattern. Especially for multi–class problems, the complexity of the network for one single class will be considerably lower than one network for the whole data set. For the combination of Gaussian mixture and a RBF network this approach will not be profitable, since all the basis functions must be evaluated to perform the classification task.

Further can the speed of the RAND algorithm evaluating a newly presented pattern be optimized in various ways. In the paper is mentioned that the covariance matrices Σ can be forced to be diagonal. Especially for high dimensional problems this leads to a considerable reduction in the computing time needed for the matrix inversions. Another way of speeding–up, not mentioned in the paper, is to keep the list of basis functions sorted according to the number of training patterns that fall inside the function. In this manner the functions that represent areas of the input domain with a high data density are evaluated first. When the newly presented pattern is non–novel, the algorithm can be stopped after finding the first basis function that encloses the vector and thus identifies the pattern as an inlier. Since the bulk of the newly presented pattern vectors is expected to lay inside the novelty boundary, keeping the basis functions sorted on their importance will raise the average speed of the algorithm.

5.4.2 Usage of heuristics

A Gaussian mixture novelty detector integrated with an RBF network uses more heuristically chosen parameter values than the resource allocating network. Firstly, there is the choice of the novelty threshold which for the latter method follows from the training phase. Secondly, the number of required basis functions for the mixture estimation is subject to cross validation, while for the RAND this number is determined during training.

Chapter 6 Experiments

6.1 Introduction

6.1.1 Aim

In the last chapter we found on theoretical grounds that the RAND a suitable method for novelty detection in complex pattern recognition problems. These findings were tested in a series of explorative experiments. The uncovered strenghts and weaknesses of the method were used to make a list of properties the ideal novelty detector must have. In order to make this list complete a kernel density estimator with a novelty threshold was also incorporated in the experiments.

6.1.2 Used data set

The data for the experiments was taken from the Satimage database, which was used for the Elena benchmark study [1], [6]. This database contains spectral information of seven types earth surface. The data was generated from multi–spectral scanner images from the Landsat satellite. For these experiments the 36 dimensional database was reduced to two dimensions by taking the two first principal components. From this reduced set the classes 1 and 7 were selected to obtain a data set of 3041 patterns with a great variety in data density. This set was halved randomly into a training and a test set. The training set was used to generate the novelty detectors, the test set was only used for evaluation of the obtained detectors. The sole reason for reducing the original database to two dimensions and two classes is to visualize the properties of the generated novelty boundaries. It is needless to say that the presented results cannot be compared with other experiments using the Satimage database. For these experiments is such a comparison beyond the aim.

6.1.3 Novelty boundary evaluation

A problem with designing a novelty detector for a classification problem is the absence of knowledge about the novel data. This makes it hard to evaluate the quality of a novelty detector. Overfitting (small local characteristics of the data near the novelty boundary influence the shape of the boundary too much) can be detected using a test set. A test set contains data drawn from the same random process as the training set data was drawn. The performance on the test set or the acceptation rate of the test set is the percentage of the test set that accepted by the novelty detector, e.g. that is recognized to be similar to the training set data. The occurrence of

underfitting, e.g. the novelty region is too large, is much harder to detect. In the experiments special attention is given to ways in which the novelty boundary quality can be evaluated and parameter settings can be optimized.

6.2 Kernel estimation experiment

6.2.1 Description

A density model of the data was made using kernel estimation. The smoothing parameter h was set to the average distance of the ten nearest neighbors from each point in the training set, averaged over the whole training set. For most data sets this rule of thumb gives a h close to the one obtained using cross validation [5]. The novelty threshold was then determined iterative to accept $98\% \pm 0.1$.

6.2.2 Results

The novelty boundary this method produces is together with the training data shown in figure 6.1. As described above does this method always reject a part of the training set, in the shown case 1.91%. The accepted part of the test set is 95.5%. From the figure it is clear that overfitting of the data occurs. Especially at areas with a relatively low gradient in the data density follows the border local data characteristics (for the upper cluster compare the upper and the lower boundary). A significant part of the discrepancy between the two accept rates shall be caused by this. Increasing the initial value of h will smooth the density model, but at high gradient regions the novelty boundary becomes loose fitted.

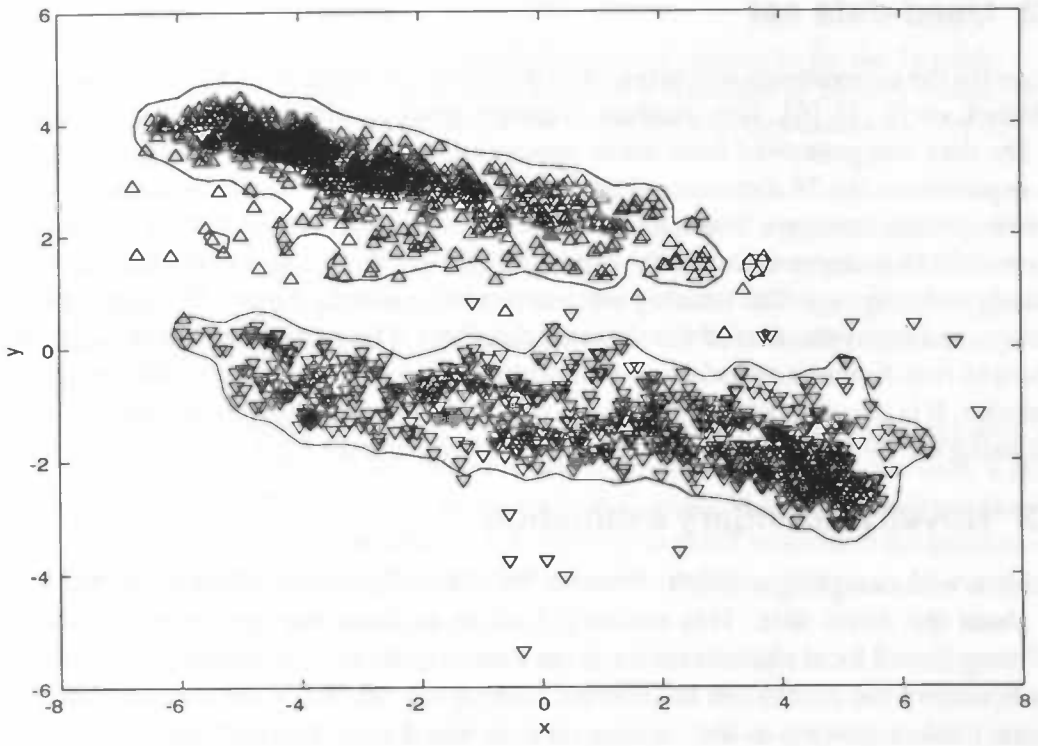

Figure 6.1: *Novelty boundary of a kernel estimation based novelty detector shown with the training data. The novelty threshold is set to accept 98% of the training data.*

6.2.3 Conclusion

Drawback of all novelty detectors based on thresholding a density model, is the unavoidable rejection of a fraction of the training set. In order to determine a novelty threshold part of the training data must be closed out.

Kernel estimation based novelty detection reacts different on data regions with different local gradient in the density. High gradients become a loose fit, low gradients become a tight fit. For the ideal novelty detector this should be the other way around.

Further is the computation of a full model of the data density as means to establish a yes/no answer overkill.

6.3 RAND experiment

6.3.1 Description

The drawbacks of the density estimation method from above are partly solved in the RAND architecture. Firstly, all training data lies inside the established novelty boundary and, secondly, the impact of data density characteristics on the training process are reduced. The latter is done by treating all basis functions with equal importance. The mixing coefficients α_i do not reflect the number of data points represented by the corresponding basis function, as is done in real mixture estimation. The coefficients are kept equal instead. Further is the evaluation of the model done by computing Mahalanobis distances instead of a data density.

For this experiment a simplified version of the RAND was implemented. The simplified method only uses spherical basis functions. After trials to set the parameters τ_ϵ, τ_η and η_0 , six groups of four networks were trained. Over the six trials all parameters, but the ϵ_{max}, were constant. Per trial of four all parameters were constant. The learning process stopped when for all basis functions the ϵ reached ϵ_{max}. For basis functions that represent only few data points in areas of low data density the ϵ goes very slowly to the maximum. Therefore the learning process was stopped after 500 epochs. Since the bulk of the basis functions reach their ϵ_{max} much earlier this does not influence the result very much.

In order to monitor the fit of the model over the trials the volume of the accepted area is estimated. The exact volume is hard to determine, since it involves computing the volume of sphere intersections. Therefore the volume is estimated by the sum of all basis function volumes. This estimation is too high since the sphere intersections are represented twice.

6.3.2 Choice of parameters

The network has four parameters that need setting: ϵ_{max},τ_ϵ, η_0 and τ_η. The first two parameters define the linear rising of ϵ, the latter two the exponential fall of η. The number of basis functions after training is depending on the novelty threshold ϵ_{max}. The chance of overfitting is expected to increase with the value of it. Roberts and Tarassenko claim that the setting of the other three parameters is not critical as long as the rate of increase of ϵ is small compared to the initial rate of decrease of η. In other words the adaptivity of a basis function decreases fast after creation. Note that this is a way to limit the influence of the data density on the resulting novelty boundary.

6.3.3 Results

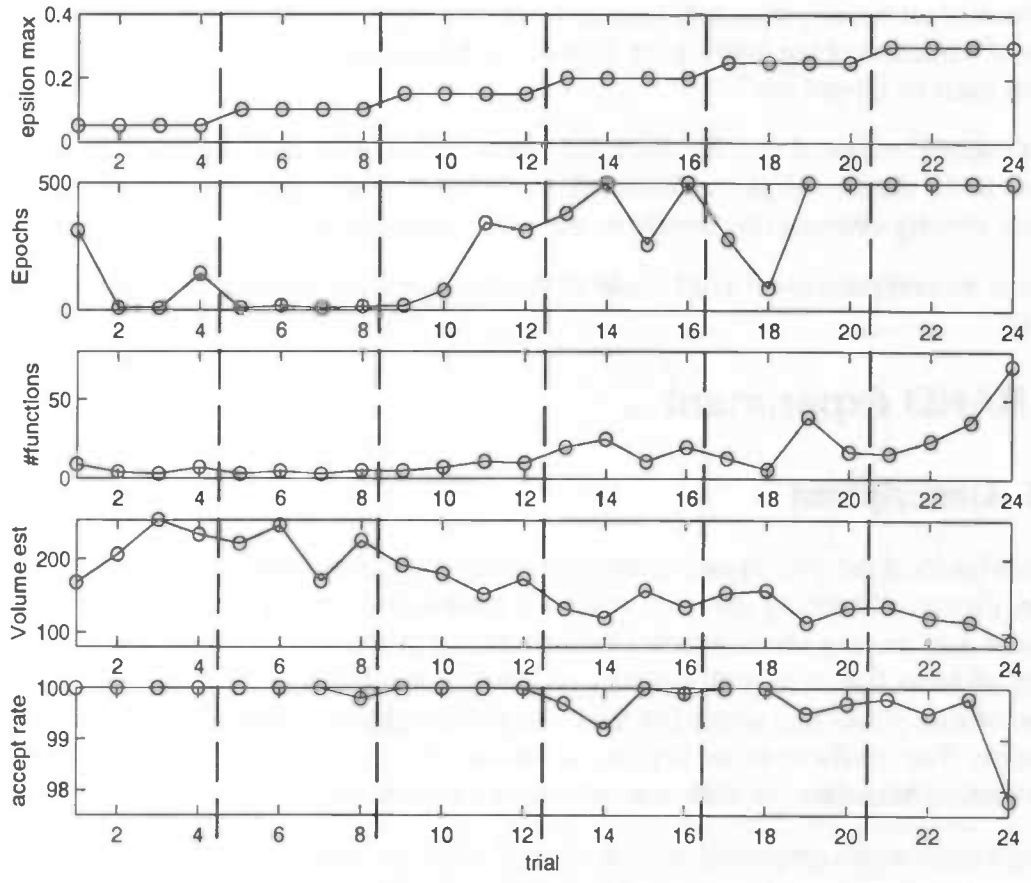

Figure 6.2: *RAND experiment, learning parameter and network properties for the 24 trials: (1) value for the parameter ϵ_{max}, (2) number of needed epochs with a maximum of 500, (3) number of basis functions after learning, (4) estimation of the volume inside the novelty boundary, (5) accept rate of the test set.*

Figure 6.2 lists the results of the trials. Every four trials the value of ϵ_{max} was increased from 0.05 in the first to 0.3 in the last quartet. With the rise of ϵ_{max} the learn time and the number of basis functions rise as well. According to the volume measure the fit becomes progressively better with the rise of ϵ_{max}. Most networks score a 100% accepted on the test set, starting from trial 13 relatively small overfitting occurs for 9 of the 12 networks.

On basis of the data in figure 6.2 network 11 has the best properties: of the networks with a 100% acceptation rate on the test set it is the one with the smallest volume together with a relatively small number (11) of basis functions. A "quick look test" of all 24 figures confirms that for network 11 the falsely accepted area outside the data domain is small relative to the number of basis functions. The novelty boundary of this network is shown in figure 6.3.

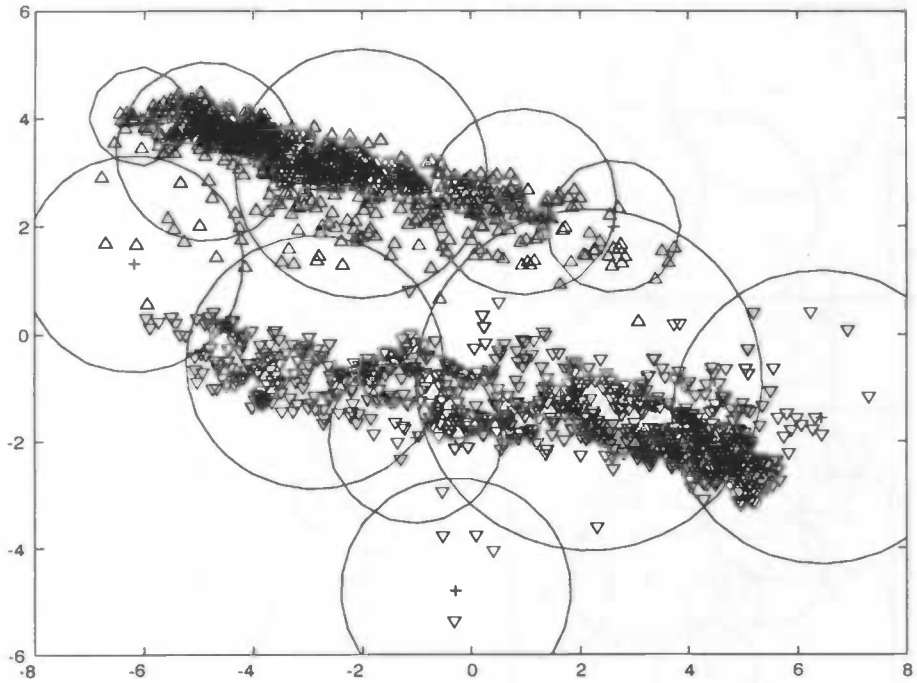

Figure 6.3: *Resulting novelty boundary for trial 11 with the training data.*

A notable phenomenon in figure 6.2 is non–smoothness of graphs. Even when the training parameters are kept equal the network properties are varying greatly. Figure 6.4 shows the number of needed basis functions for the third quartet on a different scale. Note that in spite of equal parameters the highest number of functions is twice the lowest. This discrepancy is caused by the random order in which the training data is presented during training. Figure 6.5 gives the basis functions of the networks of the third quartet. It is clear that the solutions differ greatly.

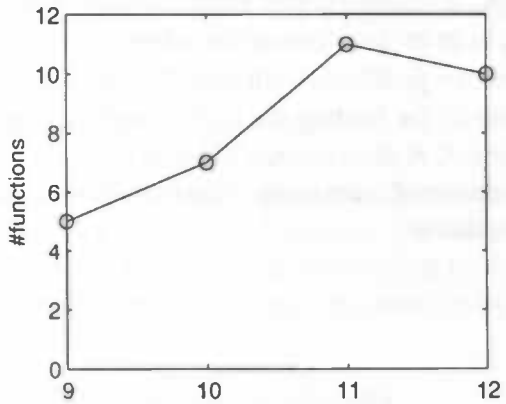

Figure 6.4: *The number of basis functions for the trials with* ϵ_{max}*=0.15.*

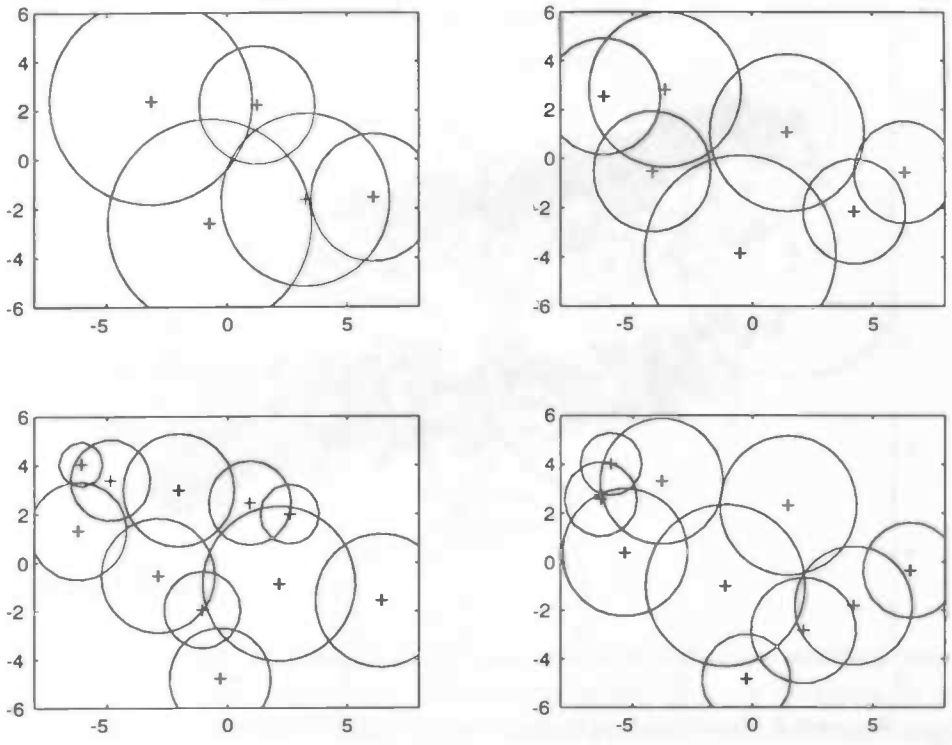

Figure 6.5: *The impact of random data order in the training: basis functions of four networks trained with the same parameters, but with different results.*

The impact of the training data order increases with the $\epsilon_{\max}$. Basis functions generated early in the training process start with a greater width of their Gauss functions than functions generated later. Since the adaptivity of the functions is limited these functions are likely to remain their greater width until the end of the training process and thus gain a greater influence region. Figure 6.6 shows an example of this. A change in the values for the initial learning parameter η_0 and the time constants τ_ϵ and τ_η in order to increase the adaptivity of the basis functions may overcome this problem. There are two problems with this. Firstly, it increases the number of parameters for which the value is critical for finding the right solution. And secondly, the occurrence of the phenomenon is hard to detect. A discrepancy in basis function influence is determinable, but whether such a discrepancy is wanted is far more difficult to investigate, especially when working in multi–dimensional input spaces.

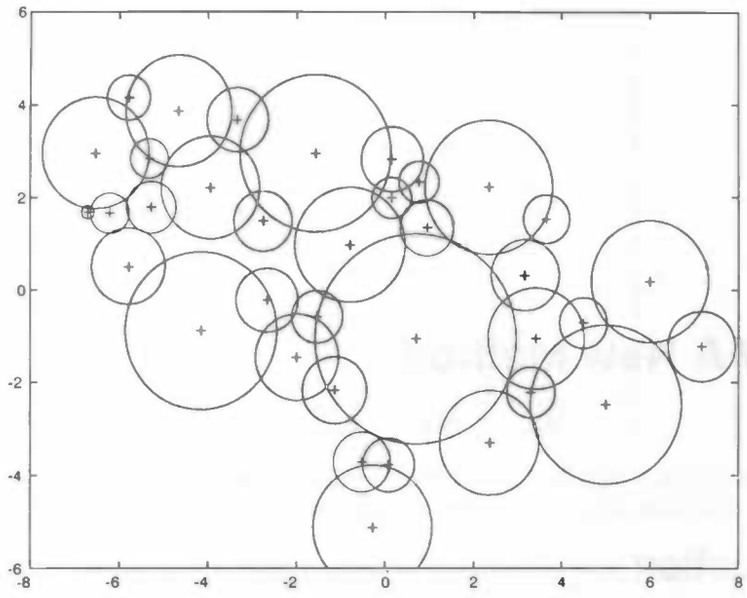

Figure 6.6: *The basis functions for network 23*

6.3.4 Conclusion

The resource allocating novelty detector has four parameters for which the optimal combination needs to be determined. This number is high when you take into account the number of measures that give the quality of the generated novelty boundary. As a result is the optimal combination hard to find.

Further is the parameter optimizing process disturbed by the high impact of random influences. The resulting novelty boundary is to a high extend depending of the training data order. Training different networks with the same parameters yield a variety of solutions among which the fit of the data varies.

6.4 Design constraints

From the above experiments two important conclusions can be drawn. Firstly, too much modelling of the data density results in a model of which only part of important for the purpose. And secondly, the optimal setting of the training parameters must be determinable. Using these conclusions we formed a list of constraints, which must be satisfied by a good novelty detection method:

- Model the novelty boundary, not the data density.
- Low number of parameters.
- The quality of the parameter setting must be testable.

Chapter 7 A New Method

7.1 Introduction

Using the design constraints of paragraph 6.4 a new novelty detection method was developed. This chapter first describes the algorithm. After that a comparison with the methods used in chapter 6 is done using 2D data. At last the method is used for a licence plate number recognition problem.

7.1.1 Constraints

The method is developed to meet the constraints stated in paragraph 6.4:

- Model the novelty boundary, not the data density.
- Low number of parameters.
- The quality of the parameter setting must be testable.

To meet the first constraint all operations whose results are influenced by data density (like mean and variance) are avoided. Only (Mahalanobis) distances between points and operations like min and max are used. The method uses two parameters that are closely correlated with measurable quantities. So the latter two constraints are met as well.

7.1.2 Mahalanobis distances

Similar to the resource allocating novelty detector this new method models the data domain by a combination of basis functions defining hyper–ellipsoids in pattern space. These ellipsoids are described by their center μ, a d–by–d–matrix Σ, and a threshold Q, where d is the dimension of the input space. A given data point x falls inside an ellipsoid when the Mahalanobis distance from the point to the center is less or equal Q:

$$(x - \mu_i)^T \Sigma^{-1} (x - \mu_i) \leq Q \tag{7.1}$$

The Mahalanobis distance can be seen as a weighted distance measure, as shown in figure 7.1. The matrix Σ is often called the covariance matrix, since then the matrix is used to describe a normal data distribution. Since here we want to avoid the use of variances we use the matrix just for describing an ellipsoid.

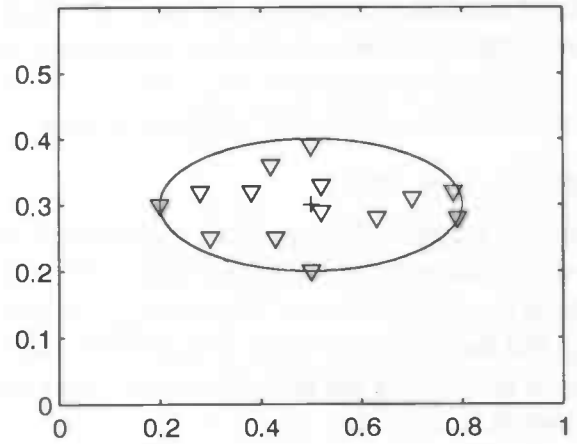

Figure 7.1: *Mahalanobis distance based novelty criterion: the euclidian distance to the center (+) is weighted heavier in the direction of y than in the direction of x.*

If the axes of an ellipsoid are kept parallel with the axes of the pattern space, then the matrix Σ will be diagonal. For multi–dimensional problems this restriction results in faster training and evaluation. Therefore we describe our new method for diagonal matrices. At the end will be discussed how the method can be generalized to use full covariance matrices.

7.2 Training process

7.2.1 Outline

The training process consists of three parts:

- Data clustering
- Minimal coverage of clusters
- Threshold adjustment

7.2.2 Data clustering

The data clustering identifies groups of data points with small mutual Mahalanobis distances. The maximum mutual distance of points in one cluster is set via the grid parameter $\gamma \geq 0$. The clustering is done in three steps:

- Choose initial basis function region
- Cover data domain with basis functions
- Nearest center classification

The initial basis function shape is chosen to reflect the variation in the data along the pattern space axes. The width of the initial ellipsoid in the dimension i is chosen to be:

$$(w)_i = \gamma\left(\max_{n=1}^{N}\left((\boldsymbol{T}_n)_i\right) - \min_{n=0}^{N}\left((\boldsymbol{T}_n)_i\right)\right) \tag{7.2}$$

where $(\boldsymbol{T}_n)_i$ is the i–th element of the n–th pattern in the database $\boldsymbol{T}$ consisting of N patterns. Thus the width of the ellipsoid in every dimensional direction is determined by computing the mini-

mum and maximum value of the data projected on the corresponding axis, taking the difference and multiplying by γ. The diagonal of the matrix Σ_{init} is filled according to:

$$(\Sigma_{init})_{ii} = \frac{1}{(w)_i^2} \tag{7.3}$$

the non–diagonal elements are set to zero.

After choosing the initial basis function the data domain is covered with these functions. This is done by testing for every data point in the training set whether the point lies inside any of the basis functions. If not, a new basis function is created. The center of the new function is set to the data point, the Σ is set to Σ_{init} and the Q is set to unity. The process starts with no basis functions at all, thus the first function is created for the first data point. After the process all training data points are covered as shown in figure 7.2.

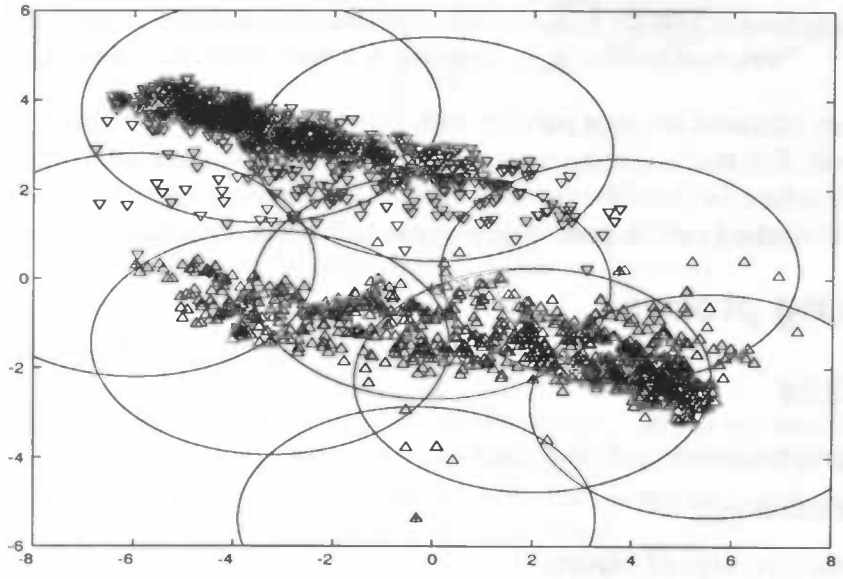

Figure 7.2: *Data clustering by data covering: data points with the same nearest center are clustered together. The grid parameter γ is in this example set to 0.5.*

All training data are covered, but a great part of the points are covered by more than one basis function. To obtain an unambiguous clustering of the data points, a nearest center classification is performed. Each point is associated with it's nearest basis function center. Data points with the same nearest center form a cluster.

Note that when γ is set to zero each data point at itself forms a cluster. After exceeding a certain value for γ all data points are grouped in one cluster. For data sets with a certain smoothness in their data domain border this point is reached around the value of one. Especially for multi–dimensional data this value can be higher.

7.2.3 Minimal cluster coverage

The second step of the training process determines for every cluster the smallest ellipsoid that covers all cluster data. The center of each basis function is in every dimension i shifted according to:

$$(\mu_j)_i = 0.5 * \left(\min_{n=1}^{N_j} \left((C_{j,n})_i \right) + \max_{n=0}^{N_j} \left((C_{j,n})_i \right) \right) \tag{7.4}$$

where $(C_{j,n})_i$ is the i–th element of the n–th data point of cluster C_j and N_j is the number of points in cluster C_j. In other words: the center is shifted to the middle of the bounding box of the cluster data. where a bounding box is defined as the smallest box that encloses a data set with the edges of the box parallel to the dimension axes. The ellipsoid is taken to be proportional with the bounding box:

$$(w_j)_i = 0.5 * \left(\min_{n=1}^{N_j} \left((C_{j,n})_i \right) - \max_{n=0}^{N_j} \left((C_{j,n})_i \right) \right) \tag{7.5}$$

The diagonal of Σ is set according:

$$(\Sigma_j)_{ii} = \frac{1}{(w_j)_i^2} \tag{7.6}$$

and the other elements are set to zero. The result is the biggest ellipsoid that fits into the bounding box. Since such an ellipsoid does not necessarily cover all the cluster data the matrix is normalized to enclose the cluster member with the highest Mahalanobis distance. The matrix must be changed in order to make the Mahalanobis distance of this furthest cluster member equal to Q, which is still set to unity:

$$D_j^{\max} = \max_{n=1}^{N_j} \left((C_{j,n} - \mu_j)^T \Sigma_j^{-1} (C_{j,n} - \mu_j) \right) \tag{7.7}$$

$$\Sigma_j = \frac{\Sigma_j}{D_j^{\max}} \tag{7.8}$$

Figure 7.3 shows the result of the second step on the basis functions of figure 7.2.

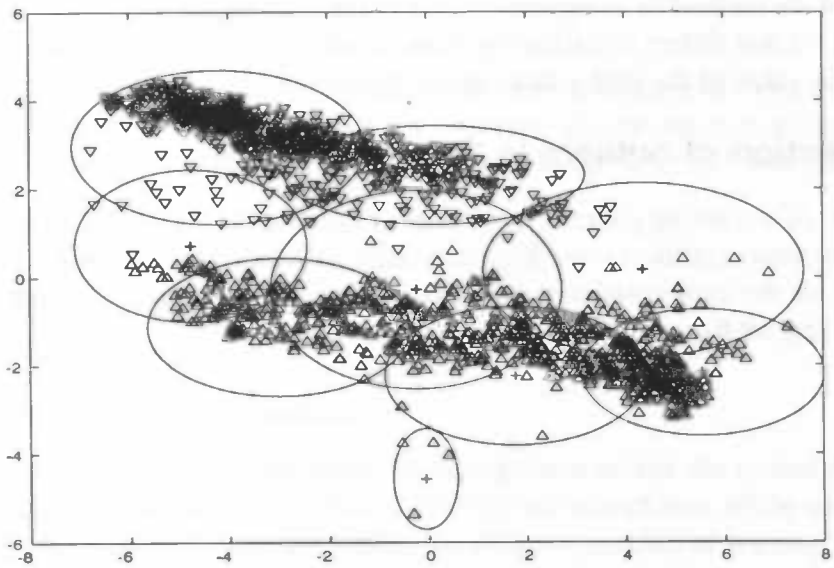

Figure 7.3: *Novelty boundary after the minimal coverage of clusters.*

7.2.4 Threshold adjustment

After the second step a novelty boundary is obtained with a minimal ellipsoid coverage of the clusters of step one. As a result of the minimal coverage there are training patterns lying on the novelty boundary (see figure 7.3). This can be seen as local overfitting, which is reflected in the

performance of the model on the test set. By increasing the threshold Q the fit can be loosened to overcome this problem. The effect of increasing Q is shown in figure 7.4.

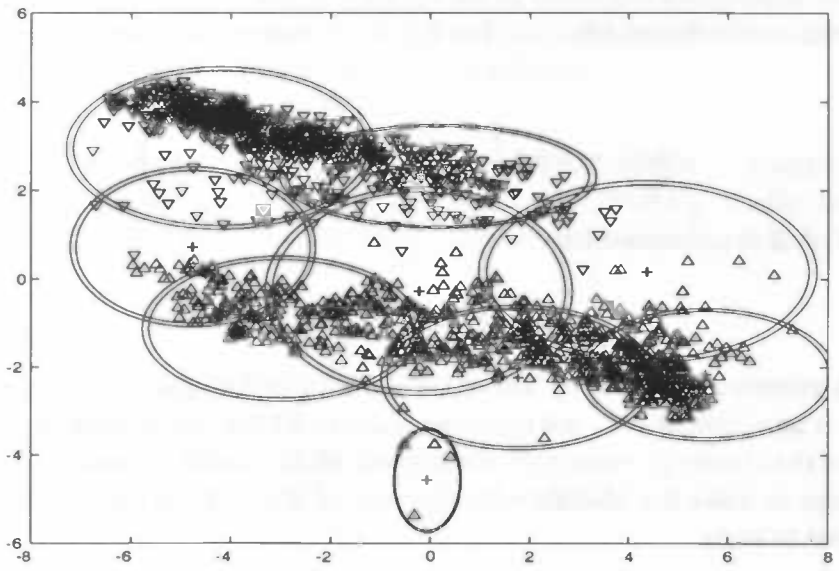

Figure 7.4: *Increasing the threshold Q to loosen the fit of the model. The Q was increased from 1.0 (the inner ellipsoids) to 1.1 (the outer ellipsoids).*

7.3 Parameter behavior

Before we discuss the behavior of the two process parameters γ and Q we want to pay attention to the ability of the method to recognize outliers in the training set. An outlier is defined as a data point in a data set that differs significantly from the other data in the set. Since this phenomenon is related to the value of the grid γ this issue is discussed here.

7.3.1 Detection of outliers in the training set

For low values of γ training patterns may form a cluster at themselves. The distance from such a point to it's nearest neighbors is too high in relation to the size of the initial basis function. Such a point is thus for the used value of γ recognized as an outlier. Figure 7.5 shows for the reduced Satimage training set the occurrence of an outlier.

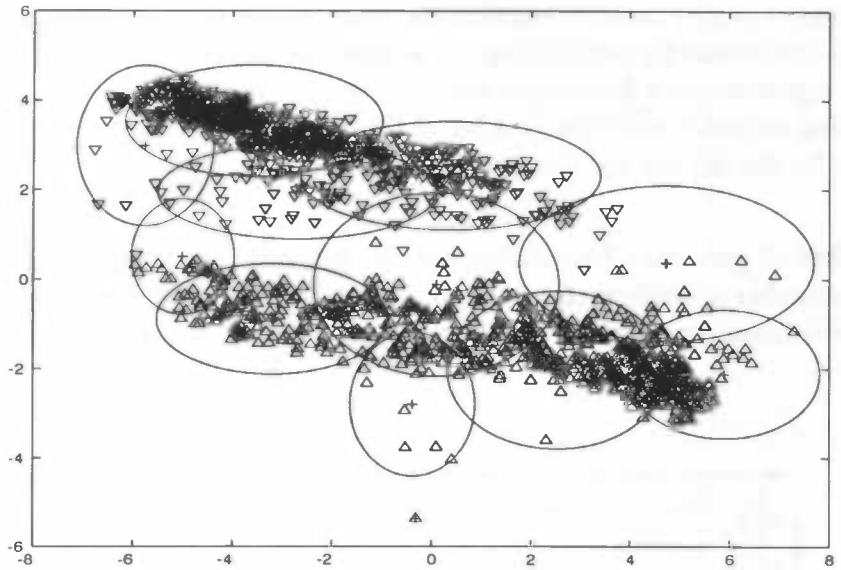

Figure 7.5: *Training pattern labeled as an outlier. The data point at the middle–low of the figure was the only member of it's cluster. Apparently the distance to the points above it was to large for this particular value of γ (=0.06).*

The actions taken when training patterns are labeled outliers depends on the confidence one has in the data collection method. If the confidence is high, then the patterns are falsely recognized as outliers and the grid parameter must be increased. If the confidence in the data collection is low, then the outlier patterns must be checked whether they truly belong to the set. If they are by mistake in the set they can be omitted, otherwise the grid parameter is set too low.

7.3.2 Quality measures

The method has four important quality measures: the number of used basis functions, the accept rate on the test set, the volume inside the novelty boundary and the number of found outliers. Using these measures the generated novelty detectors can be judged. A good detector has the following properties:

- High test set accept rate
- Low region volume
- Nihil number outliers
- Acceptable number basis functions

The first two are the most important since balancing them solves the underfitting/overfitting problem. Overfitting is reflected in a relative low test set accept rate, while underfitting is reflected in a high volume of the accepted area. Keeping the number of training set outliers low ensures that area's of the data domain with low data density fall inside the novelty boundary. The last property results in a low evaluation times.

7.3.3 Grid parameter γ

The grid parameter γ has the highest impact on the quality of the generated novelty boundary: it has a direct impact on all four quality measures. In contrast: the threshold Q is used to fine tune the test set accept rate via the accept region volume.

Figure 7.6 plots the quality measures against the value of γ for the 2D Satimage data. The volume of a ellipsoid is estimated by multiplying it's widths. The volume is thus estimated too high. As expected, the figure shows a decrease in the number of needed basis functions with the increase of γ. Overfitting occurred when the number of functions is high, which is reflected in a low acceptation rate for the test set and the recognition of outliers.

A great number of generated boundaries has the properties stated in paragraph 7.3.2. From γ=0.07 is the number of outliers equal to zero together with good figures for the test set performance and the volume. This stays more or less the same until the number of basis functions reaches one at γ=0.4.

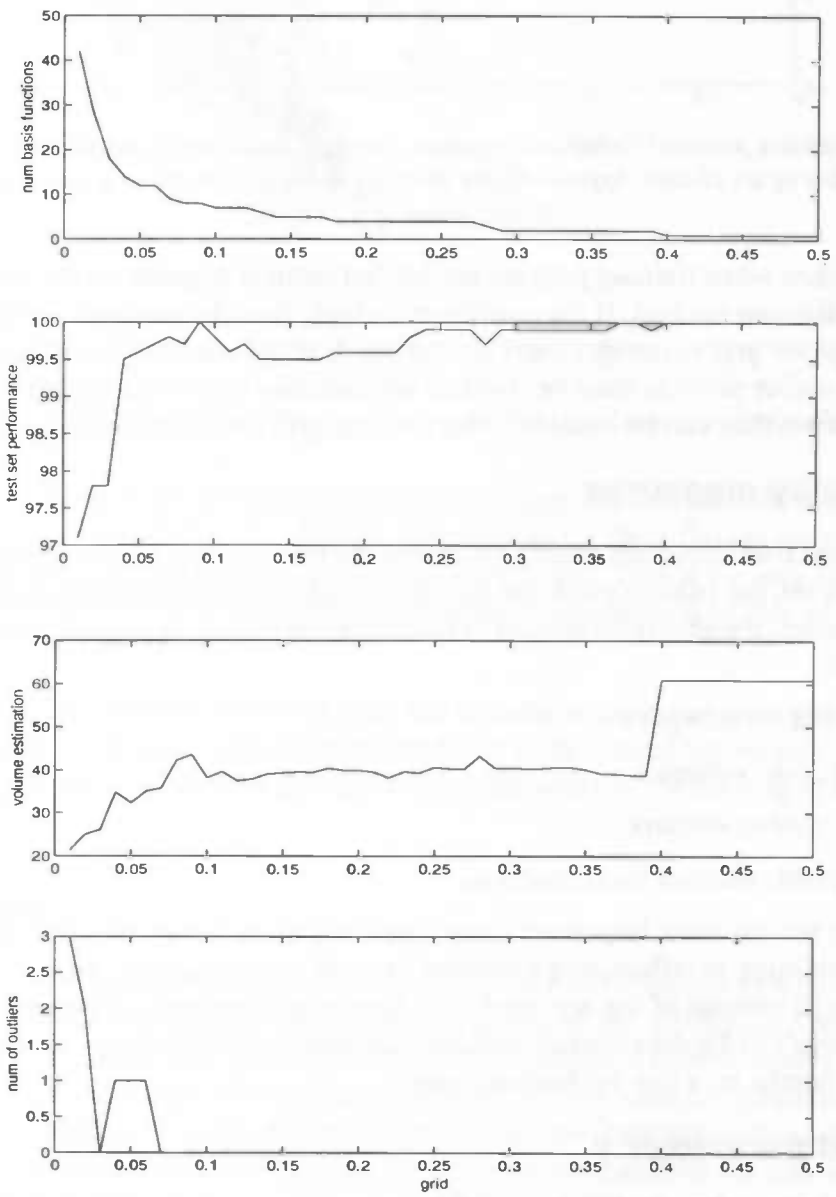

Figure 7.6: *Four quality measures against the grid parameter γ for the 2D Satimage data, the threshold Q is kept equal to 1.*

7.3.4 Threshold Q

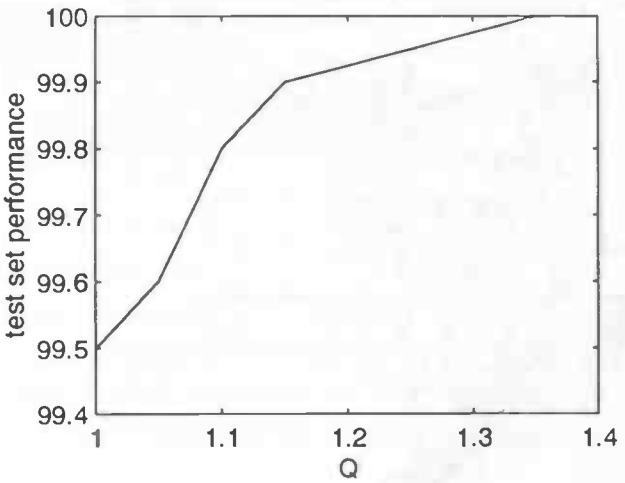

Figure 7.7: *The impact of increasing the threshold Q on the accept rate of the test set. The grid was set to 0.15. For the shown range the volume estimation kept stable on 39.3.*

The Q parameter is used to fine tune the fit of the novelty boundary. It's value is standard set to unity and is increased in order to raise the performance on the test set.

7.4 Comparison experiment

7.4.1 Used data and performance measures

To test the new method against the RAND an experiment was done. The comparison of the methods was carried out using 2D Satimage data. The training and test sets where those described in paragraph 6.1.2 and used in the examples of above. The training set is used to model the novelty boundaries, the test set is used to evaluate the models in order to optimize the parameter settings. Two data classes of the original Satimage database, unused for the test and training set, were selected to form two novelty sets. The novelty sets were not used in the design process, just in the comparison of the different methods. Both methods used only spherical basis functions during this experiment.

The accept rates on the novelty sets are used as performance measure in the comparison. The accept rate is the percentage of patterns in a given set that are accepted as valid, thus not being novel. So a good novelty detector has a high accept rate on the test set and a low accept rate on a novelty set.

The data of the two novelty sets are shown in figure 7.8. The sets are chosen because they both overlap the training data, but in a different manner. Novelty set 1 (the x–es) overlaps the area of low density in the lower–middle of the training data. Novelty set 2 overlaps the training data at an area with an higher gradient in the data density, while the bulk of the novel patterns is just outside the region of the training data. So a looser fit in the novelty boundary at that place is highly reflected in the accept rate of this novelty set.

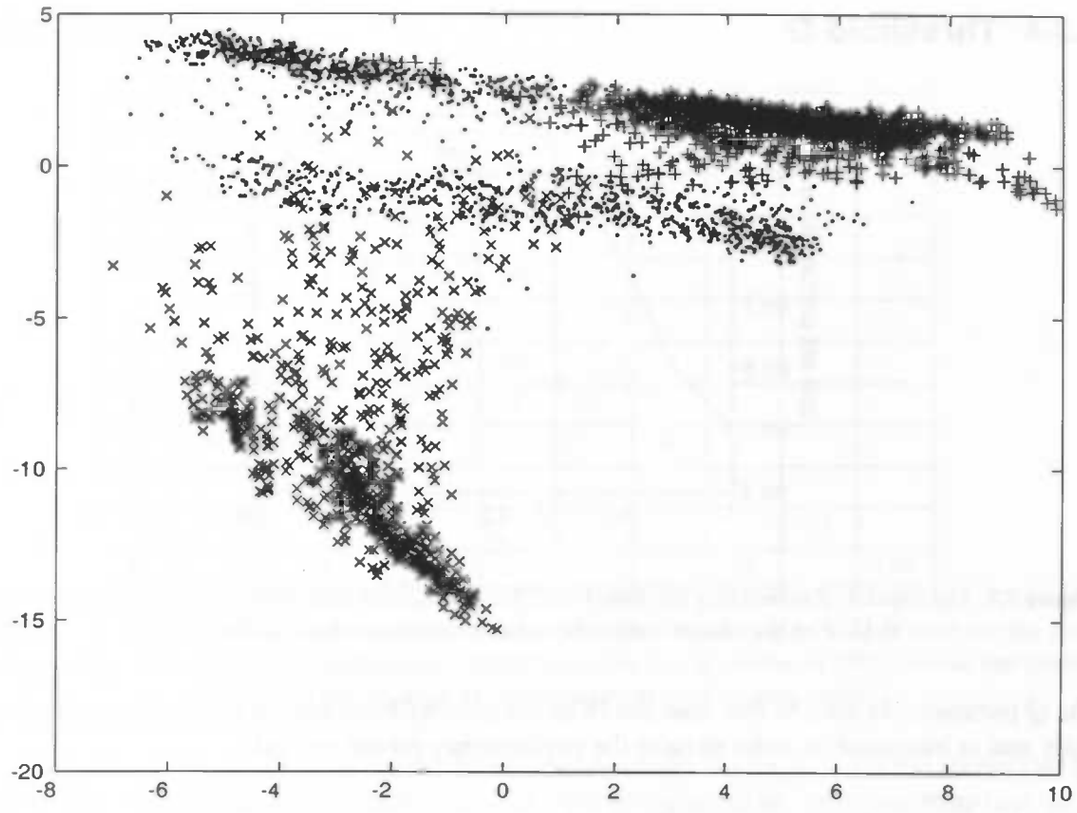

Figure 7.8: *Two classes of novel data (+, x) added to the data used to train the novelty detectors (.).*

Because of the overlap of the novelty sets with the training and test data the accept rates of the novelty sets will never reach 0%. To obtain a lower bound for these figures the novelty detector based on kernel estimation is incorporated in this experiment. This is the sole purpose of using the kernel estimation, since it's needless to say that the computational burden of a kernel model is too heavy in complex problems.

For the creation of the novelty detectors only the training and the test set together with all the design means of the particular methods were used. For each method ten versions were built and optimized independently. For every detector version the order of the training data was changed randomly.

7.4.2 Results

Table 7.1 shows the results of the experiment. For both the RAND and the new method the accept rates are shown for the version with the lowest accept rate on the novelty set 1 and the one with the lowest figure for the novelty set 2. It is clear that the proposed method scores over–all better on the novelty set accept rates. The training data fit of the new method is tighter, while the test set accept rates are similar. Another difference between the two methods is the score on the novelty set for which the detector is non–optimal. For the proposed method the non–optimal figures differ only slightly: 1.1% in both cases. For the RAND this difference is higher: 3.7% and 25.2% respectively. Thus if the new novelty detector scores good results on one novelty set, it scores also good results on the other.

Method	Accept rate		
	Test set	Novel set 1	Novel set 2
Kernel	95.5%	5.4%	10.2%
RAND (best of 10)			
Optimal for novel set 1	100%	14.2%	66.7%
Optimal for novel set 2	100%	17.9%	41.5%
New Method (best of 10)			
Optimal for novel set 1	99.9%	9.0%	34.2%
Optimal for novel set 2	99.9%	10.1%	33.1%

Table 7.1: *Results of the comparison experiment.*

7.4.3 Conclusion

Compared to the other tested methods the novelty boundary of the proposed method has a tighter fit round the training data without the occurrence of overfitting.

Another good property of the new method is the consistency of the novelty boundary quality. The quality of a local part of the novelty boundary appears to be related to the quality of another part of the boundary.

The last conclusion provides a design tool for novelty detection in classification problems. When a novelty detector is created for each class, then the data of the other classes can be used to assess local quality of the novelty boundary.

7.5 OCR Experiment

7.5.1 Licence plate numbers

To investigate the usability of the new method for a multi–dimensional problem an optical character recognition experiment was carried out. The data used for this experiment come from a database of licence plate characters used for the development of a car licence plate recognition system by the University of Groningen [28]. The character images are cut out of video images of cars driving on a highway by a segmentation process. These images are input to a feature extraction process which outputs 24 features.

From this database the numeric character classes were selected and divided into a training and a test set. Each of the sets contains 100 feature patterns for every number, totally 1000 patterns per set. Using these sets 10 novelty detectors were developed, one for each number class. For training only the 100 patterns in the training set belonging to the particular number were used. For the evaluation of the parameter settings all the data in the test database was used. The accept rate on the test set is determined for all it's ten number classes. The parameters are selected to give a high accept rate for the number class the novelty detector is designed for and low rates for the other classes.

7.5.2 Results

	Novelty detector									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**0**
1	*92*						*1*			
2		*96*	*2*							
3		*1*	*100*		*18*					
4				*96*						
5					*96*					
6					*2*	*99*				
7	*40*	*3*					*99*			
8								*98*		
9									*99*	
0								*14*		*100*

Table 7.2: *Accept rates on the test set. Each column gives for the particular novelty detector the accept rate per class. An empty cell means a zero accept rate. The test set contains 100 patterns per class, so the figures can be seen as percentages as well as cardinalities.*

Table 7.2 shows the accept rates for the 10 novelty detectors. As expected is the accept rate high for the 'own' test patterns and (nearly) zero for most of the novel test patterns. Note that the aim is not to build a classifier that separates the classes. For that purpose we have dedicated methods like the MLP. An advantage of a dedicated novelty detector separate of the classifier is that the detection process does not disturb the classification. From the novelty detection point of view is it right to accept part of the sevens as being similar to the ones. Separating the classes seven and one must be done by the classifier, the novelty detector must detect inputs that are significantly different from it's training data. In all cases for which the accept rate on a novel class is greater than zero there is some geometrical similarity between the novel characters and the characters the detector is trained for.

Figure 7.9 shows the seven from the test database that is for the human eye closest to a one. When the parameters of the novelty detector of the ones are set in such a way that it finds only one seven similar to a one, it finds the shown seven. This indicates that similarity to the human eye can be modeled by a novelty detector based on Mahalanobis distances.

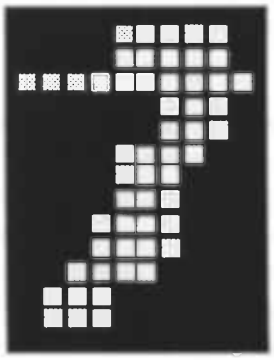

Figure 7.9: *Input character image of a 7.*

Novelty detector	1	2	3	4	5	6	7	8	9	0
Grid	7	4.8	3.5	5.4	5	5.5	4.5	4.75	2	3
Q	1	1.5	1.17	2	2	2	1.25	1.05	2.1	1.5
Number of basis functions	1	2	4	2	1	1	2	2	3	2

Table 7.3: *Parameters and number of basis functions for the results of table 7.2.*

Table 7.3 shows the parameters for each detector and the number of basis functions. The number of basis functions is low or equal to four in all cases. Some novelty boundaries are even modelled by one ellipsoid. This low number is a result of the high dimensionality in combination with the small size of the training sets. Using more data is expected to result in more basis functions and better accept rates on the test and novelty sets.

For the class three in the training set is searched for the first feature pattern that is labeled an outlier when the gamma is decreased. The image belonging to that feature pattern is shown in figure 7.10 together with nine images of 'normal' threes. Apart from the difference in size of the characters, which is undone by the feature extraction, the outlying three has also to the human eye different geometric characteristics. This also indicates that the concept of a weighted distance measure like the Mahalanobis distance is a way to model the human perception of similarity. Of course is a good feature extraction a prior condition for these results.

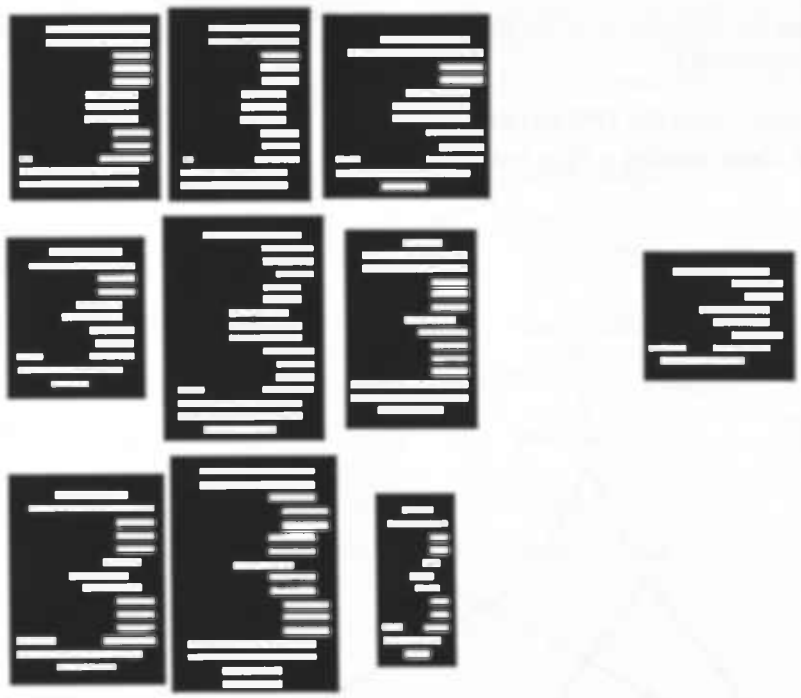

Figure 7.10: *Nine randomly chosen threes form the training set together with the first three labeled as outlier when the grid γ is decreased.*

7.6 Conclusions

The proposed method meets the design constraints stated in paragraph 6.4 and therefore it has the following properties:

- No unnecessary modeling of the data density.
- The quality of the parameter setting is testable

Because of the first property the training time is low: the training process ends after three passes through the training data. This, in combination with the second property makes the parameter optimization a fast process. Further is the model complexity low, which results in fast model evaluation.

First experiments show that:

- Compared to other methods results the method in a tighter fit round the training data without overfitting it.
- The novelty boundary is consistent: local boundary quality is a measure for the quality of the whole boundary.
- The method is suitable for complex pattern recognition problems like optical character recognition.
- There are indications that the method's model of similarity resembles the human perception of similarity.

These results come from the first experiments with the method. More experimentation needs to be done to give these results a firm basis.

Chapter 8 Conclusions

8.1 General Conclusions

The main question this research tried to answer, as stated in paragraph 1.3.1, was:

- Is it possible to determine the degree of novelty of input patterns of a pattern classifier?

The answer is: yes, this is possible for all in chapter 3 described classifiers. The best way in which such a confidence value is determined depends on the classification method. While the method best used for a particular problem depends among others on the complexity of the problem relative to the demands on time and memory complexity. This is shown in figure 8.1.

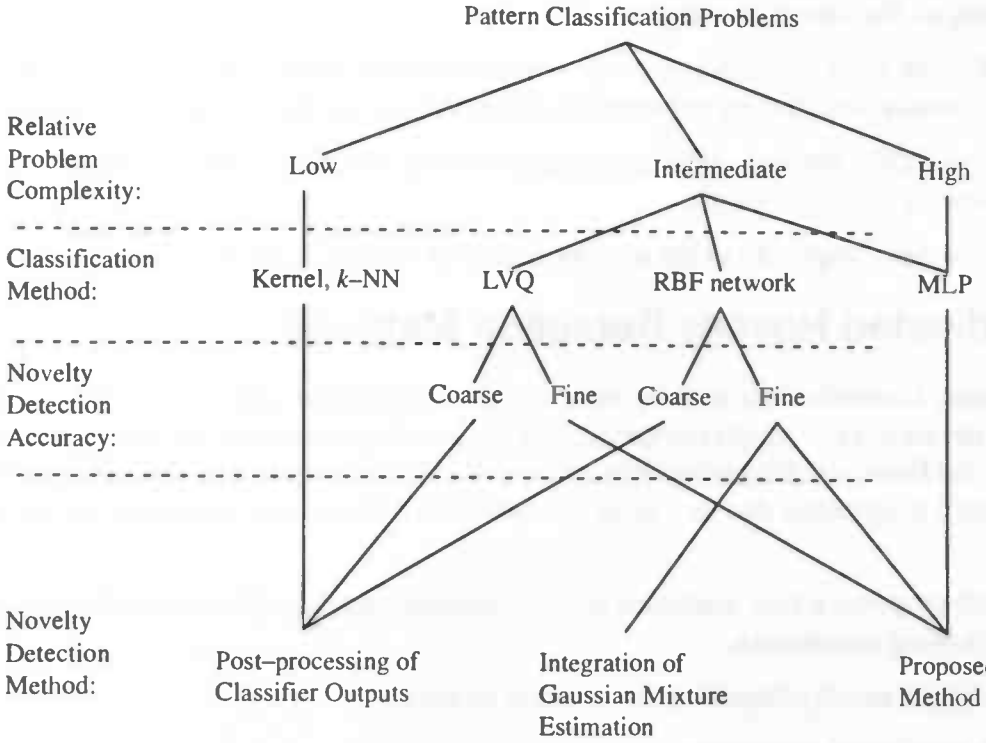

Figure 8.1: *Design decisions for classifiers featuring novelty detection. The problem complexity is relative to the system demands on speed and memory requirements.*

This conclusion is based on sub–conclusions regarding model complexity, classification accuracy and novelty detection ability of the different classification methods.

The sub–conclusions regarding model complexity and classification accuracy are:

- Classifiers based on **kernel or *k*–nearest neighbor (*k*–NN) estimation** hold the most **accurate classification models**, but have an **high model complexity**.
- Classifiers based on **radial basis function networks (RBF networks)** and **learning vector quantization (LVQ)** combine **good classification accuracy** with **low model complexity**.
- Classifiers based on a **multi–layer perceptron (MLP)** combine **good classification accuracy** with **very low model complexity**.

The sub–conclusions regarding novelty detection ability are:

- For classifiers based on **kernel and *k*–NN estimation** the degree of novelty of an input pattern can accurately be determined via **post–processing of the classifier outputs**.
- For classifiers based on **RBF networks and LVQ** the degree of novelty of an input pattern can **coarsely** be determined via **post–processing of the classifier outputs**.
- For classifiers based on **RBF networks** the degree of novelty of an input pattern can accurately be determined by an **Gaussian mixture density model** integrated in the network. This may however have a negative influence on the classification accuracy.
- For classifiers based on **MLP** the degree of novelty **cannot** be determined via **post–processing of the classifier outputs**.

So the MLP is the most suitable method for complex classification problems and the MLP does not support novelty detection by construction. From this can the following conclusion be drawn:

- For classifiers based on MLP a dedicated novelty detection method is needed that works separately of the classifier.
- The model complexity of the novelty detection method must be low.

8.2 Dedicated Novelty Detection Methods

In the literature is searched for novelty detection methods that are able to work separately from a classifier and have a low model complexity. Of the found methods only one meets the complexity demand: the Resource Allocating Novelty Detector [36]. However, experiments show that this method is hard to optimize due to a large parameter set and random influences on the training process.

This research proposes a new dedicated novelty detection method. This method is designed to meet the following constraints:

- Model the novelty boundary, not the data density.
- Low number of parameters.
- The quality of the parameter setting must be testable.

This resulted is a novelty detection for which:

- The training process is very fast
- The parameter optimization is easy
- The model complexity is low

From the first experiments with the proposed method can the following be concluded:

- Compared to other methods results the method in a tighter fit round the training data without overfitting it.
- The novelty boundary is consistent: local boundary quality is a measure for the quality of the whole boundary.
- The method is suitable for complex pattern recognition problems like optical character recognition.
- The method's model of similarity resembles the human perception of similarity.

Chapter 9 References

[1] C. Aviles–Cruz, *et al.*, *"Elena, Enhanced Learning for Evolutive Neural Architecture, Databases"*, ESPRIT–Basic Research Project Number 6891, Deliverable R3–B1–P, June 1995.

[2] J.S. Baras, and A. La Vigna, "Convergence of a Neural Network Classifier", *Advances in Neural Information Processing Systems*, Vol. 3, pp. 839 – 845, San Mateo – California: Morgan Kaufmann Publishers, 1991.

[3] U. Beyer, and F.J. Smieja, *"Quantitative Aspects of Data–Driven Information Processing"*, Technical Report 732, Gesellschaft für Mathematik und Datenverarbeitung, St Augustin, Germany, March 1993.

[4] C.M. Bishop, *"Neural Networks for Pattern Recognition"*, New York: Oxford University Press Inc., 1995.

[5] C.M. Bishop, "Novelty Detection and Neural Network Validation", in: *IEE Proceedings: Vision, Image and Signal Processing*, Vol. 141, No. 4, pp. 217 – 222, 1994.

[6] F. Blayo, *et al.*, *"Elena, Enhanced Learning for Evolutive Neural Architecture, Benchmarks"*, ESPRIT–Basic Research Project Number 6891, Deliverable R3–B4–P, June 1995.

[7] D.S. Broomhead, and D. Lowe, "Multivariable Functional Interpolation and Adaptive Networks", *Complex Systems*, Vol. 2, pp. 321 – 355, 1988.

[8] C. Chow, "On Optimum Recognition Error and Rejection Tradeoff", *IEEE Transactions on Information Theory*, No. 16, pp. 41 – 46, 1970.

[9] A.P. Dempster, M.N. Laird, and D.B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm", *J. Royal Statistical Society*, Series B, No. 39, pp. 1 – 38, 1977.

[10] L. Devroye, L. Györfi, and G. Lugosi, *"A Probablistic Theory of Pattern Recognition"*, New York: Springer–Verlag, 1995.

[11] R.O. Duda, and P.E. Hart, *"Pattern Classification and Scene Analysis"*, New York: John Wiley & Sons, 1973.

[12] E. Fix, and J. Hodges, *"Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties"*, Technical Report 4, Project Number 21–49–004, USAF School of Aviation Medicine, Randolph Field, Texas, 1951 (reprinted in: *"Nearest Neighbor Pattern Classification Techniques"*, B. Dasarathy, ed., pp 32–39, Los Alamitos: IEEE Computer Society Press, 1991).

[13] K. Fukunaga, *"Introduction to Statistical Pattern Recognition"*, 2nd Edition, San Diego: Academic Press, 1990.

[14] S. Geman, E. Bienenstock, and R. Doursat, "Neural Networks and the Bias/Variance Dilemma", in: *Neural Computation*, Vol. 4, No.1, pp. 1–58, 1992.

[15] J.B. Hampshire, and A.H. Waibel, "A Novel Objective Functions for Improved Phoneme Recognition Using Time–Delay Neural Networks", *IEEE Transactions on Neural Networks* Vol. 1, No. 2, June 1990.

[16] S. Haykin, *"Neural Networks, A Comprehensive Foundation"*, New York: Macmillan College Publishing Company, 1994.

[17] T. Heskes, "Practical Confidence and Prediction Intervals", to appear in: *Advances in Neural Information Processing Systems*, M. Moser, M. Jordan, and T. Petsche, Eds, Vol. 9, MIT Press, Cambridge, 1997.

[18] A. Hoekstra, S.A. Tholen, and R.P.W. Duin, "Estimating the Reliability of Neural Network Classifications", *Proceedings of the International Conference of Artificial Neural Networks*, Springer, 1996.

[19] L. Holmström, P. Koistinen, J. Laaksonen, and E. Oja, "Neural and Statistical Classifiers – Taxonomy and Two Case Studies", *IEEE Transactions on Neural Networks*, Vol. 8, No 1, pp. 5 – 17, January 1997.

[20] T. Kohonen, *"Learning Vector Quantization for Pattern Recognition"*, Technical Report TKK–F–A601, Helsinki University of Technology, 1986.

[21] T. Kohonen, *et al.*, *"LVQ_PAK, The Learning Vector Quantization Program Package"*, Version 3.1 (Obtainable from: http://nucleus.hut.fi/nnrc/lvq_pak/), Helsinki University of Technology, April 1995.

[22] T. Kohonen, *"Self–Organization and Associative Memory"*, 3rd Edition, Berlin: Springer–Verlag, 1989.

[23] D. Lowe, "Radial Basis Function Networks", in: *"The Handbook of Brain Theory and Neural Networks"*, M.A. Arbib, Ed., Cambridge, MA: MIT Press, 1995.

[24] D. Lowe, "On the iterative Inversion of RBF Networks: A Statistical Interpertation", *Proceedings of the 2nd IEE International Conference on Artificial Neural Networks*, pp. 29 – 33, 1991.

[25] G. McLachlan, *"Discriminant Analysis and Statistical Pattern Recognition"*, New York: Wiley, 1992.

[26] H. Mühlenbein, "Editorial", *Parallel Computing*, Special Edition on Neural Networks, Vol. 14 No. 3, pp. 247 – 248, August 1990.

[27] H. Ney, "On the Probabilistic Interpretation of Neural Network Classifies and Discriminative Training Criteria", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 17, No. 2, pp. 107 – 119, February 1995.

[28] J.A.G. Nijhuis, M.H. ter Brugge, K.A. Helmholt, J.P.W. Pluim, L. Spaanenburg, R.S. Venema, and M.A. Westenberg, "Car Licence Plate Recognition with Neural Networks and Fuzzy Logic", Presented at *7th Portugese Conference on Artificial Intelligence, EPIA'95*, October 1995.

[29] J.E. Moody, and C.J. Darken, "Fast Learning in Networks of Locally–tuned Processing Units", *Neural Computation*, Vol. 1, pp. 281 – 294, 1989.

[30] E. Parzen, "On the estimation of a Probability Density Function and Mode", *Annals of Mathematical Statistics*, No. 33, pp. 1065–1076, 1962.

[31] Penny W.D., and S.J. Roberts, *"Neural Network Predictions with Error Bars"*, Technical Report, submitted to IEEE Transactions on Neural Networks, available from http://www.ee.ic.ac.uk/research/neural/wpenny.html, 1997.

[32] M.J.D. Powell, "Radial Basis Functions for Multivariable Interpolation: a Review", in: *"Algorithms for Approximation"*, J.C. Mason, and M.G. Cox, Eds, pp. 143 – 167, Oxford: Clarendon Press, 1987.

[33] D.L. Reilly, L.N. Cooper, and C. Elbaum, "A Neural Model for Category Learning", *Biological Cybernetics*, vol. 45, No. 1, pp. 35 – 41, 1982.

[34] M.D. Richard, and R.P. Lippmann, "Neural Network Classifiers Estimate Bayesian a–posteriori Probabilities", *Neural Computation*, Vol. 3, No. 4, pp. 461 – 483, 1991.

[35] B.D. Ripley, *"Pattern Recognition and Neural Networks"*, Cambridge: Cambridge University Press, 1996.

[36] S.J. Roberts, and L. Tarassenko, "A Probablistic Resource Allocating Network for Novelty Detection", *Neural Computation*, Vol. 6, pp. 270 – 284, 1994.

[37] S.J. Roberts, W. Penny, and D.Pillot, "Novelty, Confidence and Errors in Connectionist Systems", *Proceedings of IEE Colloquium on Intelligent Sensors and Fault Detection*, Vol. 261, No 10, pp 1 – 6, September 1996.

[38] W.R. Ruck, *et al.*, "The Multilayer Perceptron as an Approximation to a Bayes Optimal Discriminant Function", in: *"Fuzzy Models for Pattern Recognition: Methods That Search for Structures in Data"*, J.C. Bezdek, and S.K. Pal, Eds, pp. 436 – 438, New York: IEEE Press, 1992.

[39] F. Smieja, "Rejection of Incorrect Answers from a Neural Net Classifier", *New Trends in Neural Computation (Proceedings of the IWANN '93)*, Springer Verlag, 1993.

[40] L. Tarassenko, *et al.*, "Novelty Detection for the Identification of Masses in Mammograms", in: *Proceedings of the Fourth International IEE Conference on Artificial Neural Networks (Cambridge, 1995)*, IEE Press, 1995.

[41] S.A. Tholen. *"Confidence Values for Neural Network Classifications"*, M.Sc. Thesis, Pattern Recognition Group, Delft University of Technology, April 1995.

[42] H.G.C. Tråvén, "A Neural Network Approach to Statistical Pattern Classification by 'Semiparametric' Estimation of Probability Density Functions", *IEEE Transactions on Neural Networks*, Vol. 2, No. 3, pp. 366 – 377, 1991.

[43] K. Tumer, and J. Ghosh, "Error Correlation and Error Reduction in Ensemble Classifiers", *Connection Science, Special Issue on Combining Artificial Neural Networks: Ensemble Approaches*, Vol. 8, No. 384, pp. 385 – 404, December 1996.

[44] P. Yee, and S. Haykin, "Pattern Classification as an Ill–posed, inverse problem: A Regularization Approach", in: *International Conference on Acoustics, Speech and Signal Processing*, Vol. 1, pp. 597 – 600, Minneapolis MN, 1993.