TR 3713

One-class classification

Concept-learning in the absence of counter-examples

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. ir. K.F. Wakker,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op 19 juni 2001 om 10.30 uur

door David Martinus Johannes TAX

doctorandus in de natuurkunde,
geboren te Ede.

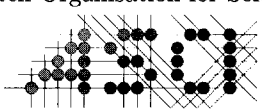
Dit proefschrift is goedgekeurd door de promotor:
Prof. dr. I.T. Young

Samenstelling promotiecommissie:

Rector Magnificus, voorzitter
Prof. dr. I.T. Young, TU Delft, promotor
Dr. ir. R.P.W. Duin, TU Delft, toegevoegd promotor
Prof. J. Kittler, University of Surrey
Prof. dr. ir. E. Backer, TU Delft
Prof. dr. P. Groeneboom, TU Delft
Prof. dr. ir. F.C.A. Groen, TU Delft
Dr. T. Heskes, University of Nijmegen

Dr. ir. R.P.W. Duin heeft als begeleider in belangrijke mate aan de totstandkoming van het proefschrift bijgedragen.

This work was partly supported by the Foundation for Applied Sciences (STW) and the Dutch Organisation for Scientific Research (NWO).



This work was carried out in graduate school ASCI. ASCI dissertation series number 65.

ISBN: 90-75691-05-x
© 2001, David Tax, all rights reserved.

CONTENTS

NOTATION

Euclidean length of vector $\mathbf{x}$ will be written $\|\mathbf{x}\|$, thus the squared Euclidean length of a vector becomes $\|\mathbf{x}\|^2$.

$\mathbf{a}$	center of the hypersphere in the SVDD
$\alpha_i (\alpha_j)$	a Lagrange multiplier in the SVDD for target object $\mathbf{x}_i$ ($\mathbf{x}_j$) to enforce data within the hypersphere
$\alpha_l (\alpha_m)$	a Lagrange multiplier in the SVDD for outlier object $\mathbf{x}_i$ ($\mathbf{x}_j$) to enforce data outside the hypersphere
b	bias
C	regularization parameter in the SVC/SVDD which regulates the tradeoff between complexity and errors or the number of classes in a multi-class classification problem
d	dimensionality of input patterns
$d_M(\mathbf{z})$	distance from object $\mathbf{z}$ to data description M
Δ^2	squared Mahanalobis distance
$E[y\|x]$	expection of y given x
$E_{\mathcal{X}^{tr}}[y]$	expectation of y over all possible instances of the training set $\mathcal{X}^{tr}$
$\varepsilon(f(\mathbf{x}), y)$	error or cost contribution of function f by one training object $(\mathbf{x}, y)$
$\mathcal{E}(f, \mathbf{w}, \mathcal{X}^{tr})$	total error or cost function for function f with weights $\mathbf{w}$ over the complete training set
$\mathcal{E}_{\mathrm{I}}$ ($\mathcal{E}_{\mathrm{II}}$)	cost for an error of the first kind (second kind)
$\mathcal{E}_{\mathrm{ret}}$	retrieval error defined in the image database retrieval application (formula (5.28))
$\mathcal{E}_{\mathrm{rank}}$	ranking measure to rank all images from an image database based on their similarity to a one-class classifier (definitions on pages 136 and 137)
ϵ	noise in the data, data which cannot be explained by the model
η	learning rate for LVQ
$f(\mathbf{x})$, $f(\mathbf{x}; \mathbf{w})$	output of (one-class) classifier (with free parameters $\mathbf{w}$) given input $\mathbf{x}$

$f(\mathbf{x}\|\omega_j)$	model for posterior probability of object $\mathbf{x}$ given class ω_j
f_T	user defined fraction of the target class which should be accepted
f_{T+}	fraction of the target objects which are accepted by a one-class classifier
f_{T-}	fraction of the target objects which are rejected by a one-class classifier
f_{O+}	fraction of the outlier objects which are accepted by a one-class classifier
f_{O-}	fraction of the outlier objects which are rejected by a one-class classifier
f_{SV}	fraction of the training objects which become support vectors in the SVDD or SVC (i.e. $\frac{n_{SV}}{N}$)
f_{SV}^{bnd}	fraction of the training objects which become support vectors *on* the boundary of the SVDD (i.e. $\frac{n_{SV}^{\text{bnd}}}{N}$)
f_{SV}^{out}	fraction of the training objects which become support vector *outside* the boundary of the SVDD (i.e. $\frac{n_{SV}^{\text{out}}}{N}$)
$f_{NN/G}$	ratio between the volumes of the feature space covered by the Gaussian and NN-d model (i.e. $f_{NN/G} = v_{NN}/v_G$)
$\Phi(\mathbf{x})$	mapping of vector $\mathbf{x}$ to a (higher dimensional) feature space
Φ_μ	mean of the mapped objects $\mathbf{x}$ into some feature space: $\Phi_\mu = \sum_i \Phi(\mathbf{x}_i)$
γ_i	Lagrange multiplier in the SVDD to enforce positive error for a target object
γ_l	Lagrange multiplier in the SVDD to enforce positive error for a outlier object
I	indicator function (see definition (2.14), page 24)
$\mathcal{I}$	identity matrix
$K(\mathbf{x}_i, \mathbf{x}_j)$	kernel function operating on objects $\mathbf{x}_i$ and $\mathbf{x}_j$
K', K''	constants used in the comparison of the mean and the product combination rule (used in inequality (5.12))
L	Lagrangian, the combination of an error function and constraints
λ	regularization parameter (tradeoff parameter between the empirical and structural error), or regularization parameter for the inversion of covariance matrix for Gaussian density
M	number of images in the image database

$\overline{m}$	average rank of n images in a database containing M images
$\boldsymbol{\mu}$	mean vector of a data set
n	degree of the polynomial kernel in the SVDD or SVC or: the number of test images in the image database application
N	number of training objects
$n_{\text{free}f}$	number of free parameters in model f
NN-d	data description based on nearest neighbor distances (see appendix B)
$\text{NN}^{tr}(\mathbf{x})$	(first) nearest neighbour of object $\mathbf{x}$
$\text{NN}_k^{tr}(\mathbf{x})$	kth nearest neighbour of object $\mathbf{x}$
n_{SV}	number of support vectors in the SVDD or SVC (these are the objects $\mathbf{x}_i$ with $0 < \alpha_i \leq C$)
n_{SV}^{bnd}	number of support vectors *on* the boundary of the SVDD (these are the support vectors $\mathbf{x}_i$ with $0 < \alpha_i < C$)
n_{SV}^{out}	number of support vectors *outside* the boundary of the SVDD (these are the support vectors $\mathbf{x}_i$ with $\alpha_i = C$)
ν	user defined parameter in the ν-SVC indicating the fraction of the data what should be captured (used in error (2.54))
$p(.)$	probability density
$p(.\|.)$	conditional probability density
$p_N(\mathbf{x};\boldsymbol{\mu},\Sigma)$	normal or Gaussian distribution, characterized by mean $\boldsymbol{\mu}$ and covariance matrix Σ
ρ	bias term in the ν-SVC
R	number of combined classifiers or radius of the hypersphere in the SVDD
s	width of the Gaussian kernel in the SVDD
σ	standard deviation
Σ	full covariance matrix
SV	support vector; object with $\alpha_i > 0$
SV^{bnd}	set of support vectors with Lagrange multipliers $0 < \alpha_i < C$
SV^{out}	set of support vectors with Lagrange multipliers $\alpha_i = C$
SVC	Support Vector Classifier
SVDD	Support Vector Data Description
θ	threshold on a probability or a distance
θ_{ij}	angle between vectors $\mathbf{x}_i$ and $\mathbf{x}_j$
$V(\mathbf{r})$	volume of an hypersphere with radius $\mathbf{r}$
v_G, v_{NN}	volume of the feature space covered by the Gaussian and NN-d model respectively

$\mathbf{w}$	weight vector
ω_j	class identifier, used in conditional probabilities $p(\mathbf{x}\|\omega_j)$
ω_T, ω_O	target, outlier class
$\mathbf{x}$	column vector, most often representing an object
$\mathcal{X}$	feature space containing all possible objects $\mathbf{x}$
$\mathcal{X}_T$	area in feature space which contains all target objects
$\mathcal{X}^{tr}$	training set containing N objects $\mathbf{x}_i$ with their labels y_i; in one-class problems often just target objects are present
ξ_i	slack variables introduced in SVC/SVDD to account for errors
χ^2_d	χ^2-distribution with d degrees of freedom
ζ_j^k	deviation of the output of one classifier $f_j^k(\mathbf{x}^k)$ for class ω_j from the average output $\bar{f}_j(\mathbf{x})$ of a set of classifiers
y_i	output label for pattern $\mathbf{x}_i$, often $y \in \{-1, 1\}$
$y(\mathbf{x})$, $y(\mathbf{x}\|\mathbf{w})$	output of a classifier or combining rule, characterized by $\mathbf{w}$, after presenting pattern $\mathbf{x}$
$y_{\text{comb}}(\mathbf{x})$	combining rule for (one-class) classifiers (*comb* can be one of: *mv*, *mwv*, *pwv*, *mp*, *pp*, see page 125 for explanation)
$\mathbf{z}$	column vector, most often representing a test object
$\mathcal{Z}$	area in feature space which contains all outlier objects

1. INTRODUCTION

What would you do if you were given a collection of apples and pears and you were asked to distinguish between the two types of fruit? This problem does not seem very complicated, everyone can immediately separate them based on how they look and feel. Moreover, it is not very hard to identify the rotten pieces, the dirt and all other objects which are neither apples nor pears.

Although this problem of distinguishing apples and pears does not look very complicated, automating this process turns out to be fairly complicated. After all, what should be the basis for the decision to call an object 'apple', and another object 'pear'? Is it the weight of the object, the height or color? Perhaps the shape, smell or the flavor? Very likely it is a combination of all of these properties. Assume for a moment we use measurements of the color and the smell of an object, how should an apple with some mud on it be classified? Is it an apple or is it dirt?

This is the problem of classification: to assign a new object to one of a set of classes (in this case apples or pears) which are known beforehand. The classifier which should perform this classification operation (or which assigns to each input object an output label), is based on a set of example objects. This thesis will not focus on this classification problem though, but on the next problem, the problem of one-class classification. Here an object should be classified as a genuine object (apple or pear), or an outlier object (other type of fruit, rotten fruit or dirt).

The one-class classification problem differs in one essential aspect from the conventional classification problem. In one-class classification it is assumed that only information of *one* of the classes, the target class, is available. This means that just example objects of the target class can be used and that no information about the other class of outlier objects is present. The boundary between the two classes has to be estimated from data of only the normal, genuine class. The task is to define a boundary around the target class, such that it accepts as much of the target objects as possible, while it minimizes the chance of accepting outlier objects.

In this first chapter we will give the framework in which the research is placed. We will start with a short introduction to statistical pattern recognition and we will give the classical problems and some possible solutions to these problems. Then the topic of this thesis, the problem of one-class classification, will be introduced and related to the conventional classification. We will see that one-class classification suffers from the same problems as conventional classification, and even worse, we will show that one-class classification has some additional problems.

1.1 Learning from examples

In many classification problems explicit rules do not exist (can you give explicit rules for distinguishing apples from pears?), but examples can be obtained easily (ask your green grocer). A classifier, i.e. a function which outputs a class label for each input object, cannot be constructed from known rules. Therefore, in pattern recognition or machine learning, one tries to infer a classifier from a (limited) set of training examples. The use of examples thus elevates the need to explicitly state the rules for the classification by the user. The goal is to obtain models and learning rules to learn from the examples and predict the labels of future objects.

The notion of 'object' is taken very broadly, it can range from apples and pears to handwritten digits, from speech signals to machine noise. In this thesis we assume that objects are described by vectors containing a set of d real valued measurements, thus an object i is represented by the feature vector $\mathbf{x}_i = (x_{i,1}, ..., x_{i,d}), x_{i,j} \in \mathbb{R}$ (or shorter $\mathbf{x}_i \in \mathcal{X} = \mathbb{R}^d$). Each object is thus represented as one point in a feature space $\mathcal{X}$. Furthermore, we assume that all components in the vector are known and that there are no missing values. In practice it might happen that some measurements are not performed, due to costs in time, money and effort or because it is expected that they provide too little information (this can happen for instance in medical applications). The missing values introduce extra complications and we will not consider them. We assume all objects are characterized with the same set of measurements.

Further, we assume that the continuity assumption holds. This is a general assumption in pattern recognition: two objects near in feature space should also resemble each other in real life. When we are at one position in feature space, representing an example object, and we change the position a bit, then this new position should represent a very similar object. This means that we assume that the objects are not randomly scattered into some feature space, but that they are distributed in cloud-like distributions. In particular, it means that when we look in the neighborhood of an object, similar objects are represented. When this continuity does not hold, we cannot expect to learn well from a few example objects. The decision boundaries could be placed almost anywhere in the feature space. Huge amounts of examples are required to find all these irregularities in the data. Only for constant class membership, it is allowed to have really different objects nearby in feature space.[1] Obviously, to be able to make a distinction between objects and classes of objects, the measurements should contain enough information to distinguish the objects. In other words, they should have enough discriminative power such that a classifier will show sensible generalization. When only noise measurements are available, we cannot expect to infer a good classification.

A general multi-class classification problem can be decomposed in several two-class classification problems [Fukanaga, 1990]. Therefore, the two-class problem is considered as the basic classification problem. In a two-class classification problem, the two classes ω_1 and ω_2 will be labeled by -1 and $+1$ respectively. A training set is a set of objects for

[1] But then the objects are not really different as far as the classifier is concerned.

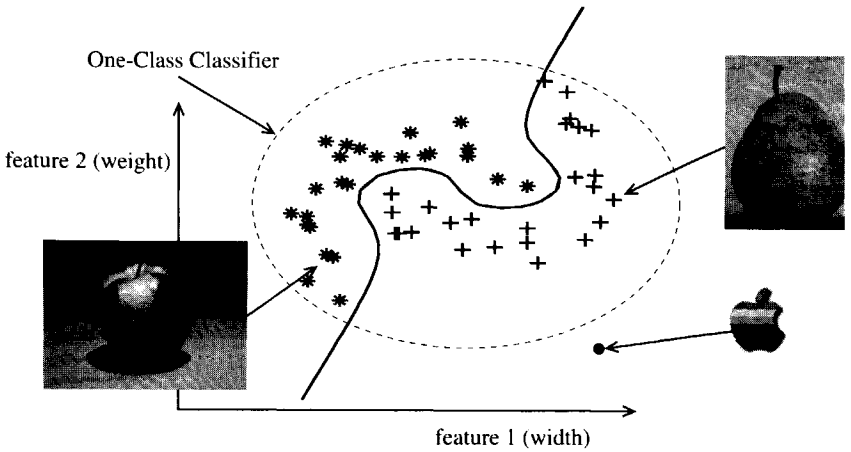


Fig. 1.1: A conventional and a one-class classifier applied to an example dataset containing apples and pears, represented by 2 features per object. The solid line is the conventional classifier which distinguishes between the apples and pears, while the dashed line describes the dataset. This description can identify the outlier apple in the lower right corner, while the classifier will just classify it as an pear.

which for each object $\mathbf{x}_i$ a label y_i is attached ($y_i \in \{-1, +1\}$):

$$\mathcal{X}^{tr} = \{(\mathbf{x}_i, y_i) | i = 1, .., N\} \tag{1.1}$$

For the classification, a function $f(\mathbf{x})$ has to be inferred from the training set. This function should be constructed such that for a given feature vector $\mathbf{x}$ an estimate of the label is obtained, $y = f(\mathbf{x})$:

$$f : \mathbb{R}^d \rightarrow \{-1, +1\} \tag{1.2}$$

In figure 1.1 an example of a training dataset is given for the apple-pear problem from the beginning of this chapter. Each object has two feature values (for instance the width and the height of the object; the exact features are not important for this discussion). Each training object $\mathbf{x}$ can therefore be represented as a point in a 2-dimensional feature space. Here the apples are indicated by stars, the pears by pluses. In principle, objects can be scattered all around the (2-dimensional) feature space, but due to the continuity assumption, apples are near apples and pears near pears. Furthermore, there are physical constraints on the measurement values (weights and sizes are positive, and are bounded by some large number).

In the apple-pear example the two classes can be separated without errors by the solid line in figure 1.1. Unfortunately, when the outlier apple in the right lower corner is introduced, it cannot be distinguished from the pears. In the world of the two class

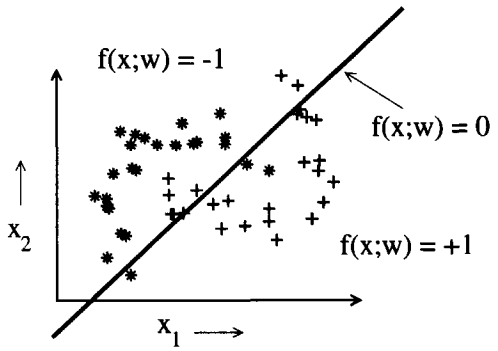


Fig. 1.2: Scatterplot of the training set available for the apple-pear classification problem. A simple (linear) classifier $f(\mathbf{x};\mathbf{w})$ is shown. From the two measurements x_1 and x_2 per object it estimates a label $f(\mathbf{x};\mathbf{w}) = +1$ or $f(\mathbf{x};\mathbf{w}) = -1$. The line $f(\mathbf{x};\mathbf{w}) = 0$ is the decision boundary (in this case linear).

classifier only apples and pears exist, and all objects are either apples or pears. Objects cannot be something else, like an outlier. To identify the outlier, a one-class classifier should be trained. An example of a one-class classification is given by the dashed line.

Most often the type of function f is chosen beforehand and just a few parameters of the function have to be determined. The function can be denoted by $f(\mathbf{x};\mathbf{w})$ to explicitly state the dependence on the parameters or weights $\mathbf{w}$ (in a specific application in the next chapter these parameters will be called $\boldsymbol{\alpha}$). Examples of these functions are linear classifiers, mixtures of Gaussians, neural networks or support vector classifiers. In figure 1.2 a scatterplot of the training set of the apple-pear problem is shown again. Twentyfive examples per class are available. A linear classifier $f(\mathbf{x};\mathbf{w})$ is drawn (although this is not the optimal linear classifier). For most objects it correctly estimates the labels. For three apples and seven pears however a wrong label is assigned.

To find the optimal parameters $\mathbf{w}^*$ for the function f on a given training set $\mathcal{X}^{tr}$, an error function $\mathcal{E}(f,\mathbf{w},\mathcal{X}^{tr})$ has to be defined. Most often the objects in the training data are assumed to be independently distributed, and the total error of function f on a training set is decomposed as:

$$\mathcal{E}(f,\mathbf{w},\mathcal{X}^{tr}) = \frac{1}{N}\sum_i \varepsilon(f(\mathbf{x}_i;\mathbf{w}),y_i) \tag{1.3}$$

Different definitions for the error function are possible, depending on the type of $f(\mathbf{x}_i;\mathbf{w})$. The first possibility is the 0-1−loss, for a discrete valued $f(\mathbf{x}_i;\mathbf{w})$. This counts the number of wrongly classified objects:

$$\varepsilon_{0\text{-}1}(f(\mathbf{x}_i;\mathbf{w}),y_i) = \begin{cases} 0, & \text{if } \; f(\mathbf{x}_i;\mathbf{w}) = y_i, \\ 1, & \text{otherwise.} \end{cases} \tag{1.4}$$

For instance, in our apple-pear example this error becomes $\mathcal{E}_{0\text{-}1} = 10$.

The most common error for real-valued functions $f(\mathbf{x}_i; \mathbf{w}) \in [-1, 1]$ is the mean squared error (MSE):

$$\varepsilon_{\text{MSE}}(f(\mathbf{x}_i; \mathbf{w}), y_i) = (f(\mathbf{x}_i; \mathbf{w}) - y_i)^2 \tag{1.5}$$

and the cross entropy (where the labels should be rescaled to positive values $y_i = \{0, 1\}$):

$$\varepsilon_{\text{ce}}(f(\mathbf{x}_i; \mathbf{w}), y_i) = f(\mathbf{x}_i; \mathbf{w})^{y_i} \left(1 - f(\mathbf{x}_i; \mathbf{w})\right)^{1-y_i} \tag{1.6}$$

By minimizing the error $\mathcal{E}$ on the training set, one hopes to find a good set of weights $\mathbf{w}$ such that a good classification is obtained.

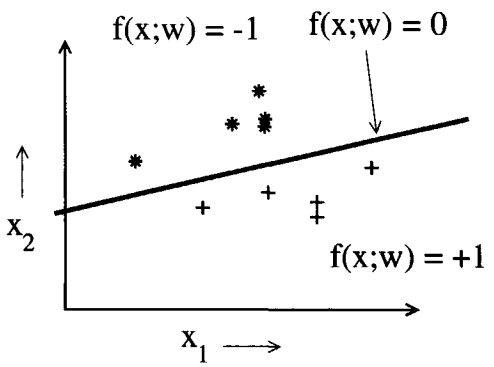


Fig. 1.3: Scatterplot of a small, atypical training set for the apple-pear classification problem. A simple (linear) classifier $f(\mathbf{x}; \mathbf{w})$ is trained on this data. Because the training set does not resemble the 'true' distribution, the generalization performance is poor.

This poses a new problem: the set of training examples might be a very uncharacteristic set. If a limited sample is available, the inherent variance in the objects and noise in the measurements might be too big to extract classification rules with high confidence. The smaller the number of training examples, the more pronounced this problem becomes. This is shown in figure 1.3, where a smaller training set for the apple-pear classification problem is given. Instead of 25 examples per class, now just 5 examples per class are available. The best classifier for this small set deviates significantly from the optimal linear classifier for the complete, 'true' data distribution.

Even worse, in some cases it might be completely unclear if the training data distribution resembles the distribution in real life. Sometimes only objects on the edges of the data distribution are known or available, or the user has to guess what normal and representative data are without much knowledge about the problem. Imagine the task of composing a training set for the apple-pear classification problem. It is not hard to buy some apples and pears at your green grocer, but it is very likely that they will not be a representative sample of the complete class of apples and pears. The type of green grocer

already influences the quality of the fruits, the sizes, the number of rotten pieces. Although it is expected that the training data is distributed in the same area in the feature space as the 'true' data, the exact data density might differ significantly.

In general, the larger the sample size, the better the characteristics of the data can be determined. But even when a good, characteristic sample is available, the number of functions which approximates or precisely fits the data is very big (perhaps even infinite!). Therefore, good classification of the training objects is not the main goal, but good classification of *new and unseen* objects is. This is called good generalization. The main goal in pattern recognition is to find classifiers that show good generalization.

To estimate how well a classifier generalizes, it should be tested with a new set of objects which has not been used for training. By using a set of objects which has not used during training (an independent set), one avoids an overly optimistic estimate of the performance. The set which is not used in training but is used for estimating the error of the classifier is called the (independent) test set. In pattern recognition holds: what you see (on the training set) is not what you get (on an independent test set)!

In most cases correctly labeled data is scarce and expensive. From these objects both a training objects as well as testing objects should be drawn. Leaving out a set of objects from a small labeled set might leave out valuable information and therefore reduces the generalization of the classifier. More advanced rotation or cross-validation methods can be used to reduce this danger [Bishop, 1995].

The optimal parameters $\mathbf{w}^*$ of the function f are the parameters which give the smallest average error over *all possible* samples:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}}\, \mathcal{E}_{\text{true}}(f, \mathbf{w}, \mathcal{X}) \tag{1.7}$$

where the true error $\mathcal{E}_{\text{true}}$ is defined as:

$$\mathcal{E}_{\text{true}}(f, \mathbf{w}, \mathcal{X}) = \int \mathcal{E}(f(\mathbf{x}; \mathbf{w}), y) p(\mathbf{x}, y) d\mathbf{x} dy \tag{1.8}$$

Note that the integration is over the whole 'true' data distribution $p(\mathbf{x}, y)$.

In (1.8) we have assumed that it is possible to define a probability density $p(\mathbf{x}, y)$ in the complete feature space $\mathcal{X}$. This $p(\mathbf{x}, y)$ contains everything that can be known about the data in this representation.[2] When we have to classify handwritten digits for instance, the optimal $f(\mathbf{x}; \mathbf{w})$ (in terms of the error (1.8)) will be far different from the optimal function that distinguishes natural textures, even when both the digits and the textures are coded as a 16×16 vector of pixels (representing a 16×16 pixel image). Because the $p(\mathbf{x}, y)$ in (1.8) differs, the optimal weight vector $\mathbf{w}^*$ will differ as well.

In almost all classification problems this $p(\mathbf{x}, y)$ will be unknown. It is hoped that the training set is a representative sample from this true distribution, but in most cases this

[2] The optimal solution of the decision boundary will still depend on factors like the varying costs for erroneous classifications, constraints on the type of decision boundary, the speed of the evaluation of the classifier etc.

might not be the case. In section 1.4, where we discuss the one-class classification problem, we will encounter a situation where it is not possible anymore to define a probability density. To find a solution, extra assumptions will have to be made.

1.2 Generalization

The classifier with the minimum error $\mathcal{E}_{\text{true}}$ is called the Bayes decision rule [Duda and Hart, 1973]. This rule assigns an object $\mathbf{x}$ to the class with the largest posterior probability $p(\omega_k|\mathbf{x})$. When $p(\omega_1|\mathbf{x})$ is the posterior probability of class ω_1 for a given $\mathbf{x}$, the Bayes rule assigns labels:

$$f_{Bayes}(\mathbf{x}) = \begin{cases} +1 & \text{if} \quad p(\omega_1|\mathbf{x}) \geq p(\omega_2|\mathbf{x}), \\ -1 & \text{if} \quad p(\omega_1|\mathbf{x}) < p(\omega_2|\mathbf{x}). \end{cases} \tag{1.9}$$

The Bayes rule is the theoretical optimal rule; it has the best classification performance over all possible classifiers [Bishop, 1995] (assuming that all erroneously classified objects are equally weighted). The problem is that the Bayes rule requires the *true* posterior probabilities of all the classes for all $\mathbf{x}$. In practice the true distribution is not known and only examples (perhaps only atypical examples) from the distribution are available. Except for artificially generated data, where the class distributions are known, the optimal Bayes rule cannot be calculated in practice. The Bayes rule is only useful when artificial data distributions are used, to investigate the performance of classifiers $f(\mathbf{x};\mathbf{w})$ and how well they approach the optimal Bayes rule.

In the computation of the true error $\mathcal{E}_{\text{true}}$ (1.8) of a user defined function $f(\mathbf{x};\mathbf{w})$ on some data, the same problems as for the use of the Bayes rule appear. It requires an integration over the complete probability density of *all* possible objects $\mathbf{x}$ and labels y. This probability density is unknown and we are only given a limited set of training examples. An induction principle has to be adopted to approximate the true error [Vapnik, 1995]. In practice error $\mathcal{E}_{\text{true}}$ is often approximated by the empirical error on the training set:

$$\mathcal{E}_{\text{emp}}(f,\mathbf{w},\mathcal{X}^{tr}) = \frac{1}{N}\sum_i \varepsilon(f(\mathbf{x}_i;\mathbf{w}),y_i) \tag{1.10}$$

This error gives an approximation of the true error, which becomes accurate when the training data is distributed like the true data distribution and the sample size is very large. Unfortunately, when it is used to optimize $f(\mathbf{x};\mathbf{w})$, a low empirical error $\mathcal{E}_{\text{emp}}$ does not *guarantee* a low true error $\mathcal{E}_{\text{true}}$ (a low error on an independent test set) [Vapnik, 1998]. The phenomenon that a function $f(\mathbf{x};\mathbf{w})$ minimizes the empirical error (1.10) very well on a training set but still shows a large true error $\mathcal{E}_{\text{true}}$ (1.8) on an independent test set, is called overtraining or overfitting. A sufficiently flexible function $f(\mathbf{x};\mathbf{w})$ can always perfectly fit the training data and thus show zero empirical error. The function then completely adapts to all available information, including noise, in the given examples.

The overfitting of a function $f(\mathbf{x};\mathbf{w})$ is shown in figure 1.4. Here a very flexible function is trained on the apple-pear example. Because the function is far too flexible for this data,

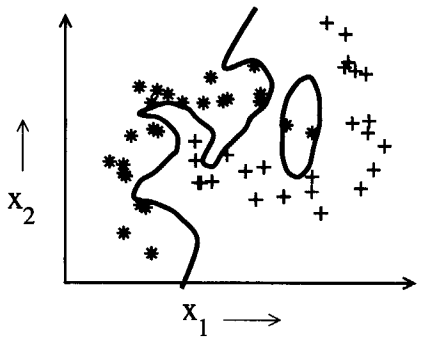


Fig. 1.4: A heavily overtrained classifier, trained on the training set available for the apple-pear classification problem. The classifier shows zero empirical error, but high true error.

it finds structure in the data which is not really there. The overtrained classifier suggests that the left class (the apple class) actually consists of two separate clusters. Objects placed in between these two clusters will be classified as pears! Testing this classifier with independent test data, will reveal that the generalization performance is not very high.

This overfitting problem becomes more severe when a large number of features per object is used. Because the function $f(\mathbf{x};\mathbf{w})$ should be defined for all possible $\mathbf{x}$ (i.e. the complete feature space), the volume that should be described increases exponentially in the number of features d. This is called the *curse of dimensionality* [Duda and Hart, 1973, Bishop, 1995]. By decreasing the number of features per object, the number of degrees of freedom in the function $f(\mathbf{x};\mathbf{w})$ decreases and the generalization performance increases. This means that increasing the number of features per object can harm the classification performance!

One solution to the curse of dimensionality and overfitting is to use feature reduction or feature selection and retain only the few best features [Devijver and Kittler, 1982, Pudil et al., 1994]. It is also possible to compute a small number of new features from the set of old features (for instance a linear combination of old features), which is called feature extraction. But when a complex function $f(\mathbf{x};\mathbf{w})$ (with a large number of free parameters $\mathbf{w}$) is used, these methods will still not be sufficient to guarantee good generalization.

An important quantity is the number of objects, the sample size N, with respect to the dimensionality of the feature space d. When we speak about small sample sizes, we mean that the number of example objects is not sufficient to estimate all free parameters $\mathbf{w}$ in the function $f(\mathbf{x};\mathbf{w})$ with enough accuracy. In this case probably low generalization performance are obtained [Raudys and Jain, 1991].

When a function $f(\mathbf{x};\mathbf{w})$ is overtrained on some data, it depends on both the flexibility of the function as well as on how well the error is minimized. When a very flexible function

is trained on a new training sample $\mathcal{X}^{tr}$ of the same size from the same distribution, a completely different solution for $\mathbf{w}$ is obtained. This will happen in the apple-pear classification problem for the classifier shown in figure 1.4. Then the weights $\mathbf{w}$ and thus function $f(\mathbf{x}; \mathbf{w})$ show a large variance over different training samples.

When we introduce the average of the function f over all training sets $\mathcal{X}^{tr}$: $E_{\mathcal{X}^{tr}}[f(\mathbf{x}; \mathbf{w})]$, it is to be expected that the variance of the individual functions $f(\mathbf{x}; \mathbf{w})$ around $E_{\mathcal{X}^{tr}}[f(\mathbf{x}; \mathbf{w})]$ will be large. (Note that in the term $E_{\mathcal{X}^{tr}}[f(\mathbf{x}; \mathbf{w})]$ the weight vector $\mathbf{w}$ will have a different optimum over the different training sets $\mathcal{X}^{tr}$.) Only by using a very large training set will the noise contributions cancel out and the function will follow the true data characteristics. The empirical error will then approach the true error [Vapnik, 1998]. This classifier is said to have high complexity.

The opposite problem of underfitting also occurs. Here the function or model $f(\mathbf{x}; \mathbf{w})$ is not flexible enough to follow all characteristics in the data. The model then shows a large bias (or has non-zero bias). Even an infinite amount of data cannot correct the systematic errors. On average the difference between the results of the model $E_{\mathcal{X}^{tr}}[f(\mathbf{x}; \mathbf{w})]$ and what the data labels y show, does not vanish. This classifier is said to have low complexity. This is what has been shown in figure 1.2. Although a reasonable amount of training data for the apple-pear classification problem was available, the chosen model was not flexible enough to capture the characteristics in the data. Another classifier, more flexible than the linear one, should have been used in this example. But in practice, because in most cases the amount of available data is limited and flexible functions are used, the underfitting problem is considered less important than the overfitting problem.

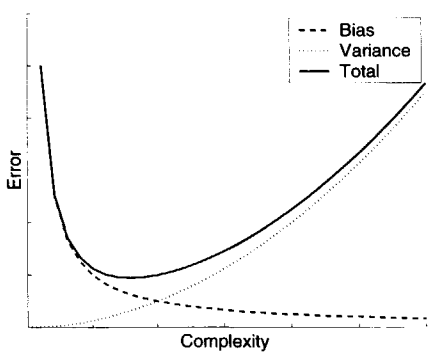


Fig. 1.5: The bias-variance tradeoff. For increasing complexity of the model, the bias component of the error decreases, but the variance component increases. For a certain model complexity, the total error is minimal. Different models show different bias-variance tradeoff curves.

The best fitting function for a given sample is therefore a tradeoff between the bias and the variance contributions [Geman et al., 1992, Friedman, 1997] (see figure 1.5). A good fitting function should have both a small bias and a small variance. The function

should be flexible enough to capture the data, but it should also be simple enough to avoid overfitting. This is called the bias-variance dilemma. In our apple-pear classification problem, the classifier with the optimal bias-variance tradeoff will be the classifier from figure 1.1. The classifier is capable of closely following the optimal Bayes classifier (small bias) but it is also robust against small changes in the training set (small variance).

These bias and variance contributions can directly be observed in a decomposition of the mean squared error $\mathcal{E}_{\text{MSE}}$. To simplify the notation, we assume that the labels in the training set y_i are noise-free.[3] Then we can compute the expected mean squared error $E_{\mathcal{X}^{tr}}[\mathcal{E}_{\text{MSE}}(f, \mathbf{w}, \mathcal{X}^{tr})]$ over all training sets $\mathcal{X}^{tr}$ and we can expand to [Geman et al., 1992, Haykin, 1999]:

$$
\begin{aligned}
E_{\mathcal{X}^{tr}}[\mathcal{E}_{\text{MSE}}(f, \mathbf{w}, \mathcal{X}^{tr})] &= E_{\mathcal{X}^{tr}}\left[\frac{1}{N}\sum_i \left(f(\mathbf{x}_i; \mathbf{w}) - y_i\right)^2\right] \\
&= E_{\mathcal{X}^{tr}}\left[\frac{1}{N}\sum_i \left(f(\mathbf{x}_i; \mathbf{w}) - E_{\mathcal{X}^{tr}}[f(\mathbf{x}_i; \mathbf{w})] + E_{\mathcal{X}^{tr}}[f(\mathbf{x}_i; \mathbf{w})] - y_i\right)^2\right] \\
&= E_{\mathcal{X}^{tr}}\left[\frac{1}{N}\sum_i \left(f(\mathbf{x}_i; \mathbf{w}) - E_{\mathcal{X}^{tr}}[f(\mathbf{x}_i; \mathbf{w})]\right)^2\right] + E_{\mathcal{X}^{tr}}\left[\frac{1}{N}\sum_i \left(E_{\mathcal{X}^{tr}}[f(\mathbf{x}_i; \mathbf{w})] - y_i\right)^2\right] \\
&\quad + E_{\mathcal{X}^{tr}}\left[\frac{1}{N}\sum_i 2\left(f(\mathbf{x}_i; \mathbf{w}) - E_{\mathcal{X}^{tr}}[f(\mathbf{x}_i; \mathbf{w})]\right)\left(E_{\mathcal{X}^{tr}}[f(\mathbf{x}_i; \mathbf{w})] - y_i\right)\right] \qquad (1.11)
\end{aligned}
$$

Here $E_{\mathcal{X}^{tr}}[f(\mathbf{x}; \mathbf{w})]$ is the expectation of the output of the classifier $f(\mathbf{x}; \mathbf{w})$ over all possible training sets (with the same sample size N).

We computed the expected mean square error over all possible training sets $\mathcal{X}^{tr}$ to investigate the average behavior. It might happen that we are lucky and we have a very good training set, which results in a very good generalization performance. On the other hand, for a very atypical training set, we can only expect poor generalization results. To investigate how well a function fits the data, we have to average out these contributions.

Because we averaged over all the training sets, we can remove the second part of the last term $(E_{\mathcal{X}^{tr}}[f(\mathbf{x}_i; \mathbf{w})] - y_i)$ from the expectation since this does not depend on $\mathcal{X}^{tr}$. Then the first part of the last term becomes zero: $E_{\mathcal{X}^{tr}}[f(\mathbf{x}_i; \mathbf{w}) - E_{\mathcal{X}^{tr}}[f(\mathbf{x}_i; \mathbf{w})]] = 0$ and

[3] If that is not the case, we have to decompose the observed labels into a model and a noise contribution: $y_i = E[y|\mathbf{x}_i] + \epsilon$. The first term $E[y|\mathbf{x}_i]$ represents the deterministic part of the data which we try to model. The second term ϵ contains all stochastic contributions and is often assumed to have zero mean. The bias-variance decomposition considers the difference between what a function f can represent and the deterministic part of the data $E[y|\mathbf{x}]$. When y_i contains some stochastic elements, replace y_i by $E[y|\mathbf{x}_i]$ in the coming decomposition.

therefore the third term in equation (1.11) vanishes:

$$\begin{aligned} & E_{\mathcal{X}^{tr}}\left[\mathcal{E}_{\text{MSE}}(f, \mathbf{w}, \mathcal{X}^{tr})\right] \\ & = E_{\mathcal{X}^{tr}}\left[\frac{1}{N}\sum_i \left(f(\mathbf{x}_i; \mathbf{w}) - E_{\mathcal{X}^{tr}}[f(\mathbf{x}_i; \mathbf{w})]\right)^2\right] + E_{\mathcal{X}^{tr}}\left[\frac{1}{N}\sum_i \left(E_{\mathcal{X}^{tr}}[f(\mathbf{x}_i; \mathbf{w})] - y_i\right)^2\right] \\ & = \text{variance} + (\text{bias})^2 \end{aligned} \tag{1.12}$$

Also for other types of errors, like the loglikelihood [Heskes, 1998], a bias-variance decomposition can be made. This indicates that the bias-variance dilemma is a fundamental problem in fitting a model to some data.

The bias-variance dilemma can be suppressed by introducing prior knowledge into the design of the function $f(\mathbf{x}; \mathbf{w})$. By including prior knowledge, e.g. constraints on the form of f to the problem at hand, the complexity of the function is decreased while still the required flexibility is retained. So the bias contribution will stay small and the variance is limited. Unfortunately, either prior knowledge is not often available or it is very hard to include the specific knowledge in the design of $f(\mathbf{x}|\mathbf{w})$. A few examples of including prior knowledge in a function f are the number of clusters in k-means clustering, the intrinsic dimensionality in a diabolo network, the rotation and translation invariance in a support vector classifier [Schölkopf, 1997] and the same invariances in a neural network, i.e. the shared weight networks [Le Cun et al., 1989].

In practice, when no extra prior knowledge is available, a (relatively) complex function (or model) is taken and an extra error term $\mathcal{E}_{\text{struct}}(f, \mathbf{w})$ is added to the empirical error (1.10). This structural error tries to measure the complexity of the function $f(\mathbf{x}; \mathbf{w})$. So instead of minimizing just $\mathcal{E}_{\text{emp}}$, we have to minimize the total error:

$$\mathcal{E}_{\text{tot}}(f, \mathbf{w}, \mathcal{X}^{tr}) = \mathcal{E}_{\text{emp}}(f, \mathbf{w}, \mathcal{X}^{tr}) + \lambda \mathcal{E}_{\text{struct}}(f, \mathbf{w}) \tag{1.13}$$

The regularization parameter λ indicates the relative influence of the structural error with respect to the empirical error. By setting $\lambda = 0$ the structural error will be ignored, while for a very large λ a very simple function is obtained which completely ignores the data. This parameter has to be set by the user. It is hoped that with this extended error, a classifier with higher generalization is obtained. [4]

In most cases the form of the structural error function is based on the continuity assumption of the data: two objects that are near in feature space also resemble each other in real life, imposing a smoothness on the function. The function should change smoothly over the feature space and highly fluctuating solutions are discouraged. The smoother the function, the lower the complexity. This thus assumes that the underlying data can be described by a simple function and that very irregular $f(\mathbf{x}; \mathbf{w})$'s are unlikely. This is the expression of the belief in the principle of Occam's razor: you should not introduce extra complexity when it is not necessary [Bishop, 1995].

[4] It is also possible to use this $\mathcal{E}_{\text{tot}}$ as an estimate for the true error the model f is going to make on the classification problem. This poses stronger constraints on the $\mathcal{E}_{\text{struct}}$, and we will not consider it in the coming discussion. We focus on the engineering problem of training a classifier with high generalization.

Therefore, the minimization of the structural error should suppress high complexity solutions for the function $f(\mathbf{x};\mathbf{w})$. It has been shown that when complexity constraints are enforced, the empirical error $\mathcal{E}_{\text{emp}}$ approaches the true error $\mathcal{E}_{\text{true}}$ better and the function $f(\mathbf{x};\mathbf{w})$ can be applied with greater confidence to new objects [Smolensky et al., 1996].

Numerous approaches have been developed to approximate or model the structural error $\mathcal{E}_{\text{struct}}$ (see [Wolpert, 1994] for an extended discussion about this topic). The simplest methods just count the number of free parameters in the function. Minimizing this number can be done by starting with a very simple model. A somewhat more advanced method is to use weight decay. Here a term $\mathcal{E}_{\text{struct}}(f,\mathbf{w}) = ||\mathbf{w}||^2$ is introduced which minimizes the norm of $\mathbf{w}$. By minimizing the norm of $\mathbf{w}$ it is hoped that unimportant or redundant parameters are minimized to 0. Similar effects can be obtained when a number of weights in a neural network is shared to reduce the number of free parameters [LeCun et al., 1990, Rumelhart and McClelland, 1986]. It is also possible to remove the least important weights after training. Different approaches to the selection of these weights are possible, like 'optimal brain damage' [LeCun et al., 1990] or the 'optimal brain surgeon' [Hassibi and Stork, 1993].

In the training of neural networks, it is also possible to apply early stopping. In neural networks often sigmoidal transfer functions are used. When small parameter values are used, these transfer functions can be approached well by linear functions. This results in a simple linear combination of inputs for the output of the network, instead of a complex non-linear function of the inputs. This drastically reduces the complexity of the network. Before training the free parameters are initialized with small random values and during training the parameters grow. By stopping the training early, it is avoided that the parameters reach too large values and that the network develops too wild non-linearities and thus, that it overtrains on the data.

Another way to encourage smooth functions f is to directly minimize the curvature in the function. Large fluctuations in the function are therefore actively discouraged. This is done in regularization theory [Smolensky et al., 1996]. A complexity term in regularization is, for instance, the Tikhonov stabilizer:

$$\mathcal{E}_{\text{struct}} = \sum_{k=0}^{K} \int_{x_0}^{x_1} \left\| \frac{\partial^k}{\partial \mathbf{x}^k} f(\mathbf{x};\mathbf{w}) \right\|^2 \mu_k d\mathbf{x} \tag{1.14}$$

Here $\mu_k \geq 0$ for $k = 0, ..., K-1$ and $\mu_k > 0$ for $k = K$ is some weighting function which indicates how smooth the k^{th} derivative of function $f(\mathbf{x};\mathbf{w})$ should be [Haykin, 1999, Bishop, 1995].

In the Bayesian approach the weights $\mathbf{w}$ are not optimized to one particular optimum. Here it is not only assumed that the data is a sample from a data distribution, but also that the weights are a sample from a theoretical weight distribution [MacKay, 1992]. When a classification (or regression) task has to be solved, one tries to find a complete probability density for the weights. For weights $\mathbf{w}$ with a high probability, the function $f(\mathbf{x};\mathbf{w})$ gives a good approximation to the data. When a model is fitted to some data, not only are the most likely weights used in the prediction (the model with the weights with maximal

probability), but the rest of the weight probability density is (in principle) used as well. An important ingredient in the Bayesian approach is the use of a prior probability over the weight space. This results in the regularization of the models (the prior probability of large weights is set small) and this prevents from severe overtraining.

In minimum message length [Wallace and Dowe, 1999] and minimum description length [Rissanen, 1978] one tries not only to minimize the size of the weights $\mathbf{w}$, but tries to directly implement Occam's razor by minimizing the total encoding length of the output given the dataset. To make a short encoding of the output, the output is split into a model and a noise part. The model encodes all regularities in the data. All deviations from the model (which is then considered noise) are encoded separately. To make a short encoding of the model, the model should contain few parameters and these parameters should also have easily encodable values.

Finally, a way of measuring the complexity $\mathcal{E}_{\text{struct}}$ of a classifier is by looking at the worst-case performance of the classifier. When a certain number of arbitrary labeled objects with an arbitrary spatial placing can *always* be classified without errors by a classifier, the classifier has not learned anything from the data. Only when more and more data is available, the classifier really has to model the characteristics in the data. The maximum number of objects which can be learned without error for any arbitrary labelings, gives an indication about the complexity of the classifier, measured by the VC-dimension [Vapnik, 1998]. The VC-dimension measures the complexity of a classifier by counting the maximum number of objects for all possible labelings in a certain feature space that a classifier can separate [Vapnik, 1995].

Although numerous classification functions, errors and optimization routines are available, the 'true' structure in the data is often hard to model completely. One model cannot handle all the 'delicate' details of the data. It appears that the classification performance can often be improved by extending one classifier to a *combination* of classifiers. Using just the optimal classifier to solve a classification problem often disregards valuable information contained in the other, suboptimal classifiers. They may be superior in specific areas in the feature space. It has been shown that when a set of R classifiers is averaged, the variance contribution in the bias-variance decomposition (1.12) decreases by $\frac{1}{R}$ [Tumer and Ghosh, 1996] (the presence of bias in the individual classifiers might still have large negative influence). Finally, this smaller variance contribution results in a smaller expected error.

1.3 One-class classification

Much effort has been expended to solve classification tasks. Numerous methods have been proposed which differ in the function (or model) $f(\mathbf{x}; \mathbf{w})$, the definition of the empirical and structural error $\mathcal{E}_{\text{true}}$ and $\mathcal{E}_{\text{struct}}$, the optimization routine for minimizing (1.10) or (1.13) and the choice of the regularization parameter λ. Although the problem of classification is far from solved in practice, the problem of data description or one-class classification is also of interest.

Basically, the second task is the problem posed at the beginning of this chapter: identify all objects which are neither apples nor pears. The problem in one-class classification is to make a description of a target set of objects and to detect which (new) objects resemble this training set. The difference with conventional classification is that in one-class classification only examples of *one* class are available. The objects from this class will be called the target objects. All other objects are per definition the outlier objects. In figure 1.6 a one-class

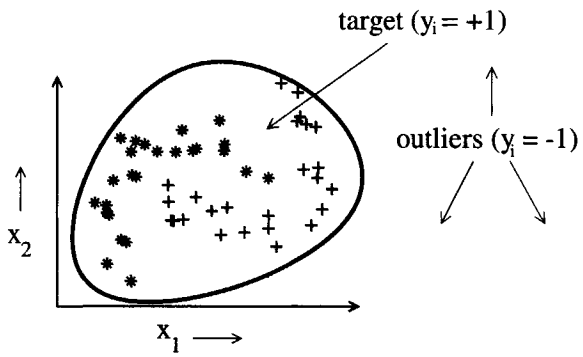


Fig. 1.6: Scatterplot of the apple-pear classification problem. A one-class classifier is shown which distinguishes apples and pears from all other possible objects.

classifier is applied to our small example. The examples of apples and pears are now the target objects and are labeled $y = +1$. All other possible objects (neither apples nor pears) are the outliers $y = -1$. No examples of outlier objects are available. The solid line shows a possible one-class classifier which distinguishes between the apple-pear objects and the outliers.

In the literature a large number of different terms have been used for this problem. The term one-class classification originates from Moya [Moya et al., 1993], but also outlier detection [Ritter and Gallegos, 1997], novelty detection [Bishop, 1994b] or concept learning [Japkowicz, 1999] are used. The different terms originate from the different applications to which one-class classification can be applied.

Obviously, the first application for data description is outlier detection, to detect uncharacteristic objects from a dataset, examples which do not resemble the bulk of the dataset in some way. These outliers in the data can be caused by errors in the measurement of feature values, resulting in an exceptionally large or small feature value in comparison with other training objects. In general, trained classifiers or regressors only provide reliable estimates for input objects resembling the training set. Extrapolations to unknown and remote regions in feature space are very uncertain [Roberts and Penny, 1996]. Neural networks, for instance, can be trained to estimate posterior probabilities [Richard and Lippmann, 1991], [Bishop, 1995], [Ripley, 1996] and tend to give high confidence outputs for objects which are remote from the training set. In these cases an outlier detection should first be used to detect and reject outliers to avoid unfounded confident classifications.

Secondly, data description can be used for a classification problem where one of the classes is sampled very well, while the other class is severely undersampled. The measurements on the undersampled class might be very expensive or difficult to obtain. For instance, in a machine monitoring system where the current condition of a machine is examined, an alarm is raised when the machine shows a problem. Measurements on the normal working conditions of a machine are very cheap and easy to obtain. On the other hand, measurements of outliers would require the destruction of the machine in all possible ways. It is very expensive, if not impossible, to generate all faulty situations [Japkowicz et al., 1995]. Only a method trained on just the target data can solve the monitoring problem.

The last possible use of the outlier detection is the comparison of two data sets. Assume that a classifier has been trained (in a long and difficult optimization process) on some (possibly expensive) data. When a resembling problem has to be solved, the new data set can be compared with the old training set. In case of non-comparable data, the training of a new classifier is needed.

1.4 One-class and two-class classification

The problems we encountered in the conventional classification problems, such as the definition of the error, atypical training data, measuring the complexity of a solution, the curse of dimensionality, the generalization of the method, also appear in one-class classification. Some problems become even more prominent. Because in conventional classification, data from two classes are available, the decision boundary is supported from both sides by example objects. Most conventional classifiers assume more or less equally balanced data classes and do not work well when one class is severely undersampled or even completely absent. Because in one-class classification only one class of data is available, only one side of the boundary can be determined. It is hard to decide on the basis of just one class how tightly the boundary should fit in each of the directions around the data. It is even harder to decide which features should be used to find the best separation of the target and outlier class.

For the computation of the true error $\mathcal{E}_{\text{true}}$, (definition (1.8)) the complete probability density $p(\mathbf{x}, y)$ should be known. In case of one-class classification only $p(\mathbf{x}|\omega_T)$ (the probability density of the target class, ω_T) is known. This means that only the number of target objects which are not accepted by the description, the false negatives, can be minimized. This is called the error of the first kind, $\mathcal{E}_{\text{I}}$. This is trivially satisfied when the data description captures the complete feature space. Without example outlier objects, or an estimate of the outlier distribution $p(\mathbf{x}|\omega_O)$, it is not possible to estimate the number of outlier objects which will be accepted by the description. This number, the number of false positives, is called the error of the second kind $\mathcal{E}_{\text{II}}$.

In table 1.1 all possible situations of classifying an object in one-class classification, are shown. A target object which is accepted by the classifier, is classified correctly. The fraction of target objects which is accepted by the classifier, will be called f_{T+}. When a target object is rejected, it is (erroneously) classified as an outlier object. This contributes

Table 1.1: The four situations of classifying an object in one-class classification. The true positive and the true negative objects do not contribute to the error because they are classified correctly. The false positive and false negative objects are classified wrong. The fraction of target objects which will be classified as target and outlier will be called f_{T+} and f_{T-} respectively. The fraction of the outlier objects which will be classified as target and outlier object, will be called f_{O+} and f_{O-} respectively.

	object from target class	object from outlier class
classified as a target object	true positive, f_{T+}	false positive, f_{O+} $\mathcal{E}_{\text{II}}$
classified as an outlier object	false negative, f_{T-} $\mathcal{E}_{\text{I}}$	true negative, f_{O-}

to the error $\mathcal{E}_{\text{I}}$. The fraction of target objects which is rejected, will be called f_{T-}. The fraction of outlier objects which is classified as target and outlier object will be called f_{O+} and f_{O-} respectively. Error $\mathcal{E}_{\text{II}}$ contains all false positives. Further, note that $f_{T+}+f_{T-}=1$ and that $f_{O-}+f_{O+}=1$. Thus, the main complication in one-class classification is, that only f_{T+} and f_{T-} can be estimated and nothing is known about f_{O+} and f_{O-}.

To avoid the trivial solution of accepting all data, an assumption about the outlier distribution has to be made. In this thesis we will therefore assume that, when no example outliers are available, the outliers are uniformly distributed around the target data. Using the Bayes rule, the posterior probability for the target class can be computed by:

$$p(\omega_T|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_T)p(\omega_T)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|\omega_T)p(\omega_T)}{p(\mathbf{x}|\omega_T)p(\omega_T)+p(\mathbf{x}|\omega_O)p(\omega_O)} \tag{1.15}$$

When it is assumed that $p(\mathbf{x}|\omega_O)$ is independent of $\mathbf{x}$, i.e. it is uniformly distributed in the area of the feature space that we are considering, $p(\mathbf{x}|\omega_T)$ can be used instead of $p(\omega_T|\mathbf{x})$. The $p(\omega_T|\mathbf{x})$ is transformed into $p(\mathbf{x}|\omega_T)$ by a strictly increasing function. So when $p(\omega_T|\mathbf{x}_1) < p(\omega_T|\mathbf{x}_2)$ also $p(\mathbf{x}_1|\omega_T) < p(\mathbf{x}_2|\omega_T)$ holds. The values of $p(\omega_T)$ and $p(\omega_O)$ have to be assumed a priori.

Using a uniform outlier distribution also means that when $\mathcal{E}_{\text{II}}$ is minimized, the data description with minimal volume is obtained. So instead of minimizing both $\mathcal{E}_{\text{I}}$ and $\mathcal{E}_{\text{II}}$, a combination of $\mathcal{E}_{\text{I}}$ and the volume of the description can be minimized to obtain a good data description. Of course, when the true outlier distribution deviates from the uniform

distribution, another data description will show better generalization performance, but this cannot be checked without the use of example outliers. Therefore, the generalization of a method can only be given on the target data.

So, without example outliers, the empirical error $\mathcal{E}_{\text{emp}}$ can only be defined on the target data. To define the empirical error on the outlier data, outlier objects have to be created artificially or a measure of the volume of the data description has to be used. The next problem is to define the structural error $\mathcal{E}_{\text{struct}}$ for a one-class classifier. In conventional classification, smoothness constraints on $f(\mathbf{x};\mathbf{w})$ are often imposed. For one-class classifiers not only smoothness should be enforced, but also constraints for a closed boundary around the data. In general, these extra constraints (which might be a harmful bias for conventional classifiers) make the problem harder, and make other problems, like the curse of dimensionality, more prominent. In one-class classification a boundary should be defined in *all* directions around the data. In particular when the boundary of the data is long and non-convex, the required number of training objects might be very high. So it is to be expected that one-class classification will require a larger sample size in comparison with conventional classification.

1.5 One-class classification methods

For one-class classification several models $f(\mathbf{x};\mathbf{w})$ have been proposed. Most often the methods focus on outlier detection. Conceptually the most simple solution for outlier detection is to generate outlier data around the target set. Then an ordinary classifier is trained to distinguish between the target data and outliers [Roberts et al., 1994]. Koch [Koch et al., 1995] used ART-2A and a Multi-Layered Perceptron for the detection of (partially obscured) objects in an automatic target recognition system. Unfortunately this last method requires the availability of a set of near-target objects (possibly artificial) belonging to the outlier class. The methods scale very poorly in high dimensional problems, especially when the near-target data has to be created and is not readily available. In this thesis we will not use this approach, but we will focus on solutions of one-class classification without the creation of artificial outliers for training. Artificial outliers will be used in the evaluation of the different models though.

In classification or regression problems a more advanced Bayesian approach can be used for detecting outliers ([Bishop, 1995], [MacKay, 1992], [Roberts and Penny, 1996]). Instead of using the most probable weights for a classifier (or regressor) to compute the output, the output is weighted by the probability that the weights for the classifier or regressor is correct, given the data. The classifier outputs are automatically moderated for objects remote from the training domain. These methods are not optimized for outlier detection and they require a classification (or regression) task to be solved. They are therefore very suitable to detect the outlier objects in a classification or regression task. When just a set of objects is available, these methods cannot be used. They also tend to be computationally expensive.

Another possible approach is to use a density method which directly estimates the

density of the target objects $p(\mathbf{x}|\omega_T)$ [Barnett and Lewis, 1978]. By assuming a uniform outlier distribution and by the application of Bayes rule (1.15) the description of the target class is obtained. Only the prior probabilities of the target and outlier class, $p(\omega_T)$ and $p(\omega_O)$ should be chosen beforehand. This directly influences the choice where the probability $p(\omega_T|\mathbf{x})$ should be thresholded to obtain a target and an outlier region. For instance, in [Bishop, 1994b] and [Tarassenko et al., 1995] the density is estimated by a Parzen density estimator. In [Parra et al., 1996] one Gaussian model is used. In [Ritter and Gallegos, 1997] not only the target density is estimated, but also the outlier density. Unfortunately, this procedure requires a complete density estimate in the complete feature space. Especially in high dimensional feature spaces this requires huge amounts of data. Furthermore, it assumes that the training data is a typical sample from the true data distribution. In most cases the user has to generate or measure training data and he (she) might not know beforehand what the true distribution might be. He can only hope to cover the 'normal state' area in the feature space. This makes the application of the density methods problematic. On the other hand, when a large sample of typical data is available, the density method is expected to work well.

In some cases, prior knowledge might be available and the generating process for the objects can be modeled. When it is possible to encode an object $\mathbf{x}$ in the model and to reconstruct the measurements from this encoded object, the reconstruction error can be used to measure the fit of the object to the model. It is assumed that the smaller the reconstruction error, the better the object fits to the model and the more likely that it is not an outlier. These methods will therefore be called the reconstruction methods. Due to the introduction of prior knowledge, it is expected that these methods will work well, and that they will suffer less from poor generalization and low sample size. On the other hand, when the model does not fit the data well, a large bias might be introduced which completely destroys all good characteristics.

Finally, boundary methods have been developed which only focus on the boundary of the data. They try to avoid the estimation of the complete density of the data (which might be impossible from small sample sizes) and therefore also work with an uncharacteristic training data set. These *pure* one-class classification methods are relatively new and are completely focused on the problem of one-class classification (the other methods were mainly used in conventional classification problems). The main advantage of these methods is that they avoid the estimation of the complete probability density. This not only gives an advantage when just a limited sample is available, it is even possible to learn from data when the exact target density distribution is unknown. A user might be able to sample the feature space just very sparsely, without knowledge as to, what more typical examples are and what the exceptions are. For the boundary methods, it is sufficient that the user can indicate just the boundary of the target class by using examples. The user does not have to model or sample the complete distribution. An attempt to train just the boundaries of a data set is made in [Moya and Hush, 1996], [Moya et al., 1993]. Here neural networks are trained with extra constraints to give closed boundaries. Unfortunately, this method inherits the weak points in neural network training, i.e. the choice of the size of the network, weight initialization and the stopping criterion.

Although in principle the boundary methods are more efficient than the density estimation, it is not directly clear how one should define a boundary around a target set $\mathcal{X}^{tr}$, how to define the resemblance of an object $\mathbf{x}$ to a target set $\mathcal{X}^{tr}$ and where to put the threshold. In most cases a distance $d(\mathbf{x})$ to the target set is defined which is a function of (Euclidean) distances between objects, between the test object and the target objects, and between the target objects themselves (for instance in [Knorr et al., 2000], where it is used for finding outliers in large databases). This requires well defined distances in the feature space and thus well-scaled features. We will encounter this type of method in the next chapter of this thesis.

1.6 Outlook of this thesis

In this chapter we introduced the problem of one-class classification and how it relates to conventional two-class classification. In the rest of this thesis we will discuss the problems in the construction and the training of one-class classifiers in more detail. We will present some possible one-class models and we will compare them as well. In chapter 2 we start with a new one-class classification method which directly finds a boundary around the data and which minimizes the volume of the description. It does not depend on a density estimate of the data, thus making it more resistant to the curse of dimensionality and able to cope with atypical training data.

In chapter 3 several other (simple) models for one-class classification are investigated. Most models are well-known in pattern recognition, but are sometimes adapted to cope with the one-class classification problem. The three types of models will be treated: the density estimators, the reconstruction methods and the boundary methods. The methods differ in the definition of the function $f(\mathbf{x};\mathbf{w})$ and the error $\mathcal{E}$ and in the minimization method. The functions implicitly assume different characteristics in the data, and might therefore show different generalization, different biases over different one-class problems and different overfitting behavior. Their performance will be investigated and compared in chapter 4 on several simple artificial datasets and some real world datasets.

Finally, in chapter 5 the possibilities of combining several one-class classifiers will be investigated. It is well known that combining the results of conventional classifiers can significantly improve performance in conventional classification problems. Due to the different nature of one-class classifiers, it will be investigated how far these characteristics are preserved in the combination of one-class classifiers.

2. SUPPORT VECTOR DATA DESCRIPTION

In this chapter we present and examine a method for directly obtaining the boundary around a target data set. In the most simple case a hypersphere is computed which contains all target objects. To minimize the chance of accepting outliers, the volume of this hypersphere is minimized. The model can be rewritten in a form comparable to the support vector classifier (SVC)[Vapnik, 1995], and it will therefore be called the support vector data description (SVDD). It offers the ability to map the data to a new, high dimensional feature space without much extra computational costs. By this mapping more flexible descriptions are obtained. It will be shown how the outlier sensitivity can be controlled in a flexible way. Finally, an extra option is added to include example outliers into the training procedure (when they are available) to find a more efficient description.

In sections 2.1, 2.2 and 2.3 we present the basic theory, which is already partly presented in [Tax and Duin, 1999]. It contains the normal data description, the description using negative examples and an introduction to the use of kernels. In section 2.4 we discuss some special properties of the SVDD which makes it possible to estimate the expected error on the target set and which makes it possible to determine the values of some free parameters in the method. In section 2.5 we will show that the SVDD gives solutions similar to the hyperplane approach of [Schölkopf et al., 1999]. Other characteristics, like the required number of examples, and the efficiency of the removal of outliers, will be shown in section 2.6. In section 2.7 the method will be applied to a real life problem and will be compared with some other (simple) one-class classification methods. Section 2.8 contains some conclusions.

2.1 Spherical data description

To start, we define a model $f(\mathbf{x};\mathbf{w})$ which gives a closed boundary around the data: a hypersphere. The sphere is characterized by a center $\mathbf{a}$ and radius R (figure 2.1) and we demand that the sphere contains all training objects $\mathcal{X}^{tr}$.[1] When we demand that the sphere contains all training objects $\mathcal{X}^{tr}$ the empirical error is set to 0. So, analogous to

[1] This is identical to the approach which is used in [Burges, 1998] to estimate the VC-dimension of a classifier which is bounded by the diameter of the smallest sphere enclosing the data.

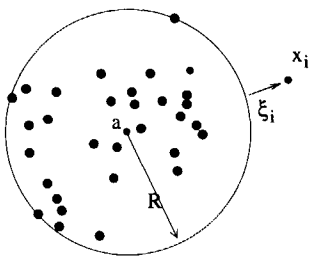


Fig. 2.1: The hypersphere containing the target data, described by the center $\mathbf{a}$ and radius R. Three objects are on the boundary, the support vectors. One object $\mathbf{x}_i$ is outside and has $\xi_i > 0$.

the support vector classifier [Vapnik, 1998], we define the structural error:

$$\mathcal{E}_{\text{struct}}(R, \mathbf{a}) = R^2 \tag{2.1}$$

which has to be minimized with the constraints:

$$\|\mathbf{x}_i - \mathbf{a}\|^2 \leq R^2, \qquad \forall i \tag{2.2}$$

To allow the possibility of outliers in the training set, and therefore to make the method more robust, the distance from objects $\mathbf{x}_i$ to the center $\mathbf{a}$ should not be strictly smaller than R^2, but larger distances should be penalized. This means that the empirical error does not have to be 0 by definition. In figure 2.1 an example of a data description is given. It shows one object which is rejected by the description. The error now contains both a structural and an empirical error contribution. We introduce slack variables $\boldsymbol{\xi}, \xi_i \geq 0, \forall_i$ and the minimization problem changes into:

$$\mathcal{E}(R, \mathbf{a}, \boldsymbol{\xi}) = R^2 + C \sum_i \xi_i \tag{2.3}$$

with constraints that (almost) all objects are within the sphere:

$$\|\mathbf{x}_i - \mathbf{a}\|^2 \leq R^2 + \xi_i, \qquad \xi_i \geq 0, \qquad \forall i \tag{2.4}$$

The parameter C gives the tradeoff between the volume of the description and the errors.

The free parameters, $\mathbf{a}$, R and $\boldsymbol{\xi}$, have to be optimized, taking the constraints (2.4) into account. Constraints (2.4) can be incorporated into formula (2.3) by introducing Lagrange multipliers and constructing the Lagrangian [Strang, 1988]:

$$\begin{aligned} L(R, \mathbf{a}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\gamma}) = R^2 + C \sum_i \xi_i \\ - \sum_i \alpha_i \{R^2 + \xi_i - (\mathbf{x}_i \cdot \mathbf{x}_i - 2\mathbf{a} \cdot \mathbf{x}_i + \mathbf{a} \cdot \mathbf{a})\} - \sum_i \gamma_i \xi_i \end{aligned} \tag{2.5}$$

with the Lagrange multipliers $\alpha_i \geq 0$ and $\gamma_i \geq 0$, and where $\mathbf{x}_i \cdot \mathbf{x}_j$ stands for the inner product between $\mathbf{x}_i$ and $\mathbf{x}_j$. Note that for each object $\mathbf{x}_i$ a corresponding α_i and γ_i are defined. L has to be minimized with respect to $R, \mathbf{a}$ and $\boldsymbol{\xi}$, and maximized with respect to $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$.

Setting partial derivatives to 0 gives the constraints:

$$\sum_i \alpha_i = 1 \tag{2.6}$$

$$\mathbf{a} = \sum_i \alpha_i \mathbf{x}_i \tag{2.7}$$

$$C - \alpha_i - \gamma_i = 0, \qquad \forall_i \tag{2.8}$$

The last constraint can be rewritten into an extra constraint for $\boldsymbol{\alpha}$ (see appendix A on page 151 for a more elaborate explanation):

$$0 \leq \alpha_i \leq C, \qquad \forall_i \tag{2.9}$$

This results in the final error L:

$$L = \sum_i \alpha_i (\mathbf{x}_i \cdot \mathbf{x}_i) - \sum_{i,j} \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j) \tag{2.10}$$

with $0 \leq \alpha_i \leq C$.

The minimization of this error with the constraints is a well-known problem; it is called a quadratic programming problem. For the optimization of (2.10) with constraints (2.9) standard algorithms exist. These find the optimal values for the Lagrange multipliers $\boldsymbol{\alpha}$.

By the Lagrange formulation of the problem, further interpretation of the values of $\boldsymbol{\alpha}$ can be given. When an object $\mathbf{x}_i$ satisfies the inequality in constraint (2.4), the corresponding Lagrange multiplier will be equal to 0 ($\alpha_i = 0$). For objects satisfying the equality $\|\mathbf{x}_i - \mathbf{a}\|^2 = R^2 + \xi_i$ (the object is located at or outside the boundary), the constraint has to be enforced and the Lagrange multiplier will become unequal 0 (i.e. positive, $\alpha_i > 0$). The upper bound in inequality (2.9) limits the influence of objects on the final solution. When an object obtains $\alpha_i = C$, the object is regarded as an outlier and will not be accepted by the data description. An object with $\alpha_i > 0$ can therefore be on the boundary or outside the boundary.

In equation (2.7), the center of the sphere $\mathbf{a}$ is expressed as a linear combination of objects with weights α_i. Therefore, for the computation of $\mathbf{a}$ objects with 0 weight ($\alpha_i = 0$) can be disregarded. Only objects with positive weight $\alpha_i > 0$ are needed in the description of the data set. It turns out that in the minimization of (2.10), often a large fraction of the weights becomes 0. The sum in equation (2.7) is then over just a few objects $\mathbf{x}_i$ with non-zero α_i. These objects will be called the *support objects* of the description or the *support vectors* (SVs).

Because we are able to give an expression for the center of the hypersphere $\mathbf{a}$, we can test if a new object $\mathbf{z}$ is accepted by the description. For that, the distance from the object

$\mathbf{z}$ to the center of the hypersphere $\mathbf{a}$ has to be calculated. A test object $\mathbf{z}$ is accepted when this distance is smaller than or equal to the radius:

$$\|\mathbf{z}-\mathbf{a}\|^2 = (\mathbf{z}\cdot\mathbf{z}) - 2\sum_i \alpha_i(\mathbf{z}\cdot\mathbf{x}_i) + \sum_{i,j}\alpha_i\alpha_j(\mathbf{x}_i\cdot\mathbf{x}_j) \leq R^2 \tag{2.11}$$

By definition, R^2 is the (squared) distance from the center of the sphere $\mathbf{a}$ to one of the support vectors on the boundary:

$$R^2 = (\mathbf{x}_k\cdot\mathbf{x}_k) - 2\sum_i \alpha_i(\mathbf{x}_i\cdot\mathbf{x}_k) + \sum_{i,j}\alpha_i\alpha_j(\mathbf{x}_i\cdot\mathbf{x}_j) \tag{2.12}$$

for any $\mathbf{x}_k \in SV^{\text{bnd}}$, i.e. the set of support vectors for which $0 < \alpha_k < C$.

We will call this one-class classifier the support vector data description (SVDD). It can now be written as:

$$\begin{aligned} f_{SVDD}(\mathbf{z};\boldsymbol{\alpha},R) &= I\left(\|\mathbf{z}-\mathbf{a}\|^2 \leq R^2\right) \\ &= I\left((\mathbf{z}\cdot\mathbf{z}) - 2\sum_i \alpha_i(\mathbf{z}\cdot\mathbf{x}_i) + \sum_{i,j}\alpha_i\alpha_j(\mathbf{x}_i\cdot\mathbf{x}_j) \leq R^2\right) \end{aligned} \tag{2.13}$$

where the indicator function I is defined as:

$$I(A) = \begin{cases} 1 & \text{if } A \text{ is true,} \\ 0 & \text{otherwise.} \end{cases} \tag{2.14}$$

Note that in formulae (2.10), (2.12) and (2.13) objects $\mathbf{x}$ only appear in the form of inner products with other objects $\mathbf{y}$: $(\mathbf{x}\cdot\mathbf{y})$. Analogous to [Vapnik, 1998] the inner products can be replaced by a kernel function to obtain more flexible descriptions. We will investigate this further in section 2.3.

The figure 2.2 shows a SVDD for a small 2-dimensional banana-shaped data set. The objects are plotted with pluses, the gray value indicates the distance to the center of the sphere (dark is close, light is remote). The solid circles indicate the support vectors, the dashed line is the boundary of the data description. Only the 3 support objects are required to describe the complete data set.

In this section we have shown the basic hypersphere model of the SVDD. How can we optimize the placing of the center of the hypersphere, and how can we evaluate a novel object. In the next section we expand this hypersphere model to include example *outlier* objects in the training.

2.2 Data description with negative examples

When negative examples (objects which should be rejected) are available, they can be used during the training to improve the description (i.e. to obtain a tighter boundary around the

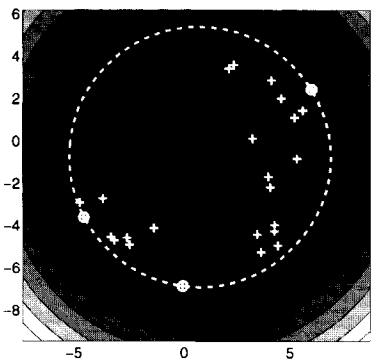


Fig. 2.2: Example of a data description.

data in the areas where outlier objects are present). In contrast with the target examples, which should be within the sphere, the negative examples should be *outside* it.

Assume that we train a SVDD on examples of two classes. One of the classes is considered the target class (and thus the other class is the outlier class). It is possible to train both a SVDD and a traditional (two-class) classifier on this data. The SVDD differs from a conventional classifier because the SVDD satisfies the constraint that it always obtains a closed boundary around one of the classes (the target class). Furthermore, it does not require a strict representative sample of the target distribution; a sample with (in some way) extreme objects is also acceptable. This is made explicit in the definition of the error of the SVDD: the SVDD minimizes the volume of the description plus the sum of the distances ξ_i the objects $\mathbf{x}_i$ are outside the description.

A conventional classifier, on the other hand, distinguishes between two (or more) classes without special focus on any of the classes. A classifier minimizes the probability of classification error. The conventional classifier is expected to perform very poorly when just a few outlier examples are available and the outlier class is extremely undersampled. When both a representative sample from the target class and a large amount of example outliers is available and when it is assumed that these objects are independently drawn from the same target and outlier distributions,[2] an ordinary two-class classification problem is obtained. In that case the conventional classifier will work better than the SVDD. The SVDD is then limited by the constraint of a closed boundary around the data.

The choice between a SVDD and an ordinary classifier is therefore influenced by both the number of outlier objects available for training and how well they represent the target and the outlier distributions. In the rest of the thesis we will assume that just a few outliers are present in the training set (let's say less than 5% of the training objects). Only in section 2.6.4 extra outliers are introduced to measure the robustness against outliers.

In this section the target objects are enumerated by indices i, j and the negative exam-

[2] This is called i.i.d. data: independent and identically distributed data.

ples by l, m. For further convenience we assume that target objects are labeled $y_i = 1$ and outlier objects are labeled $y_i = -1$. Again we allow for errors in the target and the outlier set and introduce slack variables $\xi_i \geq 0$ and $\xi_l \geq 0$:

$$\mathcal{E}(R, \mathbf{a}, \boldsymbol{\xi}) = R^2 + C_1 \sum_i \xi_i + C_2 \sum_l \xi_l \tag{2.15}$$

and the constraints

$$\|\mathbf{x}_i - \mathbf{a}\|^2 \leq R^2 + \xi_i, \qquad \|\mathbf{x}_l - \mathbf{a}\|^2 > R^2 - \xi_l, \qquad \xi_i \geq 0, \xi_l \geq 0 \quad \forall i, l \tag{2.16}$$

(note that objects with $\xi_i > 0$ are the false negatives, objects with $\xi_l > 0$ are false positives.)

These constraints are incorporated in formula (2.15) and by introducing the Lagrange multipliers $\alpha_i, \alpha_l, \gamma_i, \gamma_l$, we obtain:

$$\begin{aligned} L(R, \mathbf{a}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\gamma}) = R^2 + C_1 \sum_i \xi_i + C_2 \sum_l \xi_l - \sum_i \gamma_i \xi_i - \sum_l \gamma_l \xi_l \\ - \sum_i \alpha_i \left[R^2 + \xi_i - \|\mathbf{x}_i - \mathbf{a}\|^2\right] - \sum_l \alpha_l \left[\|\mathbf{x}_l - \mathbf{a}\|^2 - R^2 + \xi_l\right] \end{aligned} \tag{2.17}$$

with $\alpha_i \geq 0, \alpha_l \geq 0, \gamma_i \geq 0, \gamma_l \geq 0$.

As in equations (2.6), (2.7) and (2.8), the partial derivatives of L with respect to R, $\mathbf{a}$ and ξ_i (ξ_l) are set to 0. This results in the new constraints:

$$\sum_i \alpha_i - \sum_l \alpha_l = 1 \tag{2.18}$$

$$\mathbf{a} = \sum_i \alpha_i \mathbf{x}_i - \sum_l \alpha_l \mathbf{x}_l \tag{2.19}$$

$$0 \leq \alpha_i \leq C_1, \qquad 0 \leq \alpha_l \leq C_2 \qquad \forall i, l \tag{2.20}$$

When formulae (2.18), (2.19) and (2.20) are resubstituted into (2.17) we obtain

$$\begin{aligned} L = \sum_i \alpha_i (\mathbf{x}_i \cdot \mathbf{x}_i) - \sum_l \alpha_l (\mathbf{x}_l \cdot \mathbf{x}_l) - \sum_{i,j} \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j) \\ + 2 \sum_{l,j} \alpha_l \alpha_j (\mathbf{x}_l \cdot \mathbf{x}_j) - \sum_{l,m} \alpha_l \alpha_m (\mathbf{x}_l \cdot \mathbf{x}_m) \end{aligned} \tag{2.21}$$

The notation can now be simplified when new variables α_i' are defined which include the labels $y_i = \pm 1$:

$$\alpha_i' = y_i \alpha_i \tag{2.22}$$

Index i now enumerates both target and outlier objects. Using α_i' the SVDD with negative examples becomes identical to the original SVDD (formula (2.10)). The first two terms in (2.21) collapse to the first term in (2.10), the last three terms become the second term in (2.10). The constraints given in formulae (2.18) and (2.19) change into $\sum_i \alpha_i' = 1$ and $\mathbf{a} =$

$\sum_i \alpha_i' \mathbf{x}_i$ and again function f_{SVDD} (2.13) can be used. Therefore, when outlier examples are available, we will use α_i' instead of α_i in the optimization and the evaluation. So, it appears that the introduction of example outliers does not result in extra computational complications for the SVDD. Only the sample size increases, which slows the quadratic optimization.

In the optimization routine different regularization values C_1 and C_2 can be given for the target and the outlier objects. Choosing different values for C_1 and C_2 incorporates different costs for false negatives (target objects which are rejected or 'error of the first kind') and false positives (outlier objects which are accepted, also 'error of the second kind'). Then by the ratio:

$$\frac{C_1}{C_2} = \frac{\text{error kind I}}{\text{error kind II}} = \frac{\mathcal{E}_{\text{I}}}{\mathcal{E}_{\text{II}}} \tag{2.23}$$

the relative values of C_1 and C_2 can be obtained. In section 2.4 a method to obtain absolute values for C_1, C_2 will be given.

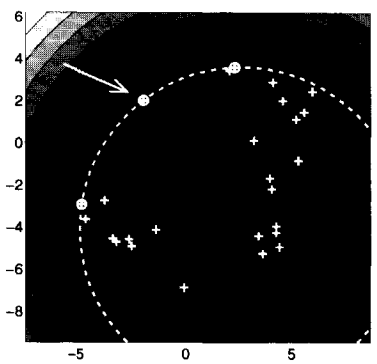


Fig. 2.3: Example of a data description with one outlier indicated by the arrow.

In figure 2.3 the same data set is shown as in figure 2.2, extended with one outlier object (indicated with the arrow). This outlier lies within the original data description on the left. A new description has to be computed to reject this outlier. When the outlier would is outside the original data description, the corresponding α_l becomes 0 and the computation of the center $\mathbf{a}$ (2.19) becomes identical to that of the SVDD without outlier objects (2.7). Thus introducing an outlier in the training, which is remote from the target data, does not change or harm the solution.

When a new outlier object is placed within the original data description (a data description not trained with this outlier), the outlier is placed on the boundary of the description. This will require a minimal adjustment to the old description and to minimize the volume of the description. This is shown in figure 2.3. Except for this single object, this dataset is identical to the data in figure 2.2. This object now becomes a support vector for the

outlier class and it cannot be distinguished from the support vectors from the target class on the basis of (2.13).

In this section we extended the SVDD to reject also some example outlier objects. Although the description is adjusted to reject the outlier objects, it does not fit tightly around the rest of the target set anymore. A more flexible description is therefore required. This is what we will show in the next section.

2.3 Flexible descriptions

The hypersphere is a very rigid model of the boundary of the data. In general, it cannot be expected that this model will fit the data well. If we can map the data to a new representation, we might obtain a better fit between the actual data boundary and the hypersphere model. Assume we are given a mapping Φ of the data which improves this fit:

$$\mathbf{x}^* = \Phi(\mathbf{x}) \tag{2.24}$$

We can apply this mapping in (2.10) and (2.13) and we obtain:

$$L = \sum_i \alpha_i \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_i) - \sum_{i,j} \alpha_i \alpha_j \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \tag{2.25}$$

and

$$f_{SVDD}(\mathbf{z}; \boldsymbol{\alpha}, R) = I\left(\Phi(\mathbf{z}) \cdot \Phi(\mathbf{z}) - 2\sum_i \alpha_i \Phi(\mathbf{z}) \cdot \Phi(\mathbf{x}_i) + \sum_{i,j} \alpha_i \alpha_j \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \leq R^2\right) \tag{2.26}$$

It can be recognized that in these formulae all mappings $\Phi(\mathbf{x})$ occur only in inner products with other mappings. It is possible to define a new function of two input variables, called a kernel function:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \tag{2.27}$$

and replace all occurrences of $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$ by this kernel. Because this kernel can be written as an inner product of two functions, it is called a Mercer kernel. This results in:

$$L = \sum_i \alpha_i K(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i,j} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{2.28}$$

and

$$f_{SVDD}(\mathbf{z}; \boldsymbol{\alpha}, R) = I\left(K(\mathbf{z}, \mathbf{z}) - 2\sum_i \alpha_i K(\mathbf{z}, \mathbf{x}_i) + \sum_{i,j} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \leq R^2\right) \tag{2.29}$$

In this formulation, the mapping Φ is never used explicitly, but it is only defined implicitly by the kernel K. A good kernel function should be defined, i.e. a kernel which maps the target data onto a bounded, spherically shaped area in the feature space and outlier objects outside this area. Then the hypersphere model fits the data and good classification performance is obtained.

The technique to replace $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$ by $K(\mathbf{x}_i, \mathbf{x}_j)$ is found by Vapnik [Vapnik, 1998] and is called the 'kernel trick'. The trick is used in the support vector classifier when the classes are not linearly separable. Then the data is mapped to another feature space where the data is, in fact, linearly separable. The advantage of the 'kernel trick' is that the introduction of the kernels does not introduce much extra computational costs. The optimization problem remains identical in the number of free parameters. The only extra cost is in the computation of the kernel functions $K(\mathbf{x}_i, \mathbf{x}_j)$. [3]

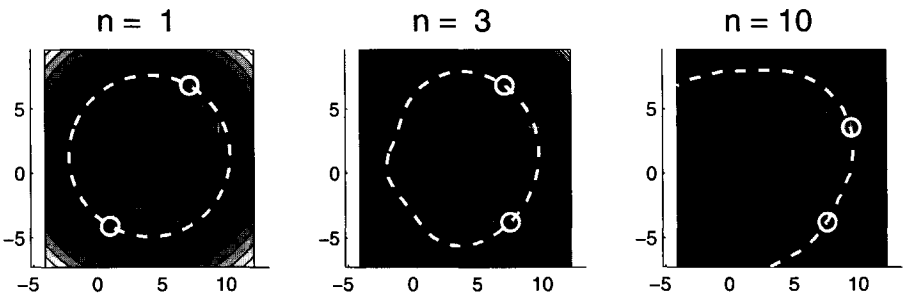


Fig. 2.4: Data description trained on a banana-shaped data set. The kernel is a polynomial kernel with different degrees, $n = 1$, $n = 3$ and $n = 10$. Support vectors are indicated by the solid circles; the dashed line is the description boundary.

[3] Although we have stated the problem in terms of inner products between objects, it is also possible to rephrase it in terms of distances between the objects. The squared (Euclidean) between two objects can be written as:

$$\|\mathbf{x}_i, \mathbf{x}_j\|^2 = \mathbf{x}_i \cdot \mathbf{x}_i + \mathbf{x}_j \cdot \mathbf{x}_j - 2\mathbf{x}_i \cdot \mathbf{x}_j = \|\mathbf{x}_i - \mathbf{0}\|^2 + \|\mathbf{x}_j - \mathbf{0}\|^2 - 2\mathbf{x}_i \cdot \mathbf{x}_j \tag{2.30}$$

where $\|\mathbf{x}_i - \mathbf{0}\|^2$ is the squared distance from object $\mathbf{x}_i$ to the origin. Using this equation we can give $\mathbf{x}_i \cdot \mathbf{x}_j$ in terms of distances $\|.\|^2$. For the SVDD this means that (2.28) and (2.29) can be rewritten to:

$$L = \sum_i \alpha_i \alpha_j \|\mathbf{x}_i - \mathbf{x}_j\|^2 \tag{2.31}$$

$$f_{SVDD}(\mathbf{z}; \boldsymbol{\alpha}, R) = I\left(\sum_i \alpha_i \|\mathbf{z} - \mathbf{x}_i\|^2 - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j \|\mathbf{x}_i - \mathbf{x}_j\|^2 \leq R^2\right) \tag{2.32}$$

Thus, instead of changing the definition of the inner product by introducing the kernel function (which might not be very intuitive), the distance definition is changed. The reader is free to choose either of these interpretations.

Several kernel functions have been proposed for the support vector classifier [Smola et al., 1998]. For some of the kernels K, explicit mappings Φ can be reconstructed. It appears that, for the support vector data description, not all kernel functions are equally useful. We will investigate the two most important kernel functions, the polynomial kernel and the Gaussian kernel.

The polynomial kernel is given by:

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^n \tag{2.33}$$

where the free parameter n gives the degree of the polynomial kernel. This kernel creates extra features with terms in $\mathbf{x}_i$ and $\mathbf{x}_j$ of all orders from 0 up to n. For the polynomial kernel the mapping Φ can be given explicitly (for small dimensionalities d and degrees n). For instance, for $d = 3$ and $n = 2$:

$$\Phi(\mathbf{x}) = (1, x_1, x_2, x_3, x_1 \cdot x_2, x_1 \cdot x_3, x_2 \cdot x_3, x_1^2, x_2^2, x_3^2)^T \tag{2.34}$$

Then $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$ reduces to expression in (2.33). All orders from 0-th to n-th order are included. Already for these very small values for d and n, a total of 10 features appear. In general, the number of features becomes [Burges, 1998] (for homogeneous polynomial kernels $(\mathbf{x}_i \cdot \mathbf{x}_j)^n$):

$$n^* = \frac{(n+d)!}{d!\,n!} \tag{2.35}$$

which explodes for larger n and d. The introduction of the kernel avoids the explicit computation of all these features, while still retaining the characteristics of the representation.

One of the characteristics of the polynomial kernel greatly influences the performance of the support vector data description. Recall that $\mathbf{x}_i \cdot \mathbf{x}_j = \cos(\theta_{ij})\|\mathbf{x}_i\| \cdot \|\mathbf{x}_j\|$, where θ_{ij} is the angle between the object vectors $\mathbf{x}_i$ and $\mathbf{x}_j$. When data is not centered around the origin, object vectors become large and the angles between them become small. Then for larger degrees n, $\cos(\theta_{ij}) \sim 1$ and the polynomial kernel can be approximated by:

$$(\mathbf{x}_i \cdot \mathbf{x}_j)^n = \cos^n(\theta_{ij})\|\mathbf{x}_i\|^n \cdot \|\mathbf{x}_j\|^n \simeq \|\mathbf{x}_i\|^n \cdot \|\mathbf{x}_j\|^n \tag{2.36}$$

Equation (2.36) looses the sensitivity for θ_{ij} in the neighborhood of the training data (where θ_{ij} becomes small). The objects with the largest norm in the training set will overwhelm all other terms in the polynomial kernel.

In figure 2.4 the influence of large norm objects is shown. For a simple 2-dimensional data set descriptions are obtained using a polynomial kernel with different degrees, for $n = 1$ (left), $n = 3$ (middle) and $n = 10$ (right). Again the solid circles indicate the support vectors; the dashed line is the description boundary mapped in the input space. The rigid spherical description is obtained for $n = 1$. For degree $n = 10$ the description is a 10-th order polynomial. The training objects most remote from the origin (the objects on the right) become support objects and the data description only distinguishes on the basis of the norm of the vectors. Large regions in the input space without target objects will still be accepted by the description.

What can be done to suppress this divergence of the training objects by this polynomial mapping? Centering the data in the original input space, or in the new feature space, by subtracting the averaged mapped $\mathbf{x}$ (as explained in [Schölkopf, 1997]) does not resolve the problem of the large differences in vector norms. Assume that the explicit mapping Φ is available, we can compute the mean Φ_μ in the feature space and map the data as follows:

$$\mathbf{x}^* = \Phi(\mathbf{x}) - \frac{1}{N}\sum_i \Phi(\mathbf{x}_i) = \Phi(\mathbf{x}) - \Phi_\mu \tag{2.37}$$

This can directly be substituted into (2.28) and rewriting gives (liberally using (2.6) or (2.18)):

$$\begin{aligned} L &= \sum_i \alpha_i(\mathbf{x}_i^* \cdot \mathbf{x}_i^*) - \sum_{i,j} \alpha_i\alpha_j(\mathbf{x}_i^* \cdot \mathbf{x}_j^*) && (2.38)\\ &= \sum_i \alpha_i \left[\Phi(\mathbf{x}_i)\cdot\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_i)\Phi_\mu - \Phi_\mu\sum_i \Phi(\mathbf{x}_i) + \Phi_\mu^2\right] \\ &\quad - \sum_{ij} \alpha_i\alpha_j \left[\Phi(\mathbf{x}_i)\cdot\Phi(\mathbf{x}_j) - \Phi(\mathbf{x}_i)\Phi_\mu - \Phi_\mu\sum_j \Phi(\mathbf{x}_j) + \Phi_\mu^2\right] && (2.39) \end{aligned}$$

$$\begin{aligned} &= \sum_i \alpha_i\Phi(\mathbf{x}_i)\cdot\Phi(\mathbf{x}_i) - \Phi_\mu\sum_i \Phi(\mathbf{x}_i) - \Phi_\mu\sum_j \Phi(\mathbf{x}_j) + \Phi_\mu\cdot\Phi_\mu \\ &\quad - \left[\sum_{ij} \alpha_i\alpha_j\Phi(\mathbf{x}_i)\cdot\Phi(\mathbf{x}_j) - \Phi_\mu\sum_i \alpha_i\Phi(\mathbf{x}_i) - \Phi_\mu\sum_j \alpha_j\Phi(\mathbf{x}_j) + \Phi_\mu\cdot\Phi_\mu\right] \\ &= \sum_i \alpha_i K(\mathbf{x}_i,\mathbf{x}_i) - \sum_{i,j} \alpha_i\alpha_j K(\mathbf{x}_i,\mathbf{x}_j) && (2.40) \end{aligned}$$

Also in the computation of the distance to the center of the sphere, the original formulation reappears:

$$\|\mathbf{z}^* - \mathbf{a}^*\|^2 = ((\Phi(\mathbf{z}) - \Phi_\mu) - (\Phi(\mathbf{a}) - \Phi_\mu))^2 = (\Phi(\mathbf{z}) - \Phi(\mathbf{a}))^2$$

This shows that the centered SVDD (both in the original input space and in the high dimensional feature space introduced by Φ) is equivalent to the original SVDD.

Although shifting of the dataset does not suppress the influence of the large norms (which was introduced by the polynomial kernel), rescaling all feature directions in the data to unit variance can help. In figure 2.5 the rescaled version of the data from figure 2.4 is shown. For $n = 1$ identical solutions are obtained. For higher n, especially for $n = 10$, a tighter description is obtained then in figure 2.4, but still the objects with largest norm determine the description.

Note also that rescaling to unit variance might magnify the influence of noise. High dimensional data often contains some singular directions in which the variance of the

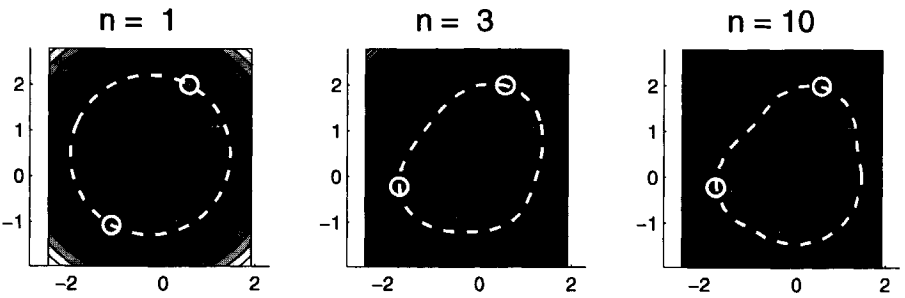


Fig. 2.5: Data description trained on a banana-shaped data set. The kernel is a polynomial kernel with three different degrees, $n = 1$, $n = 3$ and $n = 10$. Support vectors are indicated by the solid circles; the dashed line is the description boundary. The data is the same as presented in figure 2.4, only it is scaled to unit variance.

data is very small. The only contribution to the variance in these directions is due to noise. Rescaling to unit variance might therefore deteriorate the data enormously. We can conclude that, because the polynomial kernel stretches the data in the high dimensional feature space, the data is very hard to describe by a hypersphere. We either have to scale the data very carefully, or we have to use another kernel definition.

Next, we investigate the Gaussian kernel, given by:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{s^2}\right) \tag{2.41}$$

This kernel is independent of the position of the data set with respect to the origin, it only utilizes the distances $\|\mathbf{x}_i - \mathbf{x}_j\|$ between objects. For the Gaussian kernel no finite mapping $\Phi(\mathbf{x})$ of object $\mathbf{x}$ can be given. The fact that $K(\mathbf{x}_i, \mathbf{x}_i) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_i) = 1$ means that the mapped object $\mathbf{x}_i^* = \Phi(\mathbf{x}_i)$ has norm equal to 1. When objects $\mathbf{x}_i$ and $\mathbf{x}_j$ are far apart in the original input space (with respect to s) $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^* \cdot \mathbf{x}_j^*) = 0$, and this means that in the high dimensional feature space the object vectors are perpendicular. Because an infinite number of new objects can be added (with $K(\mathbf{x}_i, \mathbf{x}_j) \simeq 0$), the kernel space can be infinitely extended (and thus can be infinitely large). It appears that the data is mapped on a unit hypersphere in an infinite dimensional feature space.

Because $K(\mathbf{x}_i, \mathbf{x}_i) = 1$, both the Lagrangian (2.28) and the evaluation function (2.29) simplify. Now we have to maximize the Lagrangian (ignoring constants):

$$L = -\sum_{i,j} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{2.42}$$

still retaining the constraints $\sum_i \alpha_i = 1$ and $0 \leq \alpha_i \leq C$.

To evaluate if an object $\mathbf{z}$ is accepted by the data description, formula (2.29) can be

rewritten as:

$$f_{SVDD}(\mathbf{z};\boldsymbol{\alpha},R) = I\left(\sum_i \alpha_i \exp\left(\frac{-\|\mathbf{z}-\mathbf{x}_i\|^2}{s^2}\right) > \frac{1}{2}(B-R^2)\right) \tag{2.43}$$

where $B = 1 + \sum_{i,j} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j)$ only depends on the support vectors $\mathbf{x}_i$ and not on the object $\mathbf{z}$. Note that this evaluation function is basically a thresholded mixture of Gaussians (the left side in the inequality in (2.43) is a weighted sum of Gaussians). The character of the data description heavily depends on the value which is chosen for the width parameter s. We can distinguish three types of solution, for very small s, very large s and for intermediate values.

For small values of s, i.e. for values smaller than the average nearest neighbor distance in the data (or in the order $\min_{i,j} \|\mathbf{x}_i - \mathbf{x}_j\|$), all cross terms in formula (2.42) become small:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{s^2}\right) \to 0, \quad \forall_{i \neq j} \tag{2.44}$$

For sufficiently small s, differences between $K(\mathbf{x}_i, \mathbf{x}_j)$ for different i and j can be ignored, and taking into account the constraint that $\sum_i \alpha_i = 1$, Lagrangian (2.42) is minimized when all objects become support objects with equal $\alpha_i = \frac{1}{N}$. [4] Each individual object now supports a small, equally weighted Gaussian, and the total model is a sum of all these Gaussians. This is identical to a Parzen density estimation with a small width parameter [Duda and Hart, 1973].

For very large s, i.e. values in the order of the maximum distance between objects ($\max_{i,j} \|\mathbf{x}_i - \mathbf{x}_j\|$), the SVDD solution (2.43) approximates the original spherically shaped solution (polynomial kernel with degree $n = 1$). This can be seen from a Taylor expansion of the Gaussian kernel:

$$\begin{aligned} K(\mathbf{x}_i, \mathbf{x}_j) &= \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/s^2) \\ &= 1 - \mathbf{x}_i^2/s^2 - \mathbf{x}_j^2/s^2 + 2(\mathbf{x}_i \cdot \mathbf{x}_j)/s^2 + \ldots \end{aligned} \tag{2.45}$$

Substituting this into Lagrangian (2.42) we obtain:

$$\begin{aligned} L &= -\sum_{i,j} \alpha_i \alpha_j \left(1 - \mathbf{x}_i^2/s^2 - \mathbf{x}_j^2/s^2 + 2(\mathbf{x}_i \cdot \mathbf{x}_j)/s^2 + \ldots\right) \\ &= -1 + 2\sum_i \alpha_i \mathbf{x}_i^2/s^2 - 2\sum_{i,j} \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j)/s^2 + \ldots \end{aligned} \tag{2.46}$$

[4] The solution $\alpha_k = 1$ and $\alpha_i = 0, \forall_{i \neq k}$ has an even smaller error, but this is an unstable optimum. Computing the first derivative of L (2.42) shows that $\frac{\partial L}{\partial \alpha_i} = -2\sum_j \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) < 0, \forall_i$. If we start with $\alpha_k = 1$ and $\alpha_i = 0, \forall_{i \neq k}$, then $\frac{\partial L}{\partial \alpha_i} = -2K(\mathbf{x}_i, \mathbf{x}_k) \simeq 0$ and $\frac{\partial L}{\partial \alpha_k} = -2K(\mathbf{x}_k, \mathbf{x}_k) = -2$. The derivatives show that for the maximization of L the value of α_k should decrease while the values of α_i should stay the same. The constraint $\sum_i \alpha_i = 1$ will then increase the values of α_i a bit, and significantly decrease α_k. On the other hand, when we consider $\alpha_i = \frac{1}{N}, \forall_i$, all derivatives are equal to $\frac{\partial L}{\partial \alpha_i} = -\frac{2}{N}\sum_j K(\mathbf{x}_i, \mathbf{x}_j)$ and all α_i are decreased by the same amount. Because the solution is constrained by $\sum_i \alpha_i = 1$, the α_i cannot be decreased further than $\frac{1}{N}$ and the solution is obtained.

This is equal to (2.10) up to a scaling factor $2/s^2$ and an offset -1 (ignoring higher orders).

Finally, for moderate values of s, the minimization of (2.42) gives a weighted Parzen density as the description of the data (left hand side of the inequality in (2.43)). In the maximization of (2.42) the Lagrange multipliers α_i and α_j tend to become 0 for large values of $K(\mathbf{x}_i, \mathbf{x}_j)$. These large $K(\mathbf{x}_i, \mathbf{x}_j)$ are obtained when objects $\mathbf{x}_i$ and $\mathbf{x}_j$ are near. Only for the smallest $K(\mathbf{x}_i, \mathbf{x}_j)$ the corresponding α_i and α_j become larger than 0. The corresponding $\mathbf{x}_i$ and $\mathbf{x}_j$ are then the most dissimilar objects found at the boundary of the data set. These are the objects which become support vectors ($\alpha_i > 0$). The dissimilarity is measured with the distance s. Both the number of kernels and their weights are obtained automatically by the quadratic optimization procedure.

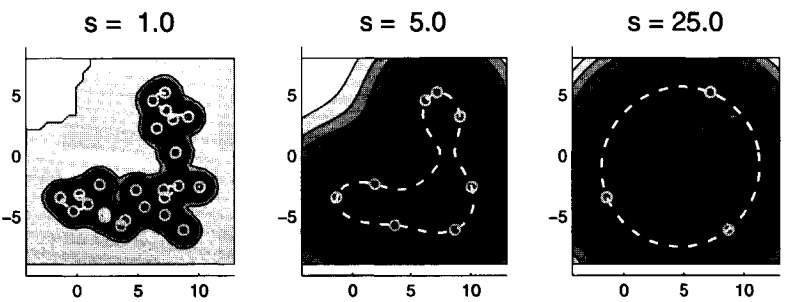


Fig. 2.6: Data description trained on a banana-shaped data set. The kernel is a Gaussian kernel with different width sizes s. Support vectors are indicated by the solid circles; the dashed line is the description boundary.

These three situations, ranging from a Parzen density estimation to the rigid hypersphere, can be observed in figure 2.6. On the left, a small $s = 1.0$ width parameter is used, and on the right a large $s = 25.0$ is used. This distance should be compared with the average nearest neighbor distance in the target objects, which is in this case 1.1. In all cases, except for the limiting case where s becomes huge, the description is tighter than the original spherically shaped description or the description with the polynomial kernels. Note that with increasing s the number of support vectors decreases (this fact will be used in the next section).

Using the Gaussian kernel instead of the polynomial kernel results in tighter descriptions, but it requires more data to support the more flexible boundary. In figure 2.7 the banana-shaped data set is shown with a SVDD with a Gaussian kernel, using $s = 7.0$. The left plot contains just the target objects. It already is tighter than the description in figure 2.2. Using one outlier in the training can tighten the description even further. This is shown in the right plot in figure 2.7.

In this section we showed how the rigid hypersphere model in the SVDD can be made more flexible by introducing kernel functions. We looked at two kernel functions, the polynomial and Gaussian kernel. Although the polynomial kernel function does not result in

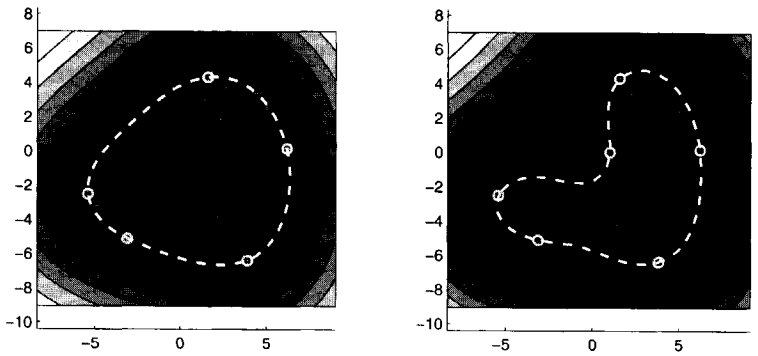


Fig. 2.7: An example of a data description without outliers (left) and with one outlier (right). A Gaussian kernel with $s = 7.0$ is used.

tighter descriptions, the Gaussian kernel does. By changing the value of the free parameter s in the Gaussian kernel, the description transforms from a solid hypersphere, via a mixture of Gaussians to a Parzen density estimator. Because of these favorable features of the Gaussian kernel, we will use this kernel for the SVDD in the rest of the paper (except when indicated otherwise).

2.4 Target error estimate

In section 2.1 we have seen that the SVDD is described by just a few objects, the support vectors. It not only gives a speed-up of the evaluation of a test object $\mathbf{z}$, but surprisingly enough, it is also possible to estimate the generalization of a trained SVDD. When it is assumed that the training set is a representative sample from the true target distribution, then the number of support vectors is an indication of the expected error made on the target set. When the training set is a sample which only captures the area in feature space, but does not follow the true target probability density, it is expected that the error estimate is far off.

This estimate can be derived by applying leave-one-out estimation on the training set [Vapnik, 1998, Bishop, 1995]. For that the notion of *essential support vectors* has to be introduced. The expansion of the center of the hypersphere description $\mathbf{a} = \sum_i \alpha_i \mathbf{x}_i$ is not unique. In some unfortunate cases, more objects are on the boundary of the sphere than is necessary for the description of the sphere (for instance, when 4 objects are on a circle in a 2-dimensional feature space where 3 objects are sufficient). Several expansions are possible where different objects become support vectors. The essential support vectors are these objects which appear in all possible expansions.

To estimate the leave-one-out error, we distinguish 4 cases:

1. When one of the internal points (for which $\alpha_i = 0$) is left out during training and the

data description is computed, the same solution is obtained as with the training set including this training object. During testing this object will therefore be accepted by the description.

2. When an essential support object on the boundary (a support vector with $0 < \alpha_i < C$) is left out, a smaller description is found. This support point will be rejected by the new solution.

3. When a non-essential support vector is left out during training (also an object on the boundary, with $0 < \alpha_i < C$), the original solution is still obtained, using the remaining support vectors in the expansion. During testing, this non-essential support object will still be accepted.

4. The support objects with $\alpha_i = C$, SV^{bnd}, are already outside the sphere. When they are left out of the training set, they will again be rejected during testing.

When we ignore, for the moment, the non-essential support vectors (they are pretty rare), the fraction of objects which become support objects and outliers, is the leave-one-out estimate of the error on the target set. When non-essential support vectors are present, this becomes an over-estimate for the error. If we define:

n_{SV}^{bnd}	the number of support vectors on the boundary, i.e. objects with $0 < \alpha_i < C$
n_{SV}^{out}	the number of support vectors outside the boundary, i.e. objects with $\alpha_i = C$
n_{SV}	the number of support vectors, i.e. objects with $0 < \alpha_i \leq C$
f_{SV}^{bnd}	fraction of the training objects which become support vectors on the boundary, i.e. $\frac{n_{SV}^{\text{bnd}}}{N}$
f_{SV}^{out}	fraction of the training objects which become support vectors outside the boundary, i.e. $\frac{n_{SV}^{\text{out}}}{N}$
f_{SV}	fraction of the training objects which become support vectors, i.e. $\frac{n_{SV}}{N}$

The error estimate then becomes:

$$\bar{\mathcal{E}} \leq f_{SV}^{\text{bnd}} + f_{SV}^{\text{out}} = f_{SV} \tag{2.47}$$

When the support vector data description is trained on a set with both target and outlier objects present, only the fraction of support vectors on the target set should be considered. The fraction on the outlier class can be used for estimating the error on the outlier set, but only when the outliers are drawn independently from the true outlier distribution. Because most often the outlier class is sampled very sparsely, a true representative data set is not available and no confident estimate can be obtained.

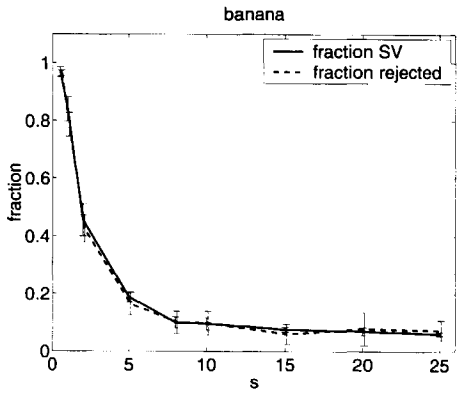


Fig. 2.8: Fraction of support vectors f_{SV} and fraction of target data rejected f_{T-} versus the width parameter of the Gaussian kernel. Fifty objects are drawn from a 2-dimensional banana-shaped distribution. The description is tested on 250 test objects. The experiment is repeated 10 times. No errors are allowed ($C = 1$).

This target error estimate opens the possibility to optimize the regularization parameter C and the width of the kernel s in the Gaussian kernel.[5] The value of the regularization parameter C can be chosen by using the fact that for objects outside the hypersphere $\alpha_i \equiv C$. Recall the constraint that $\sum_i \alpha_i = 1$. When n_{SV}^{out} objects are outside the sphere, we still have to satisfy $n_{SV}^{\text{out}} C \leq 1$. This means for the value of C that:

$$C \leq \frac{1}{N f_{SV}^{\text{out}}} \tag{2.48}$$

When a training data set without outliers is expected, C can be set to 1.0 (or larger), indicating that all training data should be accepted.

In section 2.2 (page 24) we treated the SVDD with negative examples. When outlier objects are available during training, both C_1 and C_2 have to be specified (see (2.15)). C_1 can directly be found by using the estimate of formula (2.48). C_2 can now be derived by using an estimate of the error of the second kind, thus directly applying (2.48) on the outlier data. It is also possible to use a priori defined costs for false positives and false negatives and to apply (2.23).

Secondly, when the value of C (or C_1 and C_2) is determined, the kernel width s can be optimized on the basis of the required target acceptance rate. Assume that a minimum for Lagrangian (2.42) is found for a certain s_1. For a second width value $s_2 > s_1$:

$$K(\mathbf{x}_i, \mathbf{x}_j; s_2) \geq K(\mathbf{x}_i, \mathbf{x}_j; s_1) \qquad \forall_{i,j}, \quad s_2 > s_1 \tag{2.49}$$

[5] Or the degree n in case of the polynomial kernel. We will disregard the optimization of the degree n here because we will only use the Gaussian kernel for the SVDD in the rest of the thesis. The optimization for the Gaussian kernel deserves therefore more attention.

For large enough s_2, (2.42) is maximized when the α_i and α_j corresponding to the larger terms of $K(\mathbf{x}_i, \mathbf{x}_j)$ become 0. The constraint $\sum_i \alpha_i = 1$ prevents all α_i from becoming 0, just a few α_i stay large. This means that the most isolated objects become support vectors and that the number of support vectors tends to decrease with increasing s (until we have obtained the solid hypersphere solution, see page 33). By the use of formula (2.47) this also means a decrease of the expected error on the target set. How many and which objects become support vectors depends on the relative sizes of $K(\mathbf{x}_i, \mathbf{x}_j)$, and thus on the target data distribution.

The s can now be optimized to a prespecified target error. The minimization of Lagrangian (2.42) requires s to be specified beforehand, so s cannot be optimized directly. An iterative scheme is therefore used. The scheme starts by computing a data description for small s, in the order of the minimum nearest neighbor distances in the training set:

$$s_{start} = \min_i \|\mathbf{x}_i - \mathrm{NN}^{tr}(\mathbf{x}_i)\| \tag{2.50}$$

where $\mathrm{NN}^{tr}(\mathbf{x}_i)$ is the nearest neighbor object of $\mathbf{x}_i$ in the training set. When the fraction of support vectors f_{SV} is larger than the predefined target error rate, s is increased and a new data description is trained. This is repeated until the f_{SV} is smaller than the desired fraction. s can be increased up to the maximum inter-object distance present in the training set. For distances larger than the maximum distance in the dataset the rigid hypersphere solution is obtained:

$$s_{end} = \max_{i,j} \|\mathbf{x}_i - \mathbf{x}_j\| \tag{2.51}$$

In figure 2.8 the f_{SV} and the f_{T-} are shown for several values of the width parameter s (using $C = 1$). For this dataset $s_{start} = 0.62$ and $s_{end} = 12.7$. A SVDD has been trained 10 times on a sample of 50 objects drawn from a 2-dimensional banana-shaped distribution (this is the same distribution which has been used in figure 2.2). The performance is checked using an independent test sample of 250 objects from the same target distribution. For small s all objects become support vectors and all test objects are rejected. For large s on average 3 objects become support vectors (6%) and also about 6% of the test set is rejected. The error on the test set closely follows f_{SV}, as argued by (2.47).

The difference between optimizing s and C in a SVDD with the Gaussian kernel is shown in figure 2.9. In the left subplot s is optimized to obtain $f_{SV}^{\mathrm{bnd}} = 0.5$ (we keep $C = 1$). About half of the objects are now on the boundary, but a very noisy solution is found. Because in this dataset far less than 50% of the data is on the 'true' data boundary, the SVDD is severely overtrained or overfitted. In most datasets, $f_{SV}^{\mathrm{bnd}} = 0.5$ is already too much. In the right subplot of figure 2.9 the regularization parameter C is optimized to $f_{SV}^{\mathrm{out}} = 0.5$. The width parameter s is now optimized to find $f_{SV}^{\mathrm{bnd}} = 0.1$. The solution is far smoother than in the left plot. This smoothness is largely enforced by the fraction of support vectors on the boundary, and therefore by the width parameter s. Thus, when large fractions of the target set should be rejected, i.e. more than what is expected to lie on the boundary of the dataset, the C should be adapted. In practice, when a limited number of objects is rejected, adjusting s is sufficient.

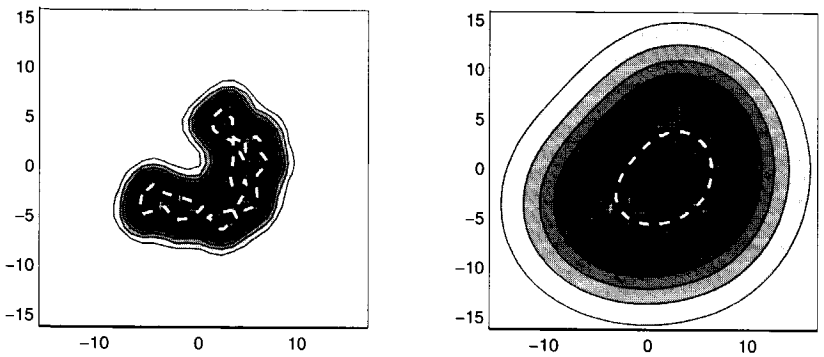


Fig. 2.9: The difference between the SVDD after optimizing over s (left) and optimizing over C (right). In both cases a 50% target rejection rate is optimized.

A general constraint for the SVDD is that enough data are available. In the previous example with 50 objects in a 2-dimensional data set, it appears that the minimum number of support vectors is 2 to 3. When a target error lower than 6% is requested, more data is required. The accuracy of the target error estimate, the required number of training objects and some other characteristics will be investigated in section 2.6.

In this section we showed that, when the SVDD is trained with a representative dataset, an estimate of the error on the target set can be obtained by looking at the the number of support vectors. Using this estimate, we can determine values for the free parameters in the SVDD (s and C). With this, the complete SVDD is specified. In the next section we will compare the SVDD with a resembling one-class classifier from literature, the ν-SVC.

2.5 The ν-SVC

This chapter started with the assumption of a hypersphere around the data as the basis for the SVDD. The hypersphere is chosen because it gives a closed boundary around the data. Schölkopf proposed another approach [Schölkopf et al., 1999], which is called the ν-support vector classifier. Here a hyperplane is placed such that it separates the dataset from the origin with maximal margin (the parameter ν will be explained in the coming sections). Although this is not a closed boundary around the data, it gives identical solutions when the data is preprocessed to have unit norm. In this section we will show how the SVDD applied to data with unit norm, is equivalent to the hyperplane approach.

For a hyperplane $\mathbf{w}$ which separates the data $\mathbf{x}_i$ from the origin with maximal margin ρ, the following holds:

$$\mathbf{w} \cdot \mathbf{x}_i \geq \rho - \xi_i, \quad \xi_i \geq 0, \qquad \forall_i \tag{2.52}$$

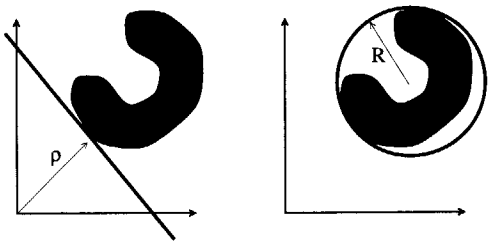


Fig. 2.10: Data descriptions by the ν-SVC (left) and the SVDD (right) without the introduction of kernels $K(\mathbf{x}_i, \mathbf{x}_j)$.

and the function to evaluate a new test object $\mathbf{z}$ becomes:

$$f_{\nu-SVC}(\mathbf{z}; \mathbf{w}, \rho) = I\left(\mathbf{w} \cdot \mathbf{z} \leq \rho\right) \tag{2.53}$$

Schölkopf now minimizes the structural error $\mathcal{E}_{\text{struct}}$ of the hyperplane, measured by $\|\mathbf{w}\|$ and some errors, encoded by the slack variables ξ_i, are allowed. This results in the following minimization problem:

$$\min \left(\frac{1}{2}\|\mathbf{w}\|^2 - \rho + \frac{1}{\nu N} \sum_i \xi_i \right) \tag{2.54}$$

with the constraints given by formulae (2.52). The regularization parameter $\nu \in (0, 1)$ is a user defined parameter indicating the fraction of the data that should be accepted by the description. It can be compared with the parameter C in the SVDD, formula (2.48) with comparable bounds. This method is therefore called the ν-support vector classifier, or ν-SVC.

An equivalent formulation of (2.52) and (2.54) is

$$\max \left(\rho - \frac{1}{\nu N} \sum_i \xi_i \right), \quad \text{subject to} \quad \mathbf{w} \cdot \mathbf{x}_i \geq \rho - \xi_i, \quad \xi_i \geq 0 \forall_i, \quad \|\mathbf{w}\| = 1 \tag{2.55}$$

An extra constraint on $\|\mathbf{w}\|$ is introduced, and the optimization is now over ρ instead of $\|\mathbf{w}\|$. Again it is possible to introduce kernel functions for the inner products. Using the appropriate kernels, especially the Gaussian kernel, data descriptions comparable with the SVDD are again obtained.

When all data is normalized to unit norm vectors, the equivalence between the SVDD and the ν-SVC can be shown. To normalize the data without the loss of information, an extra bias term of 1 is introduced and the extended vector is normalized:

$$\mathbf{x}' = \frac{(\mathbf{x}, 1)}{\|(\mathbf{x}, 1)\|} \tag{2.56}$$

We start again with a hypersphere around data (see also (2.3) and (2.4)):

$$\min \left(R^2 + C \sum_i \xi_i \right) \qquad \text{subject to} \quad \|\mathbf{x}_i - \mathbf{a}\|^2 \leq R^2 + \xi_i, \quad \forall_i \tag{2.57}$$

We transform all our data by substituting definition (2.56) into the constraints (2.57). $\mathbf{x}'$

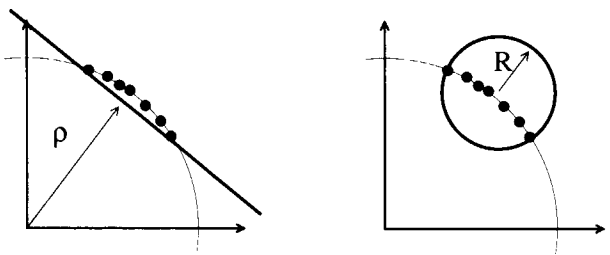


Fig. 2.11: Data descriptions by the ν-SVC and the SVDD where the data is normalized to unit norm.

and $\mathbf{a}'$ become normalized vectors:

$$\min \left(R'^2 + C' \sum_i \xi_i' \right) \qquad \text{subject to} \quad \|\mathbf{x}_i' - \mathbf{a}'\|^2 \leq R'^2 + \xi_i', \quad \forall_i \tag{2.58}$$

The constraints are rewritten as:

$$\begin{aligned} \|\mathbf{x}_i'\|^2 - 2\mathbf{a}' \cdot \mathbf{x}_i' + \|\mathbf{a}'\|^2 &\leq R'^2 + \xi_i' \\ 1 - 2\mathbf{a}' \cdot \mathbf{x}_i' + 1 &\leq R'^2 + \xi_i' \\ \mathbf{a}' \cdot \mathbf{x}_i' &\geq \frac{1}{2}(2 - R'^2 - \xi_i') \end{aligned} \tag{2.59}$$

When the signs are changed in (2.57) and an extra constant 2 is introduced, we get:

$$\max \left(2 - R'^2 - C' \sum_i \xi_i' \right) \qquad \text{subject to} \quad \mathbf{a}' \cdot \mathbf{x}_i' \geq \frac{1}{2}(2 - R'^2 - \xi_i'), \quad \forall_i \tag{2.60}$$

Now we redefine:

$$\mathbf{w} = \mathbf{a}', \quad \rho = \frac{1}{2}(2 - R'^2), \quad \frac{1}{\nu N} = C', \quad \xi_i = \frac{1}{2}\xi_i', \quad \forall_i \tag{2.61}$$

and the following optimization problem is obtained:

$$\max \; 2 \left(\rho - \frac{1}{\nu N} \sum_i \xi_i \right) \quad \text{with} \quad \mathbf{w} \cdot \mathbf{x}_i \geq \rho - \xi_i, \quad \forall_i \tag{2.62}$$

which is equal to (2.55) up to a factor 2 in the error function.

In the case of a Gaussian kernel, the data is implicitly rescaled to norm 1 (section 2.3). Therefore, the solutions of the SVDD and the ν-SVC are identical when the Gaussian kernel width equals s and $C = \frac{1}{\nu N}$ is used. In their practical implementation the ν-SVC and SVDD operate comparably. Both perform best when the Gaussian kernel is used. In the SVDD, the width parameter s is optimized to find a prespecified fraction of support vectors. The parameter C is set at a prespecified value indicating the fraction of objects which should be rejected. In the ν-SVC the ν directly gives the fraction of objects which is rejected. Although it is not specified explicitly in [Schölkopf, 1997], the s in the ν-SVC can be optimized in the same way as in the SVDD.

2.6 Data description characteristics

In the previous sections we have seen some (theoretical) properties of the SVDD; the influence of the parameters s and C. The next question is how it behaves in practical situations. How well does the SVDD capture the target set, how many support vectors does it need, how well is the target error estimated and how well can outliers be rejected? In this section these and other characteristics of the SVDD will be investigated.

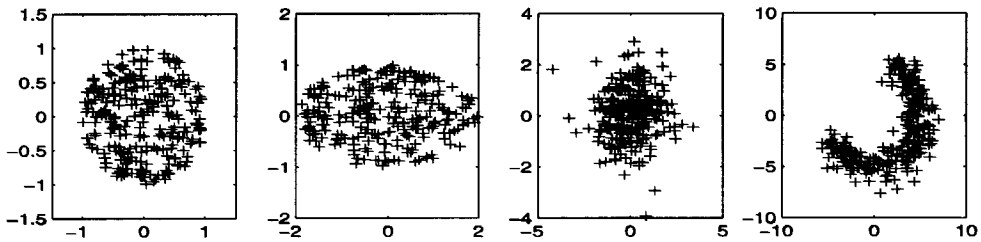


Fig. 2.12: Two dimensional versions of 4 data sets, uniform spherical, elliptical, Gaussian and banana-shaped.

We consider 4 artificial data sets, shown from left to right in figure 2.12. The first dataset is a spherical data set `sphere` where objects are uniformly distributed in a unit hypersphere. The second set is an ellipse dataset `ellipse` in which the data is drawn from the same uniform distribution, but in which one of the feature values has been multiplied by 2. The third data set is a Gaussian distribution with unit standard deviation, `Gauss`. The last data set is a 2-dimensional banana-shaped set, `banana`. Higher dimensional versions are created by drawing new samples from the same distribution and by using the new feature values as the extra features.

For measurements on real world data, conventional classification problems are taken from the UCI repository [Blake et al., 1998]. We will use the `Iris`, `sonar`, `glass` and `imox` datasets. A data description is trained on each of the classes separately. Thus for the three-class `Iris` dataset three different one-class classification problems are obtained.

2.6.1 Rigid hypersphere

First, how much data are required to find a good description of a data set in some feature space? This question cannot be answered beforehand. It not only depends on the shape of the target data set, but also on the distribution of the (unknown) outlier set. We will assume uniformly distributed outlier data, and when a rigid, spherically-shaped data description is considered, a lower limit for the number of support vectors, n_{SV}, can be obtained. For a strictly spherically shaped model, the coordinates of the center of the sphere and the radius of the sphere are required (in total, the values of $N+1$ free parameters have to be determined). In theory 2 objects are sufficient to determine the free parameters for the sphere (2 objects determine $2N$ parameters). However, the center of the sphere is described as a linear combination of the objects $\mathbf{x}_i$ (equation (2.7)) with the constraint that $\sum_i \alpha_i = 1$. The center can then only be described by 2 objects when the 2 objects are the most remote and diametrically placed with respect to the center of the sphere. Due to the construction of the SVDD, the objects should then be on the boundary of the description. Therefore, the required number of objects is, in general, $d+1$, for d-dimensional data with non-zero variance in all directions. For data in a subspace, the number becomes less (down to 2 if the data is placed on a 1-dimensional line).

Ten different samples are drawn for each of the data sets for several dimensionalities, each sample containing 250 examples. On each of the data sets a SVDD with polynomial kernel, $n = 1$, is trained (this means that a rigid hypersphere is used). In the left plot of

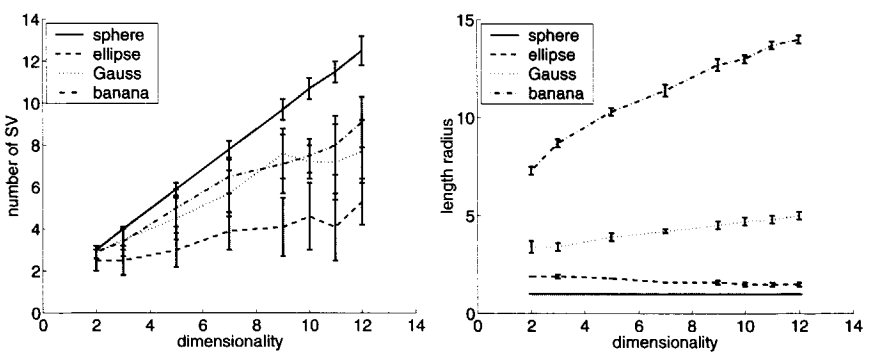


Fig. 2.13: The left plot shows the number of support vectors, for several artificial data sets using polynomial kernel with degree 1, while the right plot the length of the radius of the hypersphere. All datasets contain 250 training objects.

figure 2.13 we investigate the number of support vectors n_{SV} for different dimensionalities and different (artificial) data sets. The first observation which can be made from the left plot in figure 2.13 is that for the `sphere` data set the $n_{SV} = d + 1$ where d is the dimensionality. For higher dimensionalities the variance of the number increases and the average decreases somewhat. When one of the feature space axes is stretched, n_{SV} drops. In particular, in the `ellipse` dataset the higher dimensionalities allow for more economical

use of objects; often $d/2$ objects are enough. n_{SV} never drops to the minimum of 2, though. In the `Gauss` and the `banana` datasets, n_{SV} is between the worst case `sphere` dataset and the most efficient `ellipse` dataset.

With increasing dimensionality not only the n_{SV} increases, but also the variance of n_{SV} over different samples. These experiments show that the SVDD cannot completely escape the curse of dimensionality. Using a larger number of features requires more training objects to ensure the same target class error.

In the right plot of figure 2.13, the radius of the rigid hypersphere is shown. We see that for the `sphere` data the radius of the SVDD remains in all cases 1.0, even for higher dimensionalities, indicating that a tight description is obtained. For lower dimensionalities the radius of the SVDD on the `ellipse` data is almost 2.0, but for higher dimensionalities it decreases somewhat. In the `ellipse` dataset, the description is ultimately dependent on the two extreme tips in their distribution. It appears that these tips of the ellipse are not filled completely with samples and therefore the description can become smaller. In the spherical distribution the description cannot become smaller, because no tips are present, and all directions in the data are important. For the `Gauss` and especially for the `banana` data set, the radii increase significantly with the growing dimensionality. This is mainly caused by a few objects with large feature values, which results in large distances in higher dimensional feature spaces. This effect is largely induced by the fact that no target objects are allowed outside the description ($C = 1$). Rejecting a few of these outliers (by adjusting C) will tighten the description again and will reduce the influence of these outliers on the final data description.

Table 2.1: Minimum number of support vectors for the SVDD with a polynomial kernel with $n = 1$ (i.e. the rigid hypersphere solution) applied to different datasets and different target classes. The experiments on the artificial datasets are averaged 25 times, while for the real world datasets 5-fold cross-validation is applied. The dimensionality of the data is reduced (when possible) by PCA.

dataset	class	orig	2-D	5-D	10-D	25-D	dataset	class	orig	2-D	5-D
`Gauss`			2.75	4.94	6.78	11.28	`glass`	1	3.90	2.80	3.70
`banana`			2.87	4.95	8.67	13.65	(9D)	2	2.90	2.70	2.20
`ellipse`			2.03	2.42	2.69	3.23		3	2.20	2.20	2.20
`Iris`	1	2.10	2.10					4	4.00	3.00	4.00
(4D)	2	3.10	2.00				`imox`	1	4.10	3.00	4.10
	3	3.00	2.20				(8D)	2	2.30	2.10	2.20
`sonar`	1	5.90	2.10	2.20	5.40	5.70		3	3.80	2.10	3.00
(60D)	2	6.70	3.00	4.10	4.40	6.70		4	4.70	3.00	3.40

The number of support vectors, n_{SV}, is shown in table 2.1 for the other datasets and varying dimensionalities. Still the rigid hypersphere solution is used (a polynomial kernel with degree $n = 1$). The `Gauss` and `banana` sets have non-zero variance in all feature

directions, the **ellipse** mainly in the first direction. n_{SV} in the original feature space is shown in the third column (except for the artificial datasets). In the next columns the results are shown for datasets which are reduced to the first few principal components. For the **Gauss** and **banana** datasets n_{SV} increases with increasing dimensionality, but remains in the order of $d/2$. In the **ellipse** dataset the single large variance direction causes n_{SV} to stay very low. For the other data sets n_{SV} is bounded by the subspace in which the data is distributed. For the **sonar** database n_{SV} hardly changes above 25 dimensions, for the other datasets this already happens with 5-dimensional data. Finally, note that n_{SV} for different classes can differ, e.g. in the **imox** dataset it ranges from 2 to almost 5. When a limited amount of data is available, this minimum number of support vectors immediately gives an upper bound on the error (by formula (2.47)).

Note that in these experiments a polynomial kernel with degree 1 is used (definition (2.33)). When more flexible kernels are used (for instance, the Gaussian kernel, definition (2.41)) more support vectors are required and the SVDD suffers more from high dimensional feature spaces.

2.6.2 Flexible descriptions

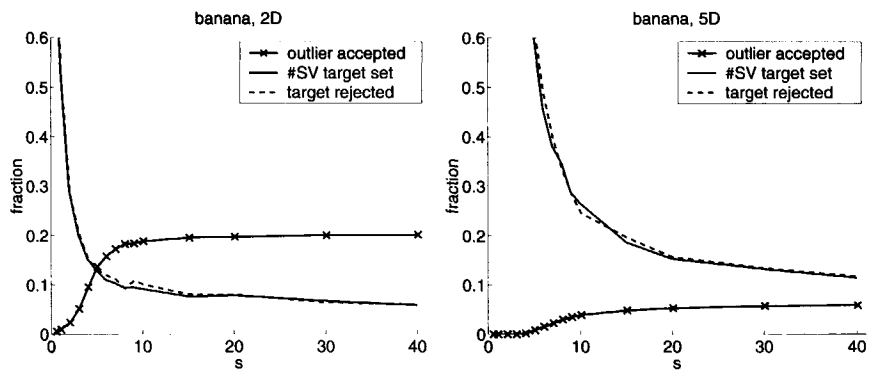


Fig. 2.14: The fraction of outliers which is accepted f_{O+}, the fraction of the target data which becomes support vectors f_{SV} and the fraction of target data rejected f_{T-} versus the width parameter s for 2-dimensional (left) and 5-dimensional (right) data. The training set size is 50.

From now on, we will use the SVDD with the Gaussian kernel (definition (2.41)). In figure 2.14 the outlier acceptance rates on the **banana** distribution are investigated. The fraction of outliers which is accepted f_{O+}, the fraction of training objects which become support vectors f_{SV} and the error on the target set f_{T-} is shown for data with two different dimensionalities (2- and 5-dimensional). For varying width parameter s a SVDD is trained in the same way as in figure 2.8. f_{T-} is then estimated by drawing a new independent test set, containing 200 objects. 1000 testing outliers are drawn from a square block

around the data. Note that by changing the dimensionality of the dataset, the one-class problem changes considerably. Most importantly, the difference between the area covered by the target and outlier data in feature space increases (we will have a more thorough discussion about this subject in section 3.1.1). Because distances between objects increase with increasing dimensionality, the error on the outlier set will decrease.

By increasing s, n_{SV} decreases, closely followed by f_{T-}. For large s, the error is bounded by the minimum number of support vectors which is needed for the description. In this case about 3 support vectors for 2-dimensional data and about 5 for 5-dimensional data are required, which results in a minimum error of about 6% or 10%. The maximal fraction of outliers accepted is the ratio of the volume of the hypersphere to the volume of the outlier data block. This fraction is bigger in 2- than in 5-dimensional data. Also the distances between objects are significantly smaller in 2 dimensions than in 5. For $s > 20$ in the 2-dimensional data and for $s > 40$ in the 5-dimensional data both fractions remain constant, indicating that the maximum scale in the data is reached. In this situation effectively the rigid hypersphere solution is obtained.

2.6.3 Training with outliers

Table 2.2: Classification error of conventional classifiers and one-class classifiers, trained on each class separately. Testing is done on an independent test set. Numbers in bold indicate the best performance for the conventional classifiers and the one-class classifiers.

dataset	class	Bayes	Parzen	SVC-p3	SVDD	SVDD-p3	SVDD-neg	SVDD-n-p3
Iris	1	**0.000**	**0.000**	**0.000**	0.047	0.033	**0.013**	0.020
	2	0.267	**0.033**	0.080	0.087	**0.067**	**0.067**	0.073
	3	0.073	**0.033**	0.060	**0.067**	0.340	0.107	0.093
glass	1	0.219	**0.187**	0.220	**0.271**	0.364	0.327	0.299
	2	0.326	0.206	**0.196**	0.402	0.617	**0.285**	0.416
	3	**0.084**	0.089	0.117	0.286	0.369	**0.117**	0.089
	4	0.071	**0.052**	0.070	0.150	0.748	**0.145**	0.752
sonar	1	0.250	0.145	**0.115**	**0.403**	0.515	0.428	0.519
	2	0.250	0.145	**0.115**	**0.307**	0.447	0.312	0.442
imox	1	0.088	**0.041**	0.046	0.129	0.656	**0.108**	0.561
	2	0.047	**0.005**	0.016	**0.068**	0.672	0.094	0.677
	3	0.063	**0.005**	0.062	0.088	**0.083**	0.098	0.123
	4	0.114	**0.041**	0.104	0.167	0.459	**0.145**	0.392

To compare data descriptions with conventional two-class classifiers, both are trained on the classification problems mentioned in the previous section. One class is the target class, and all other data are considered as outlier data. 10-fold cross validation is used to find the

classification error. To investigate the influence of the norms of the vectors in real world applications, not only the Gaussian kernel, but also the polynomial kernel is considered. Three conventional classifiers are used: a linear classifier based on normal densities (called Bayes), a Parzen classifier and the support vector classifier with polynomial kernel of degree 3 (abbreviated by SVC-p3).

Furthermore, we consider four versions of the SVDD. The first two versions do not use example outlier data in their training. The first data description is a SVDD with polynomial kernel and degree $n = 3$, SVDD-p3. The second method is a SVDD with a Gaussian kernel, where s is optimized for a target rejection rate of 10%. The last two methods use negative examples in their training, and again the polynomial and Gaussian kernel is used, SVDD-neg and SVDD-n-p3. When the polynomial kernel was used, the data is rescaled to unit variance.

The results are shown in table 2.2. Numbers in bold indicate the best performance for the conventional classifiers and the one-class classifiers. In most cases the classifiers, especially the Parzen classifier and the SVC-p3, outperform the data descriptions. Also, the data descriptions using outliers perform somewhat better than without example outliers, but in some cases it requires careful optimization of the C_1 and C_2 parameters (for instance, for the `imox` dataset which has much overlap between the classes). In most cases the data descriptions with the polynomial kernel (SVDD-p3 and SVDD-n-p3) perform much worse than with the Gaussian kernel, except for the `Iris` dataset where the performances are comparable.

Note that here one-class classifiers are compared with conventional multi-class classifiers. Although the former appear to perform somewhat worse, they obtain a closed boundary around the data in all cases. The latter classifiers are optimized for distinguishing between the target and outlier data in the training set, but in the case of atypical target data, or very sparsely sampled outlier data, it cannot be expected that the conventional classifiers will work well. This experiment just shows that using the more restricted one-class classifiers, the performance is not much worse in comparison with the conventional multi-class classifiers.

2.6.4 Outliers in the target set

Finally, we investigate how well outliers in the target set can be rejected. Artificial outliers are added to a target set, and a SVDD with Gaussian kernel is trained on this dataset. Because the set of outlier objects is known, the performance on this artificial outliers can be computed. The outliers are drawn from a rectangular uniform distribution around the target set. The edges of the box are 6 times the standard deviation of the training set. Objects within 3 times the standard deviation from the center of the target set will not be used as outliers. The training set contains 100 objects, the target objects are drawn from a normal distribution. An example of this target and outlier distribution is shown in figure 2.15. Parameter s will be optimized to obtain $f_{SV} = 0.15$ (which includes both target objects and outlier objects). The fraction outliers is varied from 0 to 30%.

In section 2.4 we showed that outliers in the training set can be rejected by adjusting

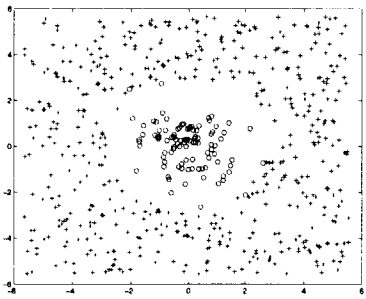


Fig. 2.15: The scatter plot of 500 artificial outlier objects (indicated by '+') created around a Gaussian distributed target set (indicated by 'o').

the regularization parameter $C \leq \frac{1}{n_{SV}^{out}} = \frac{1}{N \cdot f_{SV}^{out}}$. In order to make a direct connection with the fraction outliers in the dataset, we consider the parameter:

$$\nu = \frac{1}{NC} \tag{2.63}$$

This is the direct analog of the ν parameter in the ν-SVC as introduced in section 2.5. Note that this parameter is set by the user. The fraction of objects of the target class which is rejected by the SVDD, f_{T-} (and f_{O-} if outliers are present in the training set), is measured after the SVDD is fitted.

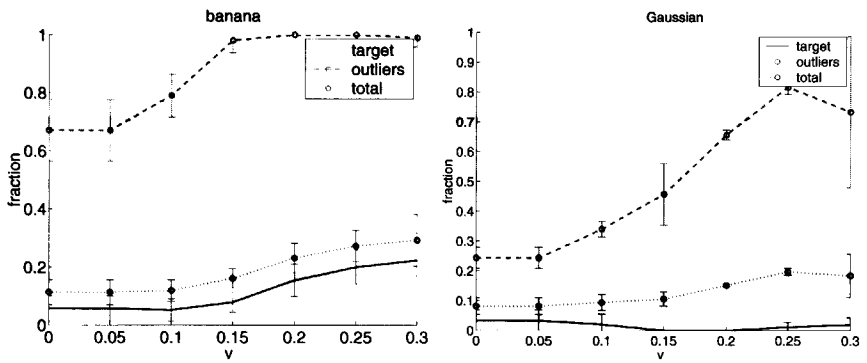


Fig. 2.16: f_{T-} and f_{O-} for banana target set (left) and the Gaussian target set (right). The results are averaged over 10 runs.

In figure 2.16 the f_{T-} and f_{O-} are shown for different values of ν. In the left plot the `banana` target distribution is used, where 10% of the objects are outliers. In the right plot the `Gauss` shaped data set is used, with 30% outlier objects. The results are averaged over 10 runs. When from the 100 training objects 10% are outliers, ν should be bounded

$\nu > 0.1$ (such that for $C < 0.1$). When $\nu < 0.1$ some outliers have to be accepted in the description. For $\nu = 0.05$ just 75% of the outliers are rejected (which is about 5% of the training data). Because the outliers, in general, do not completely surround the target data, some target objects also become support vectors. Most outliers are rejected for $\nu > 0.1$, but sometimes outliers are near the target data and are still accepted. Also 'true' target objects are now rejected. This is the classic tradeoff between false positive and false negative.

In summary, in this section we investigated how the SVDD operates on some artificial and some real world data. The experiments show that it is possible to train well-fitting data descriptions when a sufficient number of examples is available. The higher the dimensionality of the data and the more complex the distribution, the more training data is required. Furthermore, example outlier objects can be used to improve the classification performance. When a large number of outlier examples is available, conventional two-class classifiers outperform the SVDD. In the next chapter we apply the SVDD to a real one-class classification problem: the problem of machine diagnostics.

2.7 Machine diagnostics experiments

In the next chapter a more extensive comparison between different one-class classification methods will be given. The methods and the error measure will be discussed in more detail. In this section we mainly focus on the performance of the SVDD in real life problems. The SVDD will be applied to a machine diagnostics problem: the characterization of a submersible water pump [Ypma and Pajunen, 1999, Ypma et al., 1999, Tax et al., 1999].

The task is to detect when the pump deviates from its normal operation conditions. Both target objects (measurements on a normal operating pump) and negative examples (measurements on a damaged pump) are available.

To characterize the condition of a pump, vibration sensors are mounted on the machine. From the recorded time series subsamples are taken and the power spectrum of these signals is calculated. These high dimensional spectrum features are reduced in dimensionality by taking the first few principal components of the data. In a test bed several normal and outlier situations can be simulated. Several normal situations consist of situations with different loads and speeds of the pump. The outlier data contain pumping situations with a loose foundation, an imbalance and a bearing failure. The special feature of this setup is that, beforehand, a set of operating modes has to be considered. When one-class classification methods are compared, we can not only investigate the case in which the training data is sampled according to the test data (which should represent the 'true' distribution), but it is also possible to use just a few load and speed settings for the training set and see how the behavior is on the test data.

To see how well the SVDD and the SVDD with negative examples (SVDD-neg) perform, we compare them with three other, simple one-class classification methods. The first two methods estimate the probability density of the target class. The first method assumes a normal density where the mean and the covariance matrix Σ of the target data have to be

estimated. In high-dimensional feature spaces the covariance matrix of the target set can become singular. The covariance matrix is then regularized by adding a small term λ to the diagonal terms in the covariance matrix, i.e.: $\Sigma' = \Sigma + \lambda I$ [Raudys and Jain, 1991]. The regularization parameter is optimized for the problem at hand and is in most cases between 10^{-3} and 10^{-6}. The second method is the Parzen density where the width of the Parzen kernel is estimated using leave-one-out optimization [Duin, 1976].

The last method is based on nearest neighbor distances, and will be called the nearest neighbor method, NN-d. It compares the local density of an object $\mathbf{z}$ with the density of the nearest neighbor in the target set [Tax and Duin, 2000]. The distance between the test object $\mathbf{z}$ and its nearest neighbor in the training set $\mathrm{NN}^{tr}(\mathbf{z})$ is compared with the distance between this nearest neighbor $\mathrm{NN}^{tr}(\mathbf{z})$ and its nearest neighbor in the training set $\mathrm{NN}^{tr}(\mathrm{NN}^{tr}(\mathbf{z}))$. When the first distance is much larger than the second distance, the object will be regarded as an outlier. We use the quotient $\rho_{\mathrm{NN}^{tr}}$ between the first and the second distance as indication of the resemblance of the object to the training set:

$$\rho_{\mathrm{NN}^{tr}}(\mathbf{z}) = \frac{\|\mathbf{z} - \mathrm{NN}^{tr}(\mathbf{z})\|}{\|\mathrm{NN}^{tr}(\mathbf{z}) - \mathrm{NN}^{tr}(\mathrm{NN}^{tr}(\mathbf{z}))\|} \tag{2.64}$$

For more details, see [Tax and Duin, 1998], the next chapter, page 69 or appendix B.

To compare different methods, the tradeoff between f_{T-} versus f_{O+} is investigated. For all methods we try to set the thresholds such that a fraction of 0.01, 0.05, 0.1, 0.2, 0.25 and 0.5 of the target class is rejected. The f_{O-} is measured afterwards. In the next chapter we will discuss a scalar error measure based on these target acceptance and outlier rejection rates.

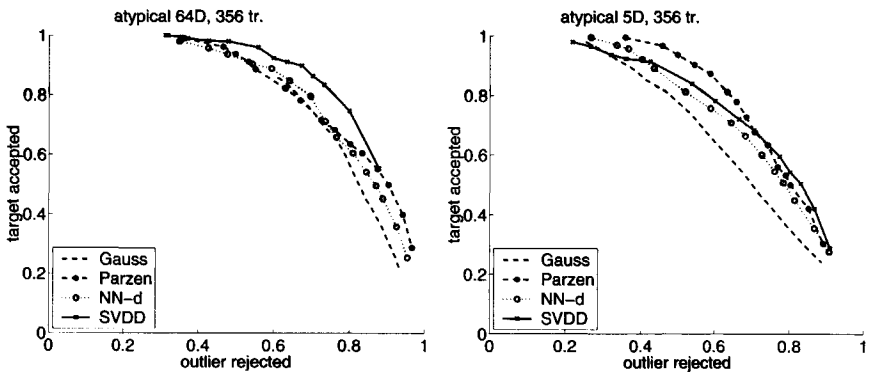


Fig. 2.17: ROC curve for the 4 one-class methods trained on all available operation conditions (vibration dataset 1). The left plot shows the results in 64 dimensions, the right plot in 5 reduced dimensions. The measurements from 5 sensors are used.

The plots in figure 2.17 show the results in the form of the Receiver-Operating Characteristic (ROC) curves [Metz, 1978], where f_{T+} is plotted versus f_{O-} for the power spectrum

features (a more elaborate treatment of ROC curves will be given in section 3.1.2, page 60). From all pump operation conditions, vibration signals are measured and used in training and testing. Measurements on 5 different vibration sensors are used. This resulted in a training set of 356 target objects, represented in a 64-dimensional feature space. In the left plot we have the results in the complete feature space, on the right we have used just the first 5 principal components of the training set.

In these datasets the training set is identically distributed as the testing data. All one-class methods perform about the same on the 64-dimensional dataset. The good performance by the normal density model indicates that the target class is a nice unimodal cluster. In the reduced 5-dimensional case the estimation of the normal density is still possible, as shown in the right plot of figure 2.17, but performance is worse. All methods show about equal performance for low target acceptance rates f_{T+}. But for high f_{T+}, the SVDD starts suffering from the fact that not enough data are available to find a description more flexible than a sphere. The Parzen density describes the target data very well, which is not very surprising, given the fact that we have 356 objects in a 5-dimensional feature space. All methods have problems in distinguishing between target and outlier objects, though. Even when 50% of the target objects are rejected, only 80% of the outlier objects can be rejected. When it is not allowed to reject more than 5% of the target data, we have to accept about 50 to 60% of the outlier data.

From these experiments we can conclude that it is possible to distinguish between target and outlier objects, but that there is some overlap between the two classes. The target and outlier objects cannot be distinguished without errors. To improve the performance, we will use domain knowledge about the placing of the vibration sensors. In this experiment, all measurements of all sensors are used, but next we will show that some sensors are placed in a position where they are not very sensitive to the failures that might occur. Because their information content might be not significant, their noise contribution might be large enough to decrease the representational power of the dataset. By removing the measurements of these sensors, the class overlap might thus decrease.

In figure 2.18 the results of all one-class methods are shown for the data from 3 of the 5 sensors (thus removing the most noisy measurements). Again, all available pump operation conditions are used. In the left plot the original 64-dimensional data is used, in the right plot only the first 3 principal components of the data. It appears that by using the original 64-dimensional representation of this data, the outliers can be distinguished far better than in the previous case where the measurements of all 5 vibration sensors were used. It is now possible to accept all target objects, while more than 80% of the outlier objects is rejected. This is a significant improvement, since in the previous case less than 40% of the outliers objects is rejected. When the first 5 principal components are used, performance again decreases to about the same level as in the previous case. It appears that here about 30 features are necessary to obtain a good separation between target and outlier objects.

In the 64-dimensional dataset the best performance is obtained using the NN-d. This indicates that the sample size might be a bit too small. It appears that the NN-d is especially apt in operating in low sample size situations; see appendix B. The fact that

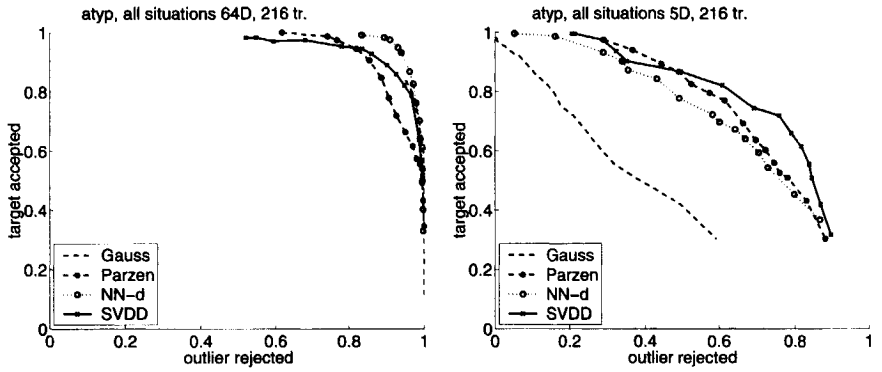


Fig. 2.18: ROC curve for the 4 one-class methods trained on all available operation conditions (vibration dataset 1). The left plot shows the results in 64 dimensions, the right plot in 5 dimensions. The measurements from 3 sensors are used.

we are dealing with a small sample size situation, is also visible in the performance of the Parzen density estimator, which is incapable of correctly classifying more than 80% of the data. The poor performance of the Gaussian model on the data in 5 dimensions, shows that now the target class is less normally distributed than in 64 dimensions. The SVDD performs well, but again suffers for high values of f_{T+}. In these datasets, where the training data is distributed identically to the testing set, the SVDD does not improve much over the normal density estimators (especially the Parzen density estimators). This will change when we consider the case in which we train with an atypical training sample.

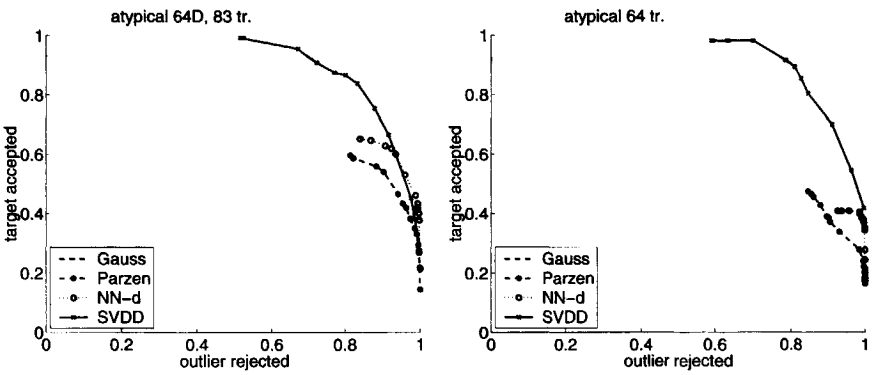


Fig. 2.19: ROC curve for the 4 one-class methods trained on subsets of the operation conditions. The left plot shows the results of 6 situations in 64 dimensions (vibration dataset 2), the right plot just 4 situations are used (vibration dataset 3).

In the setup used in the experiments two parameters determined the operation mode

of the pump. These were the speed and the load of the pump. The speed of the pump can be changed from 46 Hz to 54 Hz, the loads from 29 to 33 (in 0.1 kW, i.e. from 2.9 to 3.3 kW, the load of the pump is controlled by closing or opening a membrane which controls the water flow). The combination of all the 5 speeds and 3 loads is used as the complete, true data distribution. For all combinations of speeds and loads both the normal running condition vibration data as well as the faulty conditions are measured. Several subsets with different loads and speeds are created and used as training sets. The testing is done on the complete set of loads and speeds. Clearly, the training and testing distribution are not the same. It is hoped that by sampling just the extreme situations in the normal working conditions of the water pump, a description of the complete operating area in the feature space can be obtained. This stresses the importance of the boundary of the target data instead of the density of these objects.

Table 2.3: The training and testing sample sizes for the different vibration datasets with their settings. A speed setting of $46, 50, 54$ Hz and a load setting of $29, 33$ (0.1 kW) mean that all measurements with these settings are used (i.e. 6 situations in total). Measurements from 3 sensors are used.

dataset	settings		training set	testing set	
number	speeds (Hz)	loads (0.1 kW)	target objects	target objects	outlier objects
1	46,48,50,52,54	25,29,33	216	234	666
2	46,50,54	29,33	83	215	685
3	46,54	25,33	64	213	687
4	50	25,29,33	41	223	677
5	46,48,50,52,54	29	78	215	685

In table 2.3 the sample sizes of 5 different datasets are shown. Different combinations of loads and speeds are considered, which results in different training datasets. The first dataset contains all available situations (all combinations of all measured speeds and loads) and has the highest number of training objects. The training set in the third vibration dataset in table 2.3 consists of just 4 combinations: $46 + 25$, $46 + 33$, $54 + 25$ and $54 + 33$, which are the extrema in the pump operation modes. Here just 64 training objects are available. For testing 213 target objects and 687 outlier objects are available. The testing sets are comparable, because they contain all speeds and loads measurements from a damaged pump.

Vibration set 1 is the set which is used in figure 2.18 and which contains all loads and speeds in the training set. In the second set, the situations are subsampled. Situations with speeds of 48 and 52 Hz and with loads of 25 are not considered, but the extreme and some intermediate situations are retained. In vibration set 3 only the extreme values are used. In the last two vibration sets one of the parameters is varied over the complete range, while the other parameter is set at one specific value. The loads are varied in set 4

and the speeds in set 5.

In figure 2.19 the results on vibration sets 2 and 3 are shown. Using just a subsample of the data severely hurts the performance of the one-class methods, with the exception of the SVDD. The Gaussian model completely fails by accepting the complete feature space (thus accepting all target and outlier objects). Performance of the NN-d is somewhat better than the Parzen density, but neither of them is capable of interpolating between the individual clusters of the different speeds and loads. Even when they (try to) accept all target objects, they just accept 50% of the testing target objects.

Using just 4 instead of 6 situations (in the right plot of figure 2.19), the performance of the Parzen and the NN-d deteriorates even more, while the SVDD still shows good performance. The Gaussian model again fails. So, it appears that using just the extreme situations for training are sufficient for the SVDD to give a complete characterization of the normal operation of the water pump.

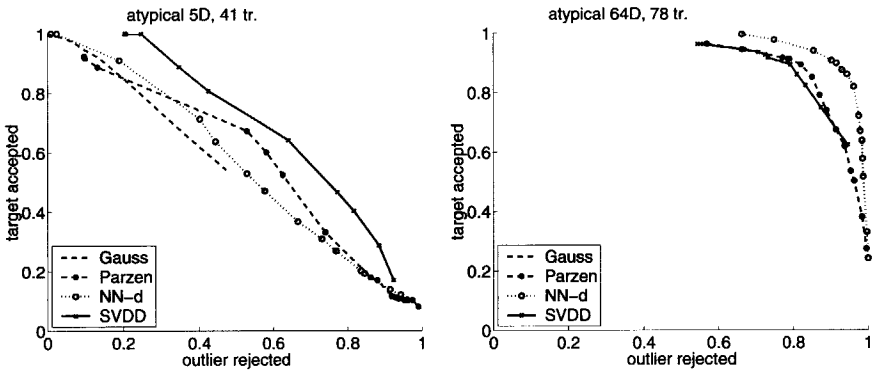


Fig. 2.20: ROC curve for the 4 one-class classifiers trained on subsets for which one of the settings is not varied. In the left plot only a speed of 50 Hz is used (vibration dataset 4), while in the right plot a load of 29 is used (vibration dataset 5).

Finally, in figure 2.20 the results on vibration sets 4 and 5 is shown. Because vibration set 4 just contains 41 objects, the data is mapped onto the first 5 principal components. The left plot shows that using just one speed for training (but retaining all possible loads) severely deteriorates the performance of the one-class classifiers. The SVDD has a slight edge over the other methods, but the performance of all methods is poor. In the right plot we see that using all speeds, but with one load, the data is characterized far better. Best performance here is obtained using the NN-d. This indicates that we are working with a low sample size dataset. The relatively poor performance of the SVDD confirms this.

The characteristics of the dataset, mentioned in the previous section, can also be visualized in a scatter plot. In figure 2.21 a scatter plot of the first 2 principal components of the training set of vibration set 5 is shown. The training objects are marked by a dark circle, the testing objects are shown by light circles. The data is clearly clustered, with 2

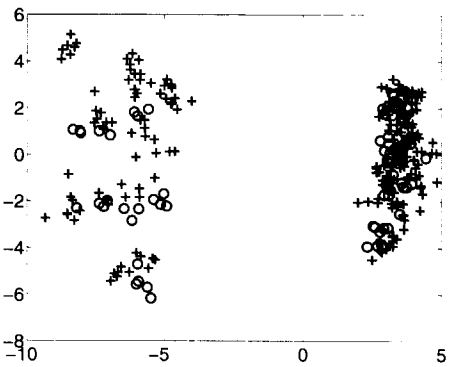


Fig. 2.21: Scatter plot of the first 2 principal components of vibration dataset 5.

large clusters, and 3 subclusters in the right cluster. The difficulty in this dataset is that the 3 subclusters contain just a few training objects each.

In this section we applied the SVDD to a machine diagnostics problem, to characterize the normal operation condition of a water pump. For representative datasets (in which the training set gives a good indication of the 'true' target distribution) and sufficiently large sample sizes conventional density estimators (for instance, Parzen estimation) work well. For high target acceptance rates, the SVDD can be hampered by the fact that it requires a minimum number of support vectors for its description. It will never be able to accept all target objects because there will always be some objects (the support objects) on the decision boundary. For non-representative data (in which the training data is completely different from the 'true' target distribution), the SVDD profits from the fact that it only models the boundary of the data. While the density methods completely fail, shows the SVDD good performance.

2.8 Conclusions

In this chapter we introduced a new method – the support vector data description (SVDD) – to solve the one-class classification problem. It fits a closed boundary around the data set, without estimating a complete probability density. By avoiding the estimation of a complete probability density, it can obtain a better data boundary from a limited data set. The boundary is described by a few training objects, the support vectors. The SVDD has the ability to replace normal inner products by kernel functions to obtain more flexible data descriptions. In contrast to the support vector classifier, the support vector data description using a polynomial kernel suffers from the large influence of the norms of the object vectors. On the other hand, the SVDD gives encouraging results for the Gaussian kernel. Using the Gaussian kernel, descriptions are obtained that are comparable to the hyperplane solution by Schölkopf et al.

An extra feature of the SVDD is, that it offers a direct estimate of the error it makes on the target set. The fraction of the target objects which become support vectors is an estimate of the fraction of target objects rejected by the description. When the maximum desired error on the target set is known beforehand, the width parameter s in the Gaussian kernel can be set to give the desired number of support vectors. When an insufficient number of objects is available, the number of support vectors remains high for any width parameter s used. This is an indication that more data is necessary. Therefore, for very small sample sizes the SVDD breaks down due to its requirement for support vectors. Extra data in the form of outlier objects can also be used to improve the support vector data description.

When the support vector data description is compared to other outlier detection methods, as normal density estimation, Parzen density estimation and a nearest neighbor method (NN-d), not many differences can be observed for identically distributed training and testing data. It appears that the support vector data description is especially useful when the training data distribution differs from the testing distribution and just the boundary of the data has to be found. In these cases the SVDD finds far better descriptions than the other methods. It is also very capable of describing the area in feature space where the target class is located. In the next chapter a more thorough comparison between different one-class classification methods will be presented.

3. ONE-CLASS CLASSIFICATION

Several methods have been proposed to solve the one-class classification problem. Three main approaches can be distinguished: the density estimation, the boundary methods and the reconstruction methods. For each of the three approaches, different concrete models can be constructed. In this chapter we will present and discuss some of these methods and their characteristics. In section 3.1 some important considerations for the one-class classification problem and for the comparison between one-class classification methods will be discussed. In section 3.2 we give some characteristics of one-class classifiers (and classifiers in general) which are of interest when a choice between classifiers have to be made. Section 3.3 describes three density methods, the Gaussian model, mixture of Gaussians and the Parzen density estimators and in section 3.4 the boundary methods containing the k-centers method, the NN-d and the SVDD. Finally, in section 3.5 the reconstruction methods such as k-mean clustering, self-organizing maps, PCA and mixtures of PCA's and diabolo networks are discussed.

These one-class classification methods differ in their ability to cope with or to exploit different characteristics of the data. Important characteristics are the scaling of features in the data, the grouping of objects in clusters, convexity of the data distribution and their placing in subspaces. In this chapter we will just discuss the characteristics of the one-class classifiers, and no experiments will be performed. Experiments to compare these one-class classifiers will be done in the next chapter (chapter 4).

3.1 Considerations

In all one-class classification methods two distinct elements can be identified.The first element is a measure for the distance $d(\mathbf{z})$ or resemblance (or probability) $p(\mathbf{z})$ of an object $\mathbf{z}$ to the target class (represented by a training set $\mathcal{X}^{tr}$).[1] The second element is a threshold θ on this distance or resemblance. New objects are accepted by the description when the distance to the target class is smaller than the threshold θ_d:

$$f(\mathbf{z}) = I(d(\mathbf{z}) < \theta_d) \tag{3.1}$$

or when the resemblance is larger than the threshold θ_p:

$$f(\mathbf{z}) = I(p(\mathbf{z}) > \theta_p) \tag{3.2}$$

[1] Strictly speaking, we should say $p(\mathbf{z}|\omega_T)$ or $d(\mathbf{z}|\omega_T)$, the chance of object $\mathbf{z}$ given target set ω_T, to indicate that we only model the target set and do not use any information about the outlier distribution.

where $I(.)$ is the indicator function (2.14). The one-class classification methods differ in their definition of $p(\mathbf{z})$ (or $d(\mathbf{z})$), in their optimization of $p(\mathbf{z})$ (or $d(\mathbf{z})$) and thresholds with respect to the training set $\mathcal{X}^{tr}$. In most one-class classification methods the focus is on the optimization of the resemblance model p or distance d. The optimization of the threshold is done afterwards. Only a few one-class classification methods optimize their model $p(\mathbf{z})$ or $d(\mathbf{z})$ to an a priori defined threshold (in particular, the SVDD proposed in the previous chapter).

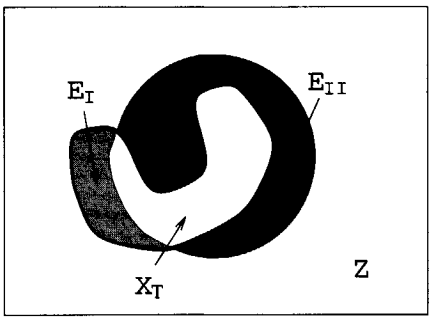


Fig. 3.1: Regions in one-class classification. Around a banana-shaped training set, a spherically shaped one-class boundary is trained. The outliers are uniformly distributed in the rectangular block. The gray areas represent the error of the first and second kind, $\mathcal{E}_{\mathrm{I}}$ and $\mathcal{E}_{\mathrm{II}}$ respectively.

The most important feature of one-class classifiers is the tradeoff between the fraction of the target class that is accepted, f_{T+}, and the fraction of outliers that is rejected, f_{O-} (or equivalently the tradeoff between the error of the first kind and the second kind, $\mathcal{E}_{\mathrm{I}}$ and $\mathcal{E}_{\mathrm{II}}$). The f_{T+} can easily be measured using an independent test set drawn from the same target distribution. To measure the f_{O-} on the other hand, an outlier density has to be assumed. In the experiments on artificial data, performed in this chapter, we assume that the outliers are drawn from a bounded uniform distribution covering the target set and the description volume. This region will be indicated by $\mathcal{Z}$. The fraction of the objects which is accepted by the methods is then an estimate of the covered volume.

This general setup is shown in figure 3.1. The banana-shaped area is the target distribution $\mathcal{X}_T$. The circular boundary is the data description which should describe the data. It makes some errors: a part of the target data is rejected and some outliers are accepted. Note that a tradeoff between f_{O+} and f_{T-} cannot be avoided. Increasing the volume of the data description in order to decrease the error $\mathcal{E}_{\mathrm{I}}$, will automatically increase the number of accepted outliers, and therefore increase the error $\mathcal{E}_{\mathrm{II}}$.

3.1.1 Volume estimation

To obtain an estimate of the volume captured by the description, objects can randomly be drawn from a uniform distribution over the target data. The f_{O+} then gives the fraction

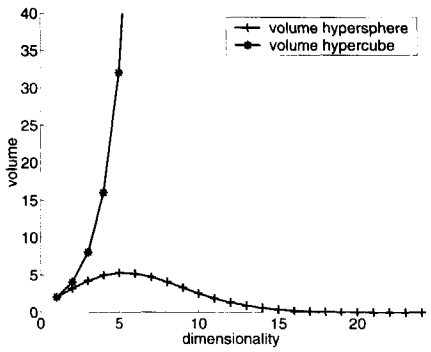


Fig. 3.2: The volume of a unit hypersphere, compared with the volume of a hypercube.

of the outlier volume that is covered by the data description. Unfortunately, the required number of test objects can become prohibitively large, especially in high dimensional feature spaces. To illustrate this, assume the target class is distributed in a hypersphere in d dimensions with radius R centered around the origin: $\mathcal{X} = \{\mathbf{x} : \|\mathbf{x}\|^2 \leq R^2\}$. Assume further that the outliers are restricted to a hypercube in d dimensions: $\mathcal{Z} = [-R, R]^d$. The corresponding volumes become:

$$V_{\mathcal{Z}} = (2R)^d \tag{3.3}$$

and

$$V_{\mathcal{X}} = \frac{2R^d \pi^{d/2}}{d\Gamma(d/2)} \tag{3.4}$$

where $\Gamma(n)$ is the gamma function:

$$\Gamma(n) = 2\int_0^\infty e^{-r^2} r^{2n-1} dr \tag{3.5}$$

In figure 3.2 the volumes of both the hyperbox and the hypersphere are shown as a function of the dimensionality and for $R = 1$.[2] For 1- and 2-dimensional data the respective volumes are almost equal (for $d = 1$ they are equal: $V_{\mathcal{X}} = V_{\mathcal{Z}} = 2$). When d is small ($d < 6$), the volumes are of the same order, but for higher dimensionalities $v_{\mathcal{X}}$ decreases to zero, while $V_{\mathcal{Z}}$ exponentially increases. Consequently, outlier objects drawn from the hypercube will fall with *zero* probability within the hypersphere. No matter how many outlier objects are drawn from the cube, all objects will be rejected by the hypersphere.

[2] Note that, strictly speaking, no continuous graph can be shown, because for each dimensionality the units of the volume change. This graph is given to show the difference in volumes for the hypercube and the hypersphere.

In practice, $\mathcal{X}$ and $\mathcal{Z}$ will have more complex shapes, but when the volumes of the two differ (which means that the outlier set contains the target set, $\mathcal{X} \subset \mathcal{Z}$) their volumes will still diverge with increasing dimensionality d. Consequently, most artificial datasets in this paper will be placed in a reasonably small feature space, with dimensionalities up to $d = 10$. For non-artificial data sets, higher dimensionalities are sometimes investigated, but for these datasets also example outlier objects are available. The volume of the data description will not be measured, but rather the overlap between a more specific outlier distribution (given by the example outliers) and the target distribution.

3.1.2 Error definition

Choosing different thresholds for the distance $d(\mathbf{z})$ or resemblance $p(\mathbf{z})$ results in different tradeoffs between f_{T+} and f_{O-}. To make a comparison between methods, which differ in definitions of $p(\mathbf{z})$ (or $d(\mathbf{z})$), for instance f_{T+}, should be fixed for the methods and the f_{O-} should be measured. A method which obtains the lowest outlier rejection rate, f_{O-}, is then to be preferred.

For some one-class classification methods the distance measure is optimized with respect to a given threshold θ: different thresholds will give a different definition of $p(\mathbf{z})$ or $d(\mathbf{z})$. For most one-class classification methods just one distance or resemblance measure to the training set $\mathcal{X}^{tr}$ is obtained though, independent of the threshold. In this thesis the threshold is derived directly from the training set. It is adjusted to accept a predefined fraction of the target class. For a target acceptance rate f_{T+}, the threshold $\theta_{f_{T+}}$ is defined as:

$$\theta_{f_{T+}} : \frac{1}{N} \sum_i I\left(p(\mathbf{x}_i) \geq \theta_{f_{T+}}\right) = f_{T+}, \quad \text{where} \quad \mathbf{x}_i \in \mathcal{X}^{tr}, \forall_i \tag{3.6}$$

For a distance based method using $d(\mathbf{z})$, an equivalent threshold is defined. To avoid overfitting on the training set, a better estimate might be obtained by using an independent validation set, but this requires more data. Because some methods do not require a separate validation set and are able to find the boundary on the target set immediately, we will use the target set for both the determination of $p(\mathbf{z})$ or $d(\mathbf{z})$ and the threshold $\theta_{f_{T+}}$. Only in one case, in the NN-d, a leave-one-out estimation has to be performed. In the NN-d all training objects have, per definition, a distance zero to the data description. When all objects have the same distance to the data description, it is not possible to reject a fraction of the data. Using leave-one-out estimation, it is possible to obtain a histogram of the resemblances of the training data to the description and to obtain a threshold for several values of f_{T+}.

Using (3.6) we can, for varying f_{T+}, compute a threshold $\theta_{f_{T+}}$ on the training set and we can measure the f_{O-} on a set of example outliers. When for all values of f_{T+}, the f_{O-} is measured, we obtain the Receiver-Operating Characteristic curve [Metz, 1978]. A graphical representation is given in figure 3.3, where the solid line A-B gives the f_{T+} and f_{O-} for varying threshold values $\theta_{f_{T+}}$. In all methods the f_{T+} is changed from (almost)

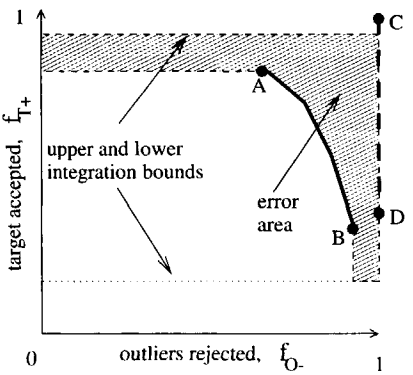


Fig. 3.3: Receiver-Operating Characteristic (ROC) curve with integrated error area.

100% acceptance down to no acceptance. A perfect data description would obtain a vertical ROC curve with $f_{O-} = 1.0$, while still accepting the predefined f_{T+} (the dashed line C-D). In practice the methods will always accept some outliers, depending on the fit of the method onto the data and the volume of the target class (when a uniform distribution of the outliers is assumed, there will always be some outliers falling within the target class). The f_{O-} will therefore be smaller than 1.0 and a curve A-B will be obtained.

Methods will perform differently for different thresholds. A method might have a very tight description for a high f_{T+}, but might fail when large parts of the target class should be rejected (which means a low f_{T+}). To find one error measure for the fit of one-class classifiers to the data, a 1-dimensional error measure is derived from the ROC curves. We set $\mathcal{E}_{\mathrm{I}} = 1 - f_{T+}$ and $\mathcal{E}_{\mathrm{II}} = 1 - f_{O-}$, so no special weightings of the errors is used (where f_{T+} and f_{O-} could be weighted differently). Now $\mathcal{E}_{\mathrm{II}}$ is integrated over varying thresholds (i.e. all possible errors of the first kind $\mathcal{E}_{\mathrm{I}}$) [Bradley, 1997].[3] This gives the error:

$$\mathcal{E}_{\mathrm{M}} = \int_0^1 \mathcal{E}_{\mathrm{II}}(\mathcal{E}_{\mathrm{I}}) d\mathcal{E}_{\mathrm{I}} = \int_0^1 \int_{\mathcal{Z}} I\left(p(\mathbf{z}) \geq \theta_f\right) d\mathbf{z} d\theta_f \tag{3.7}$$

where θ_f is measured on the target set. This error is the grey area shown in figure 3.3. This error measure does not evaluate the one-class classifiers on the basis of one single threshold value, but integrates their performances over all threshold values. (This 'integrating out' of free parameters is more common in the Bayesian treatment of classifiers [MacKay, 1992]).

A practical problem with this 'integrating out' of the threshold, is that the methods might have different effective ranges for f_{T+} (so the positions of A and B in figure 3.3 differ). In the previous chapter (section 2.4) it was shown that the SVDD does not work well in situations with high f_{T+} when limited target data is available. The SVDD requires

[3] Bradley called this error the 'Area Under the (ROC) Curve', AUC.

a minimum number of objects to support the description. When a small number of target objects is available (with respect to the dimensionality and the shape of the target distribution) the f_{T+} in situation A in figure 3.3 will be low (i.e. much smaller than one).

Therefore, the basic method in [Bradley, 1997] is extended to obtain a standard integration range for all methods. Again we consider the ROC curve A-B shown in figure 3.3. Two extreme situations can always be identified. In the first situation $(f_{O-}, f_{T+}) = (1, 0)$. This is a description with zero volume where both the complete target and outlier data are rejected. In the second situation, the complete feature space is accepted, $(f_{O-}, f_{T+}) = (0, 1)$. Given a ROC curve A-B, it is not clear though, how this curve can be extrapolated to these two extreme situations. Here we will set f_{O-} pessimistically to zero, for f_{T+} larger than the f_{T+} of situation A. At the other end of the curve, for very low f_{T+} (for f_{T+} lower than that of situation B), the best f_{O-} over larger target acceptance rates is copied (very likely this is the f_{O-} of situation B). This assumes the worst case scenario in which rejecting more target data does not decrease the f_{O-}.

In the error definition (3.7), the error $\mathcal{E}_{\text{II}}$ is weighted equally over all thresholds. In a practical application, the $\mathcal{E}_{\text{II}}$ might be limited to a tighter range or might be weighted by a weighting function $w(\theta_f)$ [Adams and Hand, 2000]:

$$\mathcal{E}_{\text{M}} = \int_0^1 \int_{\mathbf{z}} I\left(p(\mathbf{z}) \geq \theta_f\right) w(\theta_f) d\mathbf{z} d\theta_f \tag{3.8}$$

We integrate in this chapter from 0.05 to 0.5 with equal weight over the complete range (thus $w(\theta_f) = I(0.05 < \theta_f < 0.5)$, with I again the indicator function, formula (2.14)).

By this weighting we exclude the extreme cases. For $\theta_f < 0.05$, less than 5% of the target data can be rejected. This means that for sample sizes lower than 20 objects, no target object can be rejected and that the threshold on d or p is completely determined by the most dissimilar target object in the training set. We avoid this by taking $\theta_f > 0.05$. The other extreme case is when we use $\theta_f > 0.5$. In that case, more than half of the target objects is rejected. Because our primary goal is to make a description of the target class, and we assume a low density of outlier data, the rejection of more than 50% of our target data is considered too much. We assume we have to describe at least 50% of our target data.

In this section we identified two elements in one-class classifier, namely the distance d or resemblance p to the target set, and a threshold value θ. For one-class classifiers the d or p is often completely defined, but for the determination of θ the performance on both target and outlier data is required. We showed, that when no example outlier objects are available, they can be created artificially, but this becomes extremely inefficient in high dimensional feature spaces. When we have target and outlier data available, we can give the performance of a one-class classifier by an ROC curve or we can measure a scalar quantity: the integrated error of the ROC curve (by definition 3.8). These observations and definitions hold for all one-class classifiers. In the next section we investigate in what one-class classifiers can differ.

3.2 Characteristics of one-class approaches

To compare different one-class classification methods, not only the f_{T+} and f_{O-} are important, but also other features. We will consider the following characteristics:

Robustness to outliers: It has been assumed that the training set is a characteristic representation for the target distribution. It might happen that this training set is already contaminated by some outliers. Although a one-class classification method should accept objects from the target set as much as possible, these outliers should still be rejected. In most methods a more or less strict model of the data is assumed. When in a method only the resemblance or distance is optimized, it can therefore be assumed that objects near the threshold are the candidate outlier objects. For methods where the resemblance is optimized for a given threshold, a more advanced method for outliers in the training set should be applied.

Incorporation of known outliers: When some outlier objects are available, they might be used to further tighten the description. To incorporate this information into the method, the model of the data and the training procedure should be flexible enough. For instance, when one Gaussian distribution is used as the model of the target set, the model and training procedure are not flexible enough to reject the single outlier. Other methods on the other hand, like the Parzen density, can incorporate this outlier in its probability estimate. Ultimately, it should be possible to add a parameter in the one-class classifier which regulates the tradeoff between a target and an outlier error.

Magic parameters and ease of configuration: One of the most important aspects for easy operation of a method by the user, is the number of free parameters that have to be chosen beforehand, as well as their initial values. When a large number of free parameters is involved, finding a good working set might be very hard. This becomes even more prominent when the parameters involved are not intuitive quantities which can be assumed, derived or estimated a priori. When they are set correctly, good performances will be achieved, but when they are set incorrectly, the method might completely fail. These parameters are often called the 'magic parameters' because they often have a big influence on the final performance and no clear rules are given how to set them. For instance, in the training of neural networks, the number of hidden layers and the number of neurons per layer should be chosen by the user. These numbers cannot be intuitively given beforehand, and only by trial and error a reasonable network size is found. Other methods, like the SVDD, require a value for f_{T-} beforehand. This number is directly related to the problem the user tries to solve, and is thus easier to set.

Computation and storage requirements: A final consideration is the computational requirements of the methods. Although computers offer more power and storage every year, methods which require several minutes for the evaluation of a single test

object might be unusable in practice. Since training is often done off-line, training costs are often not very important. However, in the case of adapting the method to a changing environment, these training costs might become very important.

We mentioned some important features for (one-class) classifiers. In the next sections we will compare the three main approaches to one-class classification, namely the density methods, the boundary methods and the reconstruction methods. We will discuss the methods and what their characteristics are with respect to robustness to outliers, incorporation of known outliers, magic parameters and computation requirements. Note that in each of the approaches various methods exist. Most of them are already well known in traditional pattern recognition, so they will be discussed only briefly. One exception is the NN-d, which is a new approach. This method is discussed in more detail in appendix B (and has already been mentioned in section 2.7).

Of course several other one-class classification methods exist, but we have chosen to use just this set of methods because it covers a broad range of possible models. This list has to give an idea of the various approaches under several circumstances, but does not pretend to be an exhaustive enumeration of all possibilities.

3.3 Density methods

The most straightforward method to obtain a one-class classifier is to estimate the density of the training data [Tarassenko et al., 1995] and to set a threshold on this density. Several distributions can be assumed, such as a Gaussian or a Poisson distribution, and numerous tests, called discordancy tests, are then available to test new objects [Barnett and Lewis, 1978]. In this thesis we will consider three density models, the normal model,[4] the mixture of Gaussians and the Parzen density.

When the sample size is sufficiently high and a flexible density model is used (for example a Parzen density estimation), this approach works very well. Unfortunately, it requires a large number of training samples to overcome the curse of dimensionality [Duda and Hart, 1973]. If the dimensionality of the data and the complexity of the density model are restricted, this can be avoided, but then a large bias may be introduced when the model does not fit the data very well. Finding the right model to describe the target distribution and the given sample size is a typical incarnation of the bias-variance dilemma (see chapter 1).

When a good probability model is assumed (one for which the bias is small) and the sample size is sufficient, this approach has a very big advantage. When one threshold is optimized, automatically a minimum volume is found for the given probability density model. By construction, only the high density areas of the target distribution are included. Superfluous areas in $\mathcal{Z}$ will, therefore, not be accepted by the description. In this chapter

[4] In this thesis we will use the terms 'normal model' or 'Gaussian model' interchangeably for a normal density that is fitted to some target data.

the threshold will be determined by using the empirical target distribution (definition (3.6)).

3.3.1 Gaussian model

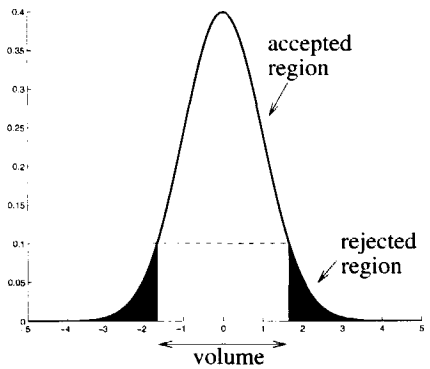


Fig. 3.4: Threshold on a 1-dimensional Gaussian distribution.

The most simple model is the normal or Gaussian density [Bishop, 1995], as shown in figure 3.4. According to the Central Limit Theorem [Ullman, 1978], when it is assumed that objects from one class originate from one prototype and are additively disturbed by a large number of small independent disturbances, then this model is correct. The probability distribution for a d-dimensional object $\mathbf{x}$ is given by:

$$p_{\mathcal{N}}(\mathbf{z};\boldsymbol{\mu},\Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{z}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{z}-\boldsymbol{\mu})\right\} \tag{3.9}$$

where $\boldsymbol{\mu}$ is the mean and Σ is the covariance matrix. The method is very simple and it imposes a strict unimodal and convex density model on the data.

Thus, the number of free parameters in the normal model is

$$n_{\text{free}\mathcal{N}} = d + \frac{1}{2}d(d-1) \tag{3.10}$$

with d parameters for the mean $\boldsymbol{\mu}$ and $d(d-1)/2$ for the covariance matrix Σ. The main computational effort in this method is the inversion of the covariance matrix. In case of badly scaled data or data with singular directions, the inverse of the covariance matrix Σ cannot be calculated and should be approximated by the pseudo-inverse $\Sigma^+ = \Sigma^T\left(\Sigma\Sigma^T\right)^{-1}$ [Strang, 1988] or by applying some regularization (adding a small constant λ to the diagonal, i.e. $\Sigma' = \Sigma + \lambda\mathcal{I}$). In the last case the user has to supply a regularization parameter λ. This is then the only magic parameter present in the normal model. Because

this model utilizes the complete covariance structure of the data, this method is insensitive to scaling of the data (assuming that a reasonable regularization parameter is chosen).

Another advantage of the normal density is, that when we assume that the data is normally distributed, it is possible to compute analytically the optimal threshold for a given f_{T+}. It is known that for d independent normally distributed random variables $\mathbf{x}_i$, the new variable $x' = \sum_i (\mathbf{x}_i - \boldsymbol{\mu}_i)^2/\sigma_i$ is distributed with a χ^2_d distribution (where d is the degrees of freedom) [Ullman, 1978]. When we consider the (squared) Mahanalobis distance, the variable

$$\Delta^2 = (\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}) \tag{3.11}$$

(for given $\boldsymbol{\mu}$ and Σ) should also be distributed like χ^2_d. The threshold $\theta_{f_{T+}}$ on Δ^2 should be set at the prespecified f_{T+}:

$$\theta_{f_{T+}} : \int_0^{f_{T+}} \chi^2_d(\Delta^2) d(\Delta^2) = f_{T+} \tag{3.12}$$

Because in most cases the target data is not normally distributed, this threshold will not be used and we would rely on a threshold on the empirical distribution (using definition (3.6)). We will only use this threshold when we do some experiments on an artificial Gaussian target set and we want to compare the results of the one-class classifiers with the 'true' normal distribution and the 'true' threshold $\theta_{f_{T+}}$.

3.3.2 Mixture of Gaussians

The Gaussian distribution assumes a very strong model of the data. It should be unimodal and convex. For most datasets these assumptions are violated. To obtain a more flexible density method, the normal distribution can be extended to a mixture of Gaussians (MoG) [Duda and Hart, 1973]. A mixture of Gaussians is a linear combination of normal distributions [Bishop, 1995]:

$$p_{MoG}(\mathbf{x}) = \frac{1}{N_{MoG}} \sum_j \alpha_j p_{\mathcal{N}}(\mathbf{x}; \boldsymbol{\mu}_j, \Sigma_j) \tag{3.13}$$

where α_j are the mixing coefficients. It has a smaller bias than the single Gaussian distribution, but it requires far more data for training (and thus shows more variance when a limited amount of data is available). When the number of Gaussians N_{MoG} is defined beforehand by the user, the means $\boldsymbol{\mu}_j$ and covariances Σ_j of the individual Gaussian components can efficiently be estimated by an expectation minimization routine [Bishop, 1995, Bishop, 1994a].

The total number of free parameters in the mixture of Gaussians is:

$$n_{\text{free}\,MoG} = \left(d + \frac{d(d+1)}{2} + 1 \right) \cdot N_{MoG} \tag{3.14}$$

that is one mean $\boldsymbol{\mu}_j$, one covariance matrix Σ_j and one weight α_j for each component. To reduce the number of free parameters, often just diagonal covariance matrices are assumed, i.e. $\Sigma_j = \text{diag}(\boldsymbol{\sigma}_j)$. The number of free parameters then drops to:

$$n_{\text{free}MoG} = (2d + 1) \cdot N_{MoG} \tag{3.15}$$

In the experiments in the next chapter the covariance matrix will be restricted even further. Because there the sample sizes are sometimes very low (down to 10 objects in 10 dimensions), only one variance for all features can be estimated in order to avoid numerical instabilities, i.e. $\Sigma_j = \sigma_j \mathcal{I}$. The number of free parameters is therefore:

$$n_{\text{free}MoG} = (d + 2) \cdot N_{MoG} \tag{3.16}$$

3.3.3 Parzen density estimation

The third method is the Parzen density estimation [Parzen, 1962], which is again an extension of the previous method. The estimated density is a mixture of, most often, Gaussian kernels centered on the individual training objects, with (often) diagonal covariance matrices $\Sigma_i = h\mathcal{I}$.

$$p_p(\mathbf{x}) = \frac{1}{N} \sum_i p_N(\mathbf{x}; \mathbf{x}_i, h\mathcal{I}) \tag{3.17}$$

The equal width h in each feature direction means that the Parzen density estimator assumes equally weighted features and it will, therefore, be sensitive to the scaling of the feature values of the data, especially for lower sample sizes.

Training a Parzen density consists of the determination of one single parameter, the optimal width of the kernel h. Thus:

$$n_{\text{free}p} = 1 \tag{3.18}$$

Here the h is optimized using the maximum likelihood solution [Kraaijveld and Duin, 1991]. Because just one parameter is estimated, and there are no magic parameters to set by the user, the data model is very weak (i.e. a non-parametric model). A good description totally depends on the representativity of the training set. The computational cost for training a Parzen density estimator is almost zero, but the testing is expensive. All training objects have to be stored and, during testing, distances to all training objects have to be calculated and sorted. This might severely limit the applicability of the method, especially when large datasets in high dimensional feature spaces are considered.

3.4 Boundary methods

Vapnik argued in [Vapnik, 1998] that when just a limited amount of data is available, one should avoid solving a more general problem as an intermediate step to solve the

original problem. To solve this more general problem more data might be required than for the original problem. In our case this means that estimating a complete data density for a one-class classifier might also be too demanding when only the data boundary is required. In the boundary methods, therefore, only a closed boundary around the target set is optimized. In this thesis we consider the k-centers method, the NN-d and the SVDD.

Although the volume is not always actively minimized in the boundary methods, most methods have a strong bias towards a minimal volume solution. How small the volume is, depends on the fit of the method to the data. In most cases distances or weighted distances d to an (edited) set of objects in the training set are computed. Due to their focus on the boundary, the threshold on the output is always obtained in a direct way.

Because the boundary methods heavily rely on the distances between objects, they tend to be sensitive to scaling of the features. On the other hand, the number of objects that is required, is smaller than in case of the density methods. So, although the required sample size for the boundary methods is smaller than the density methods, a part of the burden is now put onto well-defined distances.

Note that the output of these boundary methods cannot be interpreted as a probability. Assume that a method is trained such that a fraction f of the target set is rejected. It obtains a threshold θ_f for this rejection rate on its particular resemblance measure d (which is the replacement of $p(\mathbf{x})$ in (3.6)). Decreasing the threshold θ now tightens the description, but does not guarantee that the high density areas are captured. Changing the f might therefore require the retraining of the method. This is most prominent in the SVDD. The other methods can be adapted by just adjusting their thresholds.

3.4.1 K-centers

The first boundary method is the k-center method which covers the dataset with k small balls with equal radii [Ypma and Duin, 1998]. This description has a resemblance to the covering numbers of Kolmogorov [Kolmogorov and Tikhomirov, 1961] and it is used to characterize the complexity of an (unlabeled) dataset. Unlike with the covering numbers, here the ball centers' $\boldsymbol{\mu}_k$ are placed *on* training objects such that the maximum distance of all minimum distances between training objects and the centers is minimized. In the fitting of the method to the training data, the following error is minimized:

$$\mathcal{E}_{\text{k-centr}} = \max_i \left(\min_k \|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2 \right) \tag{3.19}$$

The k-centers method uses a forward search strategy starting from a random initialization. The radius is determined by the maximum distance to the objects that the corresponding ball should capture. By this construction the method is sensitive to the outliers in the training set, but it will work well when clear clusters are present in the data.

When the centers have been trained, the distance from a test object $\mathbf{z}$ to the target set can be calculated. This distance is now defined as:

$$d_{k-centr}(\mathbf{z}) = \min_k \|\mathbf{z} - \boldsymbol{\mu}_k\|^2 \tag{3.20}$$

To avoid a suboptimal solution during training, several random initializations are tried and the best solution (in terms of the smallest $\mathcal{E}_{\text{k-centr}}$) is used. The number of parameters which is optimized in the minimization of error (3.19) may seem to be kd at first sight, for the k centers $\boldsymbol{\mu}_k$ in d dimensions. But the centers of the balls are constrained to training objects, and therefore only k indices out of N indices have to be selected. The number of free parameters is therefore:

$$n_{\text{free}k-c} = k \tag{3.21}$$

The user has to supply both the number of balls k and the maximum number of retries.

3.4.2 Nearest neighbor method

The second method is a nearest neighbor method, NN-d. It can be derived from a local density estimation by the nearest neighbor classifier [Duda and Hart, 1973]. The method avoids the explicit density estimation and only uses distances to the first nearest neighbor. It is comparable (but not identical) to the method presented in [Knorr et al., 2000] and [Breunig et al., 2000], where an outlier detector is applied to large databases. In the nearest neighbor density estimation a cell, often an hypersphere in d dimensions, is centered around the test object $\mathbf{z}$. The volume of this cell is grown until it captures k objects from the training set. The local density is then estimated by:

$$p_{\text{NN}}(\mathbf{z}) = \frac{k/N}{V_k(\|\mathbf{z} - \text{NN}_k^{tr}(\mathbf{z})\|)} \tag{3.22}$$

where $\text{NN}_k^{tr}(\mathbf{z})$ is the k nearest neighbor of $\mathbf{z}$ in the training set and V_k is the volume of the cell containing this object.

In the one-class classifier NN-d, a test object $\mathbf{z}$ is accepted when its local density is larger or equal to the local density of its (first) nearest neighbor in the training set $\text{NN}^{tr}(\mathbf{z}) = \text{NN}_1^{tr}(\mathbf{z})$. For the local density estimation, $k = 1$ is used:

$$f_{\text{NN}^{tr}}(\mathbf{z}) = I\left(\frac{1/N}{V(\|\mathbf{z} - \text{NN}^{tr}(\mathbf{z})\|)} \geq \frac{1/N}{V(\|\text{NN}^{tr}(\mathbf{z}) - \text{NN}^{tr}(\text{NN}^{tr}(\mathbf{z}))\|)}\right) \tag{3.23}$$

This is equivalent to:

$$f_{\text{NN}^{tr}}(\mathbf{z}) = I\left(\frac{V(\|\mathbf{z} - \text{NN}^{tr}(\mathbf{z})\|)}{V(\|\text{NN}^{tr}(\mathbf{z}) - \text{NN}^{tr}(\text{NN}^{tr}(\mathbf{z}))\|)} \leq 1\right) \tag{3.24}$$

When we now use a d-dimensional cell, its volume becomes (compare to formula (3.4)):

$$V(\mathbf{r}) = V_d \mathbf{r}^d \tag{3.25}$$

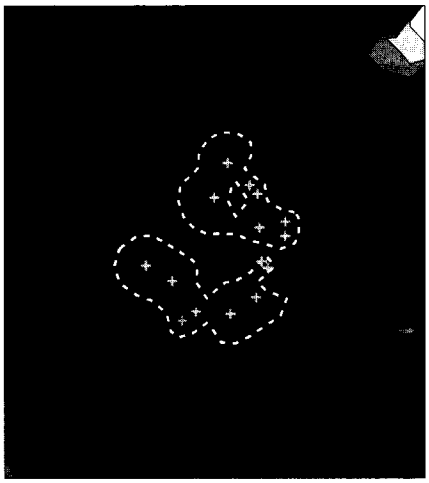


Fig. 3.5: NN-d applied to the artificial 2D banana dataset. The boundary is indicated by the dashed line. The gray values indicate the distance to the boundary, measured by the quotient in (3.26).

where V_d is the volume of a unit hypersphere in d dimensions. This can be substituted into (3.23) and rewriting gives:

$$\begin{aligned} f_{\mathrm{NN}^{tr}}(\mathbf{z}) &= I\left(\frac{V(\|\mathbf{z}-\mathrm{NN}^{tr}(\mathbf{z})\|)}{V(\|\mathrm{NN}^{tr}(\mathbf{z})-\mathrm{NN}^{tr}(\mathrm{NN}^{tr}(\mathbf{z}))\|)} \leq 1\right) \\ &= I\left(\frac{V_d\|\mathbf{z}-\mathrm{NN}^{tr}(\mathbf{z})\|^d}{V_d\|\mathrm{NN}^{tr}(\mathbf{z})-\mathrm{NN}^{tr}(\mathrm{NN}^{tr}(\mathbf{z}))\|^d} \leq 1\right) \\ &= I\left(\frac{\|\mathbf{z}-\mathrm{NN}^{tr}(\mathbf{z})\|}{\|\mathrm{NN}^{tr}(\mathbf{z})-\mathrm{NN}^{tr}(\mathrm{NN}^{tr}(\mathbf{z}))\|} \leq 1\right) \end{aligned} \tag{3.26}$$

This means that the distance from object $\mathbf{z}$ to its nearest neighbor in the training set $\mathrm{NN}^{tr}(\mathbf{z})$ is compared with the distance from this nearest neighbor $\mathrm{NN}^{tr}(\mathbf{z})$ to *its* nearest neighbor. Although this rule can be derived from two local density estimates (one around the test object $\mathbf{z}$ and one around the first nearest neighbor $\mathrm{NN}^{tr}(\mathbf{z})$), the fact that we take $k = 1$ and use the quotient of the densities, makes it possible to apply several simplifications. This then avoids the explicit computation of densities and only the boundary is approximated.

Figure 3.5 shows an example where the NN-d is applied to a banana-shaped target distribution. The data is plotted using pluses; the boundary of the data set is indicated by the dashed line and the gray values give the distance to the boundary measured by the quotient in formula (3.26). When two objects are near, the boundary becomes very tight, but for a uniform sparse distribution the method extrapolates and larger regions around the data are accepted.

In [Breunig et al., 2000] a comparable measure is introduced, the Local Outlier Factor (LOF), which is constructed to find outliers in large databases. Here, all the distances to the k nearest neighbors are averaged, and furthermore, the distance of an object $\mathbf{z}$ to its k nearest neighbors is replaced by a more robust distance definition. When objects are very near the target data, the k-th nearest neighbor distance is used, instead of the first nearest neighbor distance. This robust measure makes it hard to distinguish between objects which are near the boundary or which are deep within a tight cluster of objects. Furthermore, this algorithm requires that the user defines the number of neighbors k, which is taken in the range from 10 to 50. Although the LOF method is very well suited to find outliers in large data sets, it is hard to compare with the other methods on small data sets without clear outliers (objects on the boundary are not rejected, and large rejection rates are therefore not possible).

A characteristic of the NN-d is that it also rejects parts of the feature space which are within the target distribution. To illustrate this, assume we have a uniform target distribution. When we generate training objects from this distribution, by definition, all these target objects should be accepted. When we leave one object $\mathbf{x}$ out of the training, it would only be accepted by the description when it would become the nearest neighbor of its nearest neighbor in the training set. Thus, only the fraction of objects which are mutual nearest neighbors will be consistently accepted. This rate is independent of the sample size, but does depend on the dimensionality of the data. This can be observed in table 3.1 where the fraction of the target data (drawn from a uniform distribution) is shown which is accepted by the NN-d. For instance, when we consider a uniform distribution in 2 dimensions and we use $k = 1$, only 64% of the target data will be accepted by the NN-d.

Table 3.1: Fraction of objects drawn from a uniform distribution accepted by the NN-d. The results are averaged over 1000 runs; the standard deviation is shown between brackets.

	dimensionality				
	1	2	3	5	10
$k = 1$	0.67 (0.09)	0.64 (0.09)	0.62 (0.09)	0.62 (0.09)	0.61 (0.09)
$k = 2$	0.89 (0.06)	0.86 (0.07)	0.85 (0.07)	0.84 (0.06)	0.83 (0.07)
$k = 3$	0.97 (0.03)	0.94 (0.03)	0.94 (0.04)	0.93 (0.04)	0.91 (0.04)

Obviously, the method can easily be extended to a larger number of neighbors k. Instead of taking the first nearest neighbor into account, the kth neighbor should be considered. This increases the acceptance rate both inside and outside the target set. For low sample sizes the local character of the method will, however, be lost.

Finally, the NN-d does not have any free parameters, neither user defined 'magic' parameters, nor free parameters to optimize (we have chosen $k = 1$ in the definition of NN-d, beforehand) and thus it completely relies on the training samples. Because the

NN-d explicitly uses the distances in the evaluation of a test object (formula (3.26), this method is scale sensitive. For a more elaborate treatment of the NN-d, see appendix B.

3.4.3 Support vector data description

Finally, the support vector data description is considered. The SVDD has been described in detail in the previous chapter. In the experiments in this chapter the Gaussian kernel will be used. The free width parameter s is optimized for a specified target rejection rate f_{T-} using the procedure sketched in section 2.4, page 37. Thus, error (2.42) is minimized and the resulting $\boldsymbol{\alpha}$ are used in formula (2.43) to evaluate a test object $\mathbf{z}$. After the optimization of (2.42) the threshold value for the SVDD ($\frac{1}{2}(B - R^2)$) is completely determined and, therefore, the empirical threshold as given in (3.6) is not used. This also means, that for each different value of f_{T-} ($= 1 - f_{T+}$) the SVDD has to be optimized again.

When the width parameter s is optimized for small f_{T+} rates, we have to take care we do not overtrain. For small f_{T+} the parameter s tends to be very small with respect to distances between objects in the training set. Such situations are shown in the left subplot of figure 2.4 and 2.9. To avoid overfitting, the tradeoff parameter between the volume of the description and the number of errors on the target set, C (error (2.3)) will be set to reject up to half of the user specified f_{T-}. This will result in descriptions as shown in the right plot of figure 2.9. This means that when the user requests for $f_{T-} = 0.5$, 25% of the target objects will be outside the description, and 25% will become support vectors on the decision boundary (or $f_{SV}^{\text{out}} = 0.25$ and $f_{SV}^{\text{bnd}} = 0.25$).

The free parameters in the SVDD are the Lagrange multipliers $\boldsymbol{\alpha}$. From these Lagrange multipliers, the center $\mathbf{a}$ and the threshold value R can be computed. Thus, the number of free parameters in the SVDD is:

$$n_{\text{free}SVDD} = N \tag{3.27}$$

The user only has to supply f_{T-} and a fraction of objects which might be incorrectly classified f_{SV}^{out}. The fact that the SVDD uses f_{T-} in its optimization is unique for this one-class classifier. All other one-class classifiers require this value afterwards to set a threshold on p or d. We therefore chose not to count f_{T-} as a user defined magic parameter. The only true magic parameter in the SVDD is the f_{SV}^{out}.

Finally, it is expected that the SVDD is sensitive to the scaling of the feature values (by the use of the Gaussian kernel; definition (2.41)) and that it requires a certain minimum number of training objects, see section 2.6.2.

3.5 Reconstruction methods

The reconstruction methods have not been primarily constructed for one-class classification, but rather to model the data. By using prior knowledge about the data and making assumptions about the generating process, a model is chosen and fitted to the data. New objects can now be described in terms of a state of the generating model. We assume that

a more compact representation of the target data can be obtained and that in this encoded data the noise contribution is decreased. This representation then simplifies further processing without harming the information content.

Most of these methods make assumptions about the clustering characteristics of the data or their distribution in subspaces. A set of prototypes or subspaces is defined and a reconstruction error is minimized. We will consider the following examples of reconstruction methods: the k-means clustering, learning vector quantization, self-organizing maps, PCA, a mixture of PCAs, diabolo networks and auto-encoder networks. The methods differ in the definition of the prototypes or subspaces, their reconstruction error and their optimization routine.

With the application of the reconstruction methods, it is assumed that outlier objects do not satisfy the assumptions about the target distribution. The outliers should be represented worse than true target objects and their reconstruction error should be high. The reconstruction error of a test object by the data compression methods is therefore used as a distance to the target set. Because these methods were not developed for one-class classification, the empirical threshold $\theta_{f_{T+}}$ has to be obtained using the training set.

3.5.1 k-means, LVQ, SOM

The simplest reconstruction methods are the k-means clustering [Bishop, 1995] and the Learning Vector Quantization (LVQ) [Carpenter et al., 1991]. In these methods it is assumed that the data is clustered and can be characterized by a few prototype objects or codebook vectors $\boldsymbol{\mu}_k$. Most often the target objects are represented by the nearest prototype vector measured by the Euclidean distance. The prototype vectors define a Voronoi tessellation of the complete feature space. In the k-means clustering, the placing of the prototypes is optimized by minimizing the following error:

$$\mathcal{E}_{\text{k-m}} = \sum_i \left(\min_k \|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2 \right) \tag{3.28}$$

Different errors are minimized in the LVQ and the SOM, but we will come to that later. First we explain the differences between k-means clustering and the k-center method (from section 3.4.1).

The k-means clustering method resembles the k-center method (from section 3.4.1), but the important difference is the error which is minimized. The k-center method focuses on the worst case objects (i.e. the objects with the largest reconstruction error) and tries to optimize the centers and the radii of the balls to accept all data. In the k-means method the distances to the prototypes of all objects are averaged, and therefore the method is more robust against remote outliers. Furthermore, in the k-center method the centers are placed, per definition, on some of the training objects, while in the k-means method, all center positions are free.

In figure 3.6 examples of the boundaries of the k-means method and the k-center method are shown. The boundaries are set to reject 20% of the target data. The placing of the centers of the hyperspheres by both methods is very similar. The exact positions are

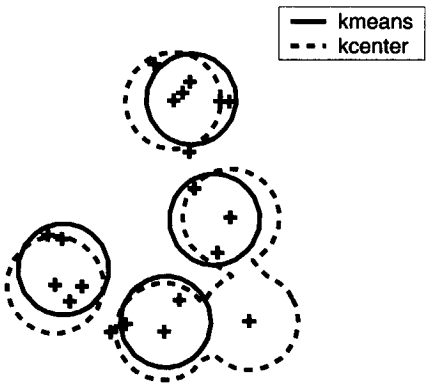


Fig. 3.6: Comparison between the boundary obtained by the k-means method and the k-center method. The threshold is set such that 20% of the target objects is rejected.

determined by the extreme objects for the k-centers and by the means of the different data clusters for the k-means clustering. The most appealing difference is that the k-center method places a hypersphere on the object in the lower right of the dataset, while the k-means method treats it as an outlier.

The distance d of an object $\mathbf{z}$ to the target set is then defined as the squared distance of that object to the nearest prototype:

$$d_{k-m}(\mathbf{z}) = \min_k \|\mathbf{z} - \boldsymbol{\mu}_k\|^2 \tag{3.29}$$

Different minimization routines exist for minimizing the k-means error (3.28). Here we will use a batch algorithm, which optimizes the prototypes $\boldsymbol{\mu}_k$ with an algorithm comparable to the EM-algorithm of the mixture of Gaussians, in section 3.3.2. The algorithm starts with a random placement of the prototypes. All objects $\mathbf{x}$ are then assigned to the nearest prototype and the prototype is updated to the mean of this set of objects. The re-estimation continues until the prototype places are stable. It appears that this k-means algorithm is the limiting situation for the EM optimization in a mixture of Gaussians [Bishop, 1995].

The LVQ algorithm is a supervised version of the k-means clustering and is mainly used for classification tasks.[5] For each of the training objects $\mathbf{x}_i$ an extra label y_i is available, indicating to which cluster it should belong. The LVQ is trained as a conventional neural network, with the exception that each hidden unit is a prototype, where for each prototype $\boldsymbol{\mu}_k$ a class label y_k is defined. It basically minimizes the classification error (error (1.4)),

[5] In [Roweis and Ghahramani, 1997] it is shown that the LVQ and the Mixture of Gaussians (and the Principal Component Analysis which we will encounter in the next section) can be unified to one single basic model. We will treat these methods separately to highlight the differences between them.

and not the error $\mathcal{E}_{\text{k-m}}$ (3.28). The training algorithm is such that it only updates the prototype nearest to the training object $\mathbf{x}_i$:

$$\Delta\boldsymbol{\mu}_k = \begin{cases} +\eta\,(\mathbf{x}_i - \boldsymbol{\mu}_k) & \text{if } y_i = y_k \\ -\eta\,(\mathbf{x}_i - \boldsymbol{\mu}_k) & \text{otherwise} \end{cases} \tag{3.30}$$

where η is the learning rate. This update rule is iterated over all training objects, until convergence is reached.[6] When just one class of objects is available, this is the direct derivative of the error function (3.29).

In both the k-means clustering and the LVQ the k means $\boldsymbol{\mu}$ have to be estimated. Therefore, the number of free parameters becomes:

$$n_{\text{free}k-m} = n_{\text{free}LVQ} = kd \tag{3.31}$$

In both the LVQ and the k-means method, the user should supply the number of clusters k and the LVQ further requires the learning rate η.

In the self-organizing map (SOM) the placing of the prototypes is not only optimized with respect to the data, but also constrained to form a low-dimensional manifold [Kohonen, 1995]. When the algorithm is converged, prototypes corresponding to nearby vectors on the manifold have nearby locations in the feature space. Often a 2- or 3-dimensional regular square grid is chosen for this manifold such that data mapped on this manifold can be visualized afterwards. Higher dimensional manifolds are possible, but the storage and optimization costs become prohibitive (for a d_{SOM} dimensional manifold, the number of neurons becomes $k^{d_{SOM}}$). When the dimensionality of the manifold does not fit the data, this topological constraint on the placing of the prototypes might result in suboptimal placing.

Thus, in the optimization of the SOM $k^{d_{SOM}}$ neurons have to be placed in the d-dimensional feature space. This means that the number of free parameters in the SOM becomes:

$$n_{\text{free}SOM} = dk^{d_{SOM}} \tag{3.32}$$

The dimensionality of the manifold d_{SOM}, the number of prototypes per manifold dimensionality k and the learning rate are supplied by the user. Furthermore, the user should also define a neighborhood function over the grid, which can even change during training. We used the defaults in the SOM-tool Matlab toolbox [Vesanto et al., 2000], i.e. a Gaussian neighborhood which decreases in size over time.

In all of these methods the Euclidean distance is used in the definition of the error and the computation of the distance.

$$d_{SOM}(\mathbf{z}) = \min_k \|\mathbf{z} - \boldsymbol{\mu}_k\|^2 \tag{3.33}$$

Consequently, all these methods are sensitive to scaling of the features.

[6] or when the user is out of patience.

3.5.2 Principal Component Analysis

Principal Component Analysis (PCA) (or the Karhunen-Loève transform) [Bishop, 1995] is used for data distributed in a linear subspace. The PCA mapping finds the orthonormal subspace which captures the variance in the data as best as possible (in the squared error sense). Neural network approaches exist for the optimization of a PCA (we will encounter one in the section 3.5.4 about auto-encoders). The simplest optimization procedure uses eigenvalue decomposition to compute the eigenvectors of the target covariance matrix. The eigenvectors with the largest eigenvalues are the principal axis of the d-dimensional data and point in the direction of the largest variance. These vectors are used as basis vectors for the mapped data. The number of basis vectors M is optimized to explain a certain, user defined, fraction of the variance in the data. The basis vectors $\mathbf{W}$ become a $d \times M$ matrix. Because they from an orthonormal basis, the number of free parameters in the PCA becomes:

$$n_{\text{free}PCA} = \binom{d-1}{M} \tag{3.34}$$

In PCA it is often assumed that the data has zero mean. When the mean of the data has to be estimated also, this will add another d free parameters to $n_{\text{free}PCA}$.

The reconstruction error of an object $\mathbf{z}$ is now defined as the squared distance from the original object and its mapped version:

$$d_{PCA}(\mathbf{z}) = \left\|\mathbf{z} - (\mathbf{W}(\mathbf{W}^T\mathbf{W})^{-1}\mathbf{W}^T)\mathbf{z}\right\|^2 = \left\|\mathbf{z} - (\mathbf{W}\mathbf{W}^T)\mathbf{z}\right\|^2 \tag{3.35}$$

where the second step is possible because the basis $\mathbf{W}$ is orthonormal.

The PCA performs well when a clear linear subspace is present. Also for very low sample sizes the data is automatically located in a subspace (10 objects are always distributed in a 9-dimensional subspace). When the intrinsic dimensionality of the data is smaller than the feature size, the PCA can still generalize from the low sample size data. When the data has variance in all feature directions, it might sometimes be impossible to reduce the dimensionality without reducing the fraction of the explained variance too much. For instance, when the user requests that 90% of the variance of some 2-dimensional data should be explained, it might happen that each of the two PCA features explain about 50% of the variance. In this case, no feature reduction can be applied and the complete feature space is described by the two features. Therefore, all data will be accepted. Also when data is distributed in separate subspaces, the PCA will produce an average subspace which may represent the data in each subspace very poorly.

The PCA is relatively sensitive to the scaling of the features, it directly influences the feature variances. Scaling changes the order of the large variance directions and thus the PCA basis. When data directions are enhanced, this improves the PCA description, but when noise is amplified, it harms the characterization. Finally, because the PCA only focuses on the variance of the target set, the PCA is incapable of including negative examples in the training.

3.5.3 Mixtures of Principal Component Analyzers

Like the extension of the Gaussian model to the mixture of Gaussians, the PCA can be extended to a mixture of PCA's. The extension of the PCA is far less obvious, though. For the optimization of the original PCA no probability distribution is assumed or optimized. The combination of several principal component analyzers then becomes a bit ad hoc: when is an object $\mathbf{z}$ described by subspace A and not by subspace B?

In [Tipping and Bishop, 1999] the PCA is reformulated within a maximum likelihood framework with a probability model for the data. It is, therefore, also called the probabilistic principal component analysis. By introducing several principal component bases, a mixture of probability models can be optimized. Each model identifies a subspace $\mathbf{W}_k$ in the data. The resemblance of a new object $\mathbf{z}$ to the total mixture becomes:

$$p_{MPCA}(\mathbf{z}) = \sum_k \left((2\pi)^{-d/2} |\mathbf{C}_k|^{-1/2} \exp\left\{ -\frac{1}{2}(\mathbf{z}-\boldsymbol{\mu}_k)^T \mathbf{C}_k^{-1} (\mathbf{z}-\boldsymbol{\mu}_k) \right\} \right) \tag{3.36}$$

where $\mathbf{C}_k = \sigma^2 \mathcal{I} + \mathbf{W}_k \mathbf{W}_k^T$ is the covariance matrix in subspace $\mathbf{W}_k$. In the mixture of PCAs the means $\boldsymbol{\mu}_k$, the subspace directions $\mathbf{W}_k$ and the covariance matrices $\mathbf{C}_k$ have to be estimated from the data. These parameters can be estimated by an EM algorithm [Tipping and Bishop, 1999] which minimizes the log-likelihood of the training set. Now for each object a probability that it is explained by one of the subspaces can be computed and a well-defined PCA mixture model can be constructed.

The number of free parameters in this method is very high and therefore the required sample size is high. Not only do we have to estimate the basis vectors for each subspace, but also the mean $\boldsymbol{\mu}_k$ of each of the subspaces has to be estimated, and finally, the noise variance σ outside the subspaces. Adding all free parameters for the mixture of PCAs gives:

$$n_{\text{free}MPCA} = \left(\binom{d-1}{M} + d \right) N_{MPCA} + 1 \tag{3.37}$$

By the estimation of the full covariance matrix the method can adapt different scales of features, but it can be very sensitive to outliers.

3.5.4 Auto-Encoders and Diabolo networks

Auto-encoders [Japkowicz et al., 1995] and diabolo networks are neural network approaches to learn a representation of the data [Hertz et al., 1991, Baldi and Hornik, 1989]. Both methods are trained to reproduce the input patterns at their output layer (so they should perform the identity operation). We will distinguish between the auto-encoders and the diabolo networks by the number of hidden layers and the sizes of the layers; see the pictures in figure 3.7. The units are represented by circles, the weights by the arrows. In the auto-encoder architecture just one hidden layer is present with a large number h_{auto} of hidden units. We will use conventional sigmoidal transfer functions. In the diabolo network three

hidden layers with non-linear sigmoidal transfer functions are used and the second layer contains a very low number of hidden units (h_{diab}). This is therefore called the bottleneck layer. The other two layers can have an arbitrary number of hidden units (larger than in the bottleneck layer). Here, we will choose to use two[7] times as many as in the bottleneck layer.

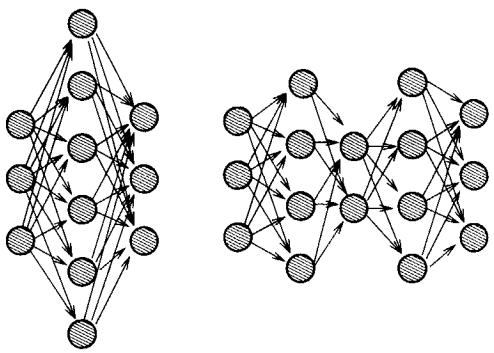


Fig. 3.7: Schematic picture of an auto-encoder network (left) and a diabolo network (right). The circles represent the neurons, the arrows the weights. Both networks have an equal number of input and output units and differ in their definition of the hidden layers. For the auto-encoder network we have $h_{auto} = 6$ and for the diabolo network $h_{diab} = 2$.

Both types of network are trained by minimizing the mean square error (using error $\mathcal{E}_{\text{MSE}}$ (1.3) with (1.5), page 4). It is hoped that target objects will be reconstructed with smaller error than outlier objects. The distance between the original object and the reconstructed object then a measure of the distance of an object $\mathbf{z}$ to the training set:

$$d_{diab}(\mathbf{z}) = \|f_{diab}(\mathbf{z};\mathbf{w}) - \mathbf{z}\|^2 \tag{3.38}$$

$$d_{auto}(\mathbf{z}) = \|f_{auto}(\mathbf{z};\mathbf{w}) - \mathbf{z}\|^2 \tag{3.39}$$

When just one hidden layer is used in the auto-encoder network, a (linear) principal component type of solution is found [Bourlard and Kamp, 1988]. It means that the auto-encoder network tends to find a data description which resembles the PCA description. On the other hand, the small number of neurons in the bottleneck layer of the diabolo network acts as an information compressor. To obtain a small reconstruction error on the target set, the network is forced to train a compact mapping of the data into the subspace coded by these hidden neurons. The number of hidden units in the smallest layer gives the dimensionality of this subspace. Due to the non-linear transfer functions of the neurons in the other layers, this subspace can become non-linear. When the size of this subspace matches the subspace in the original data, the diabolo network can perfectly reject objects which are not in the target data subspace. When the subspace is as large as the original feature space, no distinction between target and outlier data can be expected.

[7] Magic parameter!

The number of free parameters can be very high for both the auto-encoder as the diabolo network. The number of input and output neurons is given by the dimensionality d of the data. First define h_{auto} as the number of hidden units in the auto-encoder. The total number of weights in the auto-encoder (including bias terms) is:

$$n_{\text{free}auto} = d \times h_{auto} + h_{auto} + h_{auto} \times d + d = (2d+1)h_{auto} + d \tag{3.40}$$

(i.e. the number of weights from the input to the hidden layer, the biases in the hidden layer, the weights from the hidden to the output layer and the biases in the output layer).

For the diabolo network we have both the number of neurons in the bottleneck layer, h_{diab}, and the number of neurons in the other hidden layers, which we have chosen as $2h_{auto}$:

$$\begin{aligned} n_{\text{free}diab} &= d \times 2h_{diab} + 2h_{diab}(h_{diab}+1) + h_{diab}(2h_{diab}+1) + 2h_{diab}(d+1) + d \\ &= h_{diab}(4d + 4h_{diab} + 5) + d \end{aligned} \tag{3.41}$$

In the application of the auto-encoder and the diabolo networks, the same problems as in the conventional application of neural networks to classification problems are inherited. The methods are very flexible, but they require a predefined number of layers and neurons, the learning rates and the stopping criterion from the user. The advantage of these models is that they can be optimized to the problem at hand and that they can therefore obtain very good results. For non-expert users on the other hand the settings of the magic parameters might be a problem and performance of the networks can become very poor.

3.6 Robustness and outliers

In the previous three sections we listed a large number of one-class classifiers. Each of them have different characteristics concerning the number of parameters, the robustness against outliers etc. In the next two sections we shortly evaluate these characteristics to give a better overview of what can be expected of the different one-class classification methods.

An important characteristic of the one-class classifiers is their robustness against a few outliers in the training data. Using the empirical rejection rate on the training set (formula (3.6) page 60) to obtain a threshold for the density or distance measure, some robustness is automatically incorporated. Some methods are inherently noise sensitive though, and a few outliers might severely influence the final data description.

In table 3.2 the outlier robustness for the different one-class classifiers is listed, for both, a labeled and an unlabeled outlier. For instance, when we consider the Gaussian model, it is not robust against outliers in the training data (indicated by the '–' in the second column in the table). The outlier in the training data harms the estimation of the covariance matrix. But when regularization is applied, the Gaussian model can identify large outliers. When the outlier object is labeled as an outlier, the Gaussian model is still incapable in using this information to improve the description (indicated by the – in the fourth column). A second Gaussian density has to be modeled around this outlier in order

Table 3.2: The robustness of all methods to outliers in the training data. In the second and third column the outlier is not labeled as outlier, in the fourth and fifth column it is given as a negative example. A + indicates that a method is robust, − that it is not robust and −/+ indicates that it actually depends on the implementation of the method.

method	Unlabeled outliers in training		Labeled outliers in training	
Gaussian model	−	estimation of covariance matrix may suffer, when regularization is used, very large outliers can be rejected	−	outlier density should be modeled using few examples
mixture of Gaussians	−	estimation covariance matrix suffers	−	outlier density should be modeled using few examples
Parzen	+	density estimation only influenced locally	+	outlier density should be modeled.
k-centers	−	ball centers sensitive	−	outlier can obtain a ball, but overlapping balls give problems
NN-distance	−	distance quotient extremely sensitive	−	use a Nearest Neighbor Classifier
SVDD	+	a user defined fraction of the data can be rejected	+	outliers can be forced outside the description
LVQ	−	prototypes will be placed on or near outliers	+	in essence a classifier, it can repel from outliers
k-means	−	prototypes will be placed on or near outliers	−	in essence a clustering, cannot repel from outliers
PCA	−	estimation covariance matrix suffers from outliers	−	outlier density should be modeled using few examples
SOM	−	prototypes will be placed near outliers	+	repel from outliers
auto-encoder	−/+	depends on regularization strength	−/+	a bad representation should be forced
diabolo	−/+	depends on regularization strength	−/+	a bad representation should be forced

to incorporate this outlier in the training. This is possible, of course, but it requires extra assumptions and models, and this surpasses the conventional Gaussian model.

From all the methods, the NN-d is the least robust (most sensitive) to noise. One outlier will cause a large portion of the feature space to be acceptable (see also section 3.4.2). The methods which rely on some variance measure, such as the Gaussian density or the PCA, can also suffer from the outliers. The variance introduced by a remote object

can overwhelm the variance for the target class completely. The methods which find local approximations, such as the Parzen density, the k-means, LVQ and k-centers (assuming that the number of prototypes or means is sufficiently high) will very likely place one cluster around the outlier. For a remote object and a small cluster this will hardly influence the rest of the description. Finally, the methods with a strong regularization character, such as the diabolo networks, the auto-encoders and the support vector data description, find more or less global solutions and are relatively insensitive to outliers.

In some cases examples of outliers are known. Using this extra information might improve performance of the one-class classification, because it gives an indication of the relative importance of the features. When a large number of outlier objects is available, a traditional two-class classifier can be used. When just a few outliers are present, a one-class classifier should be preferred, because a closed boundary around the data is obtained. Therefore, we assume that just a few outlier objects are present in the data.

Most of the one-class classification methods are not capable of training with negative examples. In some cases the method cannot be trained with example outliers at all, such as in the normal density, the k-means clustering or the PCA. In these cases the outlier distribution should be modeled separately, which might be problematic when just a few objects are available. The Parzen density can cope with just a few samples and can incorporate these negative examples.

In other methods, such as the LVQ, auto-encoder and diabolo network and the self-organizing map, the learning rule can be adapted not only to accept the target objects, but also to repel the outlier objects. For the larger neural networks (auto-encoder and diabolo networks), the influence of just a few outlier objects might be too small to find very good performance. The LVQ is essentially a classifier, and it is, therefore, the best method to incorporate a few outlier objects in the training. Finally, in the support vector data description, it is not only possible to actively exclude negative examples, it is also possible to adjust the strength with which negative examples should be rejected (see the bound (2.48)).

3.7 Number of parameters

Here we summarize the discussion about number of free and 'magic' parameters in each of the methods. In table 3.3 the sensitivity to the scaling of the data, the number of free parameters and the user defined parameters for each of the one-class classification methods is listed.

The sensitivity of the methods to scaling indicates how a method can adapt to rescaling of one of the features. Some methods depend heavily on a properly defined distance between objects, such as the NN-d, but also the other boundary methods. Other models can adapt to that, such as the Gaussian model where a complete covariance matrix can be optimized. Because in the mixture of Gaussians only a diagonal covariance matrix is chosen with equal variance in each direction, this method also becomes scale sensitive.

The number of free parameters gives an indication of the flexibility of the method and

Table 3.3: The scaling sensitivity, the number of free parameters and the number of parameters required from the user for each of the one-class classification methods. A '+' in the second column indicates that the method will be sensitive to the scaling of the features. Further we used the conventions: d: dimensionality of the data, N: number of training objects, k: number of clusters, M: dimensionality of subspaces, h: number of hidden units.

method	scaling sensitivity	number of free parameters	number of user defined parameters
Density approaches			
normal density	-	$d + d(d+1)/2$	regularization λ
Mixture of Gaussians	+	$(d+2)N_{MoG}$	N_{MoG}, # iterations
Parzen density	+/-	1	0
Boundary approaches			
k-centers	+	k	k and # iterations
NN-d	+	0	0
SVDD	+	N	f_{SV}^{out}
SVDD-neg	+	N	f_{SV}^{out}
Reconstruction approaches			
k-means	+	kd	k, # iterations
PCA	-	$\binom{d-1}{M}$	fraction of preserved var.
mixture of PCAs	-	$\left(\binom{d-1}{M} + d\right) N_{MPCA} + 1$	dim. and # subspaces
SOM	+	$k^{d_{SOM}} d$	dim. and subspace size
auto-encoder	-	$(2d+1)h_{auto} + d$	# hidden units
diabolo network	-	$h_{diab}(4d + 4h_{diab} + 5) + d$	# hidden units, dim. subspace

the sensitivity to overtraining. Of course the training procedure and extra regularization might decrease the *effective* number of parameters (for instance, in the case of the support vector data description and the neural networks with their strong regularizing character). In other cases, such as the NN-d and the Parzen density, almost no free parameters are present and almost no model is assumed. These methods completely depend on the training set and as a consequence a bad training set (i.e. a training set which does not reflect the true distribution well) can ruin the performance of these methods.

The number of parameters which should be defined by the user indicates how convenient the application of a method is. In table 3.3 we have omitted parameter f_{T+}, because this is required in all methods and it is therefore not a 'magic' parameter for a particular method (although it is explicitly used in the support vector data description). Using a Parzen

density is extremely simple and no poorly optimized parameter can ruin the performance of the method (although in this case the low sample size probably will). In the support vector data description the user has to supply f_{SV}^{out}, the fraction of the training objects which is rejected by the description (i.e. *outside* instead of *on* the boundary). Although this parameter can be very important for the performance of the method, it is an intuitive measure which offers some extra interpretation. The auto-encoder and diabolo networks require several settings which are not intuitive to the user beforehand, such as the number of hidden units and the learning rates. Default values may be supplied, but poor choices might create such a large bias for a specific problem that the methods become useless.

3.8 Discussion

The one-class classification methods can be based upon three main principles: density estimation, direct boundary estimation and reconstruction. The density estimation gives the most complete description of the data, but might require too much data. For lower sample sizes a method which directly estimates the boundary might be preferred. Finally, a distance or a reconstruction error might be defined based on a model of the data. This model gives the ability to include extra prior knowledge of the problem.

To compare the performance of different methods on a training set, the tradeoff between the error of the first and second kind, $\mathcal{E}_{\mathrm{I}}$ and $\mathcal{E}_{\mathrm{II}}$ (or the target acceptance rate f_{T+} and the outlier rejection rate f_{O-}), has to be investigated. To measure these errors, both target and outlier data should be available. When no outlier data is present, an estimate of the covered volume of the method should be used (which then implicitly assumes that outliers are uniformly distributed). For higher dimensional data (larger than 10-dimensional) volumes are very hard to estimate and there outlier data should be available.

The tradeoff between $\mathcal{E}_{\mathrm{I}}$ and $\mathcal{E}_{\mathrm{II}}$ is present in all one-class classification methods. In all methods it is possible to set a threshold θ, such that a prespecified target acceptance rate f_{T+} on the training set is obtained. The outlier rejection rate f_{O-} has to be measured using the example outliers. Good fitting methods will obtain a high value for f_{O-}. To compare different one-class classification methods on a certain training set, this f_{O-} will be integrated over a range f_{T+} (by changing the threshold in the classifier). This integrated error, also called Area under the ROC Curve (AUC), will be used in the performance evaluation of one-class classifiers.

The methods listed in this chapter have different characteristics concerning the number of free parameters, the number of magic parameters, their flexibility to adapt to changes in scale of the features and their robustness against outliers in the training set. Table 3.3 shows that a large variation is present in the number of free and magic parameters for the different one-class classifiers. The number of free parameters should be small to avoid a too flexible model and rapid overfitting to training data (which can be expected in the mixture of PCA's, the SOM or the auto-encoder). The number of user defined parameters should be small and the individual parameters should have an intuitive meaning. This avoids the optimization of a large set of vague parameters by the user. On the other hand, when just

a few magic parameters are present, it often indicates that a strong model of the data is assumed. The Parzen density and the NN-d contain the lowest number of free parameters.

The robustness of the one-class classifiers against outliers in the training set differ from method to method (as shown in table 3.2). We discerned outliers which are labeled as outliers (thus in fact a two-class classification problem is obtained) and outliers which can only be discerned from genuine target objects by their low resemblance to the bulk of the data. Some robustness against outliers is already incorporated in all the one-class classifiers, by the fact that the threshold θ is optimized for a specific $f_{T+} < 1$. For instance, when this threshold is optimized for $f_{T+} = 0.95$, the 5% most dissimilar target data (probably containing the outliers) are rejected. Just a few one-class classifiers are able to use labeled outliers in the training. These are the Parzen density, the SVDD, the LVQ, the SOM and the neural networks.

In the next chapter we will investigate how the methods presented in this chapter work on some artificial and real-world applications.

4. EXPERIMENTS

In this chapter the one-class classification methods introduced in the previous chapter are applied to several artificial datasets. By using artificial datasets, the characteristics of the datasets can exactly be set. The datasets are constructed such that they differ in scaling, clustering, convexity and their placing in subspaces. In section 4.1 the general settings of the one-class classifiers will be discussed. The types of artificial data used are discussed in section 4.2. In section 4.3 the characteristics of the error measure (defined in section 3.1.2) will be shown. In sections 4.4 to 4.11 the influence of sample size, modality, scaling, robustness and time consumption on the one-class classification methods will be investigated. Finally, in section 4.12 we will look at the performance of the one-class classifiers on real world datasets.

4.1 Parameter settings

All (one-class) classifier methods contain two types of parameters. The first type of parameters is optimized by the optimization routine of the methods themselves during the minimization of an error function $\mathcal{E}$. These parameters include, for instance, the placing of the prototypes in the k-means method or the rotation of the subspace in PCA. Because these parameters are optimized automatically, they are of no further worry for the user. The second type of parameters are the, so called, 'magic parameters' which have to be set beforehand by the user. These are, for instance, the number of prototypes k in k-means clustering or the dimensionality M of the subspace in the PCA.

In contrast to the first type of parameters, the values of the magic parameters should be chosen by the user, and poor choices can potentially have a large influence on the performance of the methods. Ideally, these values should be tuned to the data. For instance, when insufficient prototypes are used in k-means clustering, it cannot describe the target data well and it will approximate the data very roughly, resulting in poor classification performances. On the other hand, using too many prototypes in k-means is less harmful, but might still result in some overfitting. When no prior knowledge is available (for k-means, knowing the number of clusters in the data would be very useful), the training set should be used. Given that, in general, just a limited amount of data is available, it might not be trivial to find a good value with high confidence.

In the coming sections we will investigate the behavior of the one-class methods on several artificial datasets. Because we know the characteristics of these artificial datasets (we know that the data have limited complexity; it will contain 3 clusters at most, while

the intrinsic dimensionality ranges from just 2 to 10), we will avoid an exhaustive experimentation concerning the free parameters and we will choose them to have fixed values. We will use reasonable, but not fully tuned, settings for all of the methods. This procedure then gives just an indication of how flexibly a method can adapt to the data for non-perfect settings of the magic parameters, which then provides an suggestion what an average user (without detailed knowledge about the methods and the data) can expect from the one-class models.

For the experiments in this chapter, we set the magic parameters of the one-class classifiers at these predefined values: the number of clusters in the k-means, LVQ and k-center methods will always be set to 10, which is enough to find 2 or 3 clusters in these experiments. The dimensionality of the self-organizing map is set to 2. Because the SOM uses $k = 10$ neurons for each dimension in the map, in total 10×10 neurons are required. The diabolo and the auto-encoder networks will (obviously) have the same number of output units as input units. The auto-encoder will have 10 hidden units, the diabolo network has two hidden neurons in its bottleneck layer (which then describes a 2-dimensional subspace).

Although it would be possible for the PCA to just take a 2-dimensional subspace, increasing the dimensionality is not expensive (from a storage and computational point of view). For the self-organizing map, on the other hand, the number of neurons is exponential in the map dimensionality; k^d to be precise. To take advantage of the relative ease with which extra dimensions can be added to the PCA, we will optimize it such that it will account for 90% of the variance. However, for the real world data we will be more careful in selecting these numbers.

In the comparison between the one-class classification methods both the f_{T+} and f_{O-} are measured (on independent test sets). The f_{T+} is varied from 30% acceptance up to 99%. Because this rate is optimized on the training set, the performance on the test set might significantly deviate from these values. For the integration error over the ROC curve (formula (3.7)) a more limited integration range from 50% to 95% is used. All experiments on artificial data are averaged over 25 runs.

4.2 The datasets

In the first experiments we compare all one-class classification methods on 5 different artificial datasets. These datasets are constructed to investigate different characteristics of the methods. By using artificial data instead of real world data, we avoid focusing on unsuspected and complex distributions. It gives the opportunity to just focus on some important aspects of data distributions. It is expected that the various one-class classification methods will behave differently on data with different characteristics. In the coming subsection we will present a set of datasets which show very specific features, such as scaling, clustering and the (non-)convexity of the dataset, which can illustrate the advantages and disadvantages of the one-class methods. In a special subsection we will discuss a dataset which has an atypical training set (i.e. a training set which is not a representative sample

from the 'true' target set).

After the one-class methods are applied to the artificial datasets, they will be applied to real world data to check whether it is possible to match the characteristics found in the real world data with those in the artificial data. Since real world data tend to be distributed in non-linear subspaces and in clusters of different sizes, or have different non-convex distributions, the characteristics are harder to extract.

4.2.1 Artificial outlier distributions

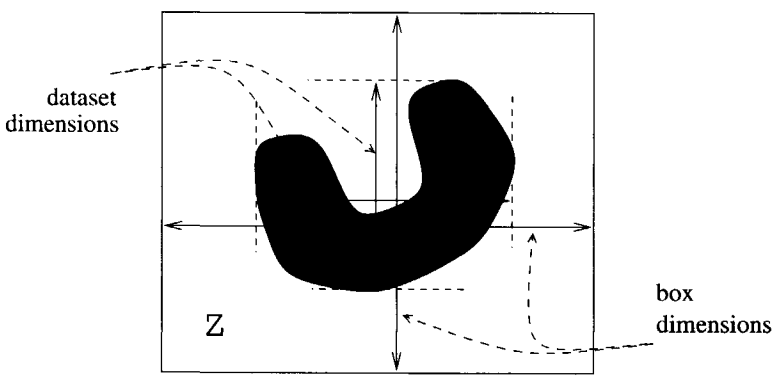


Fig. 4.1: The 'box-procedure' for constructing a box around the target distribution, from which artificial outliers are uniformly drawn. The sides of the box are 1.5 times the size of the training set (the gray area) along the feature axes.

To obtain a set of artificial outlier objects $\mathcal{Z}$, a box around the target objects is defined and from this box test outlier objects are uniformly drawn. The size of the box is determined by the training data itself. In each feature direction the minimum and maximum feature values of objects from the dataset is measured. The center of the box is the average of the minimum and maximum, the sides of this box are 1.5 times the difference of this minimum and maximum. An illustration is given in figure 4.1. Although the dimensions of the box become sensitive to outliers in the target set, this procedure guarantees that all target objects are within the box. Furthermore, by using the box-shaped uniform distribution the outlier objects are very cheap to create.

When the artificial outliers are uniformly distributed, the fraction of accepted outliers by a one-class method, f_{O+}, gives an estimate of the volume covered by this method. This implies that this volume will change when another dataset is considered. Thus, the performance on the outliers will change and therefore the error of the method as well. Assume, for example, that we have a 2-dimensional and a 10-dimensional Gaussian target distribution. In both cases artificial outliers are created using the 'box-procedure' mentioned above. Unfortunately, in the 10-dimensional feature space the ratio between

the volume of the box and the spherical Gaussian boundary will be much larger than in the 2-dimensional case (see section 3.1.1). In other words, the overlap between the outlier and target objects will be larger in the 2-dimensional feature space. It is, therefore, easier to distinguish target objects from outlier objects in high dimensional spaces in comparison with low dimensional.[1]

4.2.2 Well-sampled target distributions

The first collection of artificial data we discuss, contains datasets in which the training data is a representative sample from the 'true' distribution. In these cases, it is expected that for sufficient sample sizes the density methods will perform best. In particular, when the density models fit the data (when they are flexible enough to adapt to the data; for instance, the Parzen density estimator), it is very hard to improve over that. In these cases, the boundary one-class classification methods have to focus on the (relative) low density areas of the data, and might, therefore, show poorer extrapolation characteristics than the density methods. For the reconstruction methods the performance will depend on how well the model fits to the data.

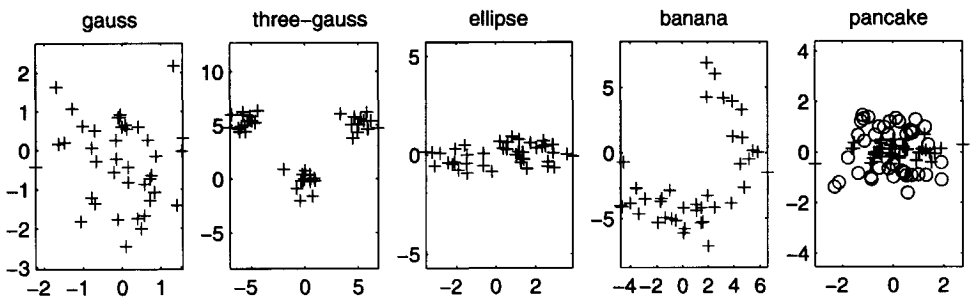


Fig. 4.2: Two dimensional plots of all artificial data. Target data are indicated by '+'; in the pancake data the outlier data are also given by 'o'. For other datasets outliers are uniformly distributed around the target data.

Two-dimensional scatter plots of the different artificial data sets are shown in figure 4.2. The target objects are indicated by '+' markers. The outlier data is created using the box-procedure from the previous section. Only for the **pancake** dataset the outlier data has a non-uniform distribution. This is explicitly shown in the scatter plot by 'o'. This pancake data is explained in more detail in appendix B.3. For all other methods the outlier data is uniformly drawn from a box around the target data. For all experiments 1000 outlier objects are used.

We focus on the following data characteristics:

[1] Note that we did not consider sample sizes. In high dimensional feature spaces larger number of training objects are required to obtain a good estimate of the boundary of the data. When we have just a limited amount of data, a lower dimensionality is to be preferred.

sample size: Because outlier objects can be anywhere in the feature space (by assumption), in any direction around the target data a boundary should be defined. When no strong model is assumed, the required number of training objects increases dramatically with the dimensionality of the data. To investigate the sample size behavior with respect to the dimensionality, a simple normal target distribution with unit covariance matrix around the origin is used (`Gauss`).

The optimal model for this problem is, of course, the Gaussian (or normal) model. Because a unit covariance matrix is used, the individual features are weighted equally. We also assume equally weighted features in most of the other methods (the mixture of Gaussians, the Parzen density, LVQ, k-means, k-centers) and, therefore, we can expect good performance for these methods. The only restriction will be the efficiency of these methods in small sample size situations. The boundary methods will very likely fail. These methods have to infer a very symmetric boundary around the data from a few objects on the boundary. Unfortunately, for the `Gauss` dataset the objects on the boundary are more or less the outlier objects and these give a bad indication of the position of the center (or the high density area) of the dataset.

scaling: To define a resemblance between a test object $\mathbf{z}$ and a target set $\mathcal{X}^{tr}$, some methods heavily depend on a proper definition of a distance and well-scaled features. This holds most explicitly for the nearest neighbor method NN-d, but also for the k-means and Parzen density estimator. An elongated normal target distribution is used to investigate the sensitivity rescaled features (`ellipse`). The standard deviation in the first feature is 4 times that of the other features. All other standard deviations are set to one.

Again the Gaussian model should perform well here. The other methods, which assume well-scaled features, will suffer, especially in case of small sample sizes. When the sample size is increased, most methods will be able to recover from this scaling. As in the `Gauss`, the `ellipse` does not have sharp boundaries, and, as a result, the boundary methods will show poor extrapolation performance. In the group of the reconstruction methods the PCA is expected to be able to use this elongated direction to fit a subspace.

clustering (length of 'data boundary'): The sensitivity to multi-modality and to the relative size of the boundary can be investigated by using clustered target data. The `Gauss` dataset is extended with two other Gaussian distributions with the same unit covariance matrix but with different mean vectors. One cluster has value 10.0 in each feature direction while the other has $+10$ or -10 for successive features (`three-gauss`) (as is shown in the second scatterplot in figure 4.2).

The best methods are the methods which explicitly model several clusters. These are the mixture of Gaussians, LVQ, the k-means and k-centers. Although the Parzen density is no clustering method, it is flexible enough to model clustered data. It can be expected that in the `three-gauss` the models suffer from the small sample sizes.

Each cluster now contains just one third of the data in comparison to the **Gauss**. In particular, for the support vector data description this is expected to cause problems.

(non-)convexity: The influence of both the convexity and the boundary size is investigated by a banana-shaped dataset, **banana**. Data is uniformly distributed in a half circle; the radial distribution is normally distributed. The fourth subplot in figure 4.2 shows the 2-dimensional distribution. In higher dimensions this distribution function is repeated and the feature values are drawn independently from the original 2-dimensional distribution.

For this **banana** dataset the one-class models have to be more complex than for the previous datasets. Unimodal models will not work here (such as the Gaussian model or the PCA), but due to the complexity of the boundary, density methods should also have problems finding the structure in the data. In small sample size situations, the non-convexity of the data is completely hidden. For large sample sizes it is expected that the flexible methods such as the Parzen density estimator, the support vector data description and the self-organizing map will perform well.

subspaces: For poorly scaled data or for data in a subspace, singular directions can appear in the data. Although for the data in the complete feature space the sample size might be extremely small, when the data is only considered in the data subspace, the sample size might be sufficient to obtain reasonable data descriptions.

To simulate a target set in a subspace, the **pancake** dataset is constructed. The Matlab code for creating such datasets can be found in appendix B.3. The target objects are normally distributed with unit variance in the first $\lfloor d/2 \rfloor$ features and a variance of 0.1 in the remaining $\lceil d/2 \rceil$ features. In this dataset, the outliers are not created randomly in a box around the data, but they are explicitly created near the subspace. Two outlier clusters with the same variance structure as the target data are created. The means of these clusters are set to $+1$ and -1 in the directions of low variance. This **pancake** dataset is the only artificial data set where the outliers are not uniformly distributed.

For this dataset the PCA and SOM should work best. When we assume that the data is in an n-dimensional linear subspace, this subspace can be described by just n objects.[2] A density method will hopelessly fail here. The other boundary methods (the nearest neighbor NN-d and SVDD) do not assume a subspace model thus only focus on distances between objects, and, therefore, it is not to be expected that they will perform very well in this case.

[2] Or $n + 1$ if we assume that the data do not have a zero mean and the mean of the data has to be estimated as well.

4.2.3 Atypical training data

In the previous subsection we focused on problems for which the training data is a representative sample from the 'true' target distribution. We have argued in the introduction that this is not usually the case. The training and testing data are not i.i.d. (not identically and independently distributed). The user has to measure and create a training set beforehand and in many practical applications the target distribution might be ill-defined or completely unknown. We have seen this in the pump characterization problem in chapter 2, page 49.

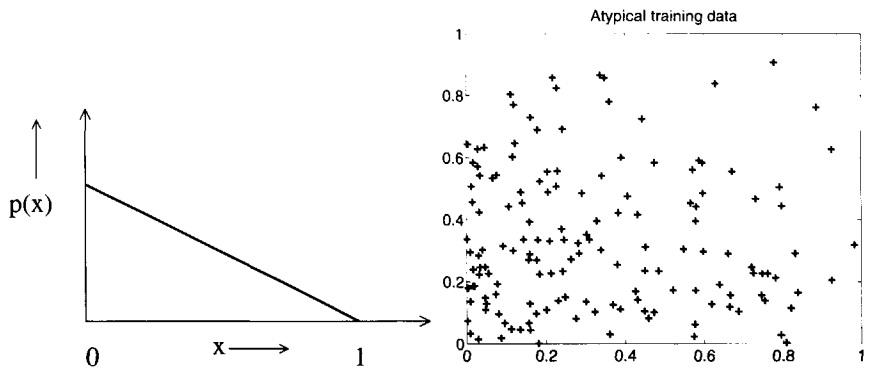


Fig. 4.3: Training distribution of the target class of the `atypical` dataset.

To emulate atypical sampling behavior, we assume that the 'true' target distribution is uniformly distributed in a unit square. The training data is sampled from a triangular distribution between 0-1 (the 1-dimensional version is shown in the left graph in figure 4.3). All feature values are drawn independently from this 1-dimensional distribution. In the left plot of figure 4.3 a scatter plot of a 2-dimensional training set is given. The target data of the test set will be uniformly distributed in the square box. This will be called the `atypical` dataset. Outlier objects are drawn in the same manner as in the previous artificial datasets, from a box around the target set.

4.3 The error for one-class classifiers

In section 3.1.2 (page 60) we defined an error $\mathcal{E}_{\mathrm{M}}$ to evaluate one-class classifiers. It integrates the f_{O+} for varying fractions of f_{T-} (or equivalently, it integrates the error $\mathcal{E}_{\mathrm{II}}$ over varying $\mathcal{E}_{\mathrm{I}}$):

$$\mathcal{E}_{\mathrm{M}} = \int_0^1 \mathcal{E}_{\mathrm{II}}(\mathcal{E}_{\mathrm{I}}) d\mathcal{E}_{\mathrm{I}} = \int_0^1 \int_{\mathcal{Z}} I\left(p(\mathbf{z}) \geq \theta_{f_{T+}}\right) d\mathbf{z} d\theta_{f_{T+}} \tag{3.7}$$

(where $\theta_{f_{T+}}$ is the threshold on the probability model $p(\mathbf{z})$ on the data). This tries to avoid the focus on one of the threshold values by averaging over a range of thresholds. First we want to check if this error measure gives information comparable to the complete ROC curves.

To show how this error behaves and how it relates to the original ROC curve, we apply it to a simple 2-dimensional Gaussian target distribution. All methods are trained on a sample of 25 objects drawn from the target distribution. In the left plot of figure 4.4 the results of all one-class methods are plotted in one ROC plot.

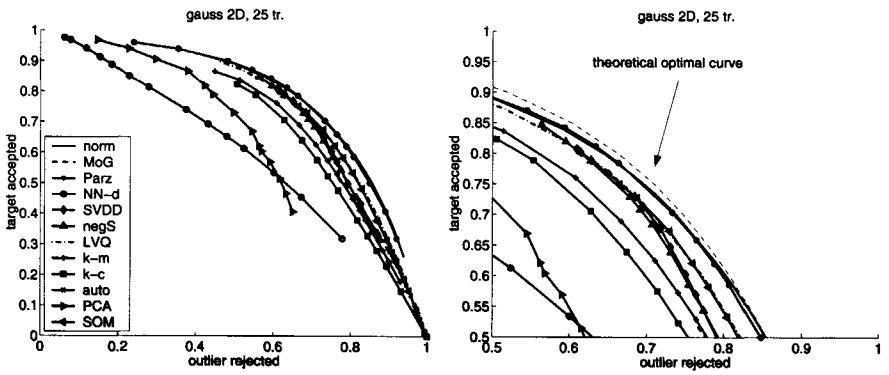


Fig. 4.4: Receiver-Operator Characteristic curve for different one-class classifiers on 2-dimensional Gaussian data with 25 training objects and 1000 test objects. The complete ROC curve is shown on the left, an enlarged version with the theoretically optimal curve on the right.

These ROC curves can serve to compare the performances of the one-class classification methods. The more the ROC curves are located in the upper right corner, the better the target and outlier objects are separated. It appears that most methods perform about the same on the Gaussian distribution. Due to the large number of methods, they can hardly be discerned in the plot (so don't try, we will discuss only the two curves in the lower left in the next section). Furthermore, it is not clear how these 2-dimensional curves can be compared. The use of the integrated error, formula (3.7), will solve this problem in the coming section.

Two ROC curves are located below all other ROC curves, i.e. the curves with the open circles and with the triangles pointing to the right. These are the ROC curves of the NN-d and the PCA methods. The NN-d is very noise sensitive (see section 3.4.2) and it suffers from a few objects that are relatively remote from the bulk of the data. It accepts large areas in feature space and on average, rejects just about half of the outlier objects. The PCA concludes that all feature directions show large variance, and thus retains the complete feature space. Although for all target and outlier data the error is very small (on average the reconstruction error for both target and outlier data is $\rho_{PCA}(\mathbf{x}) \sim 10^{-30}$), objects located around the mean of the dataset are represented slightly better than objects

lying further way. By this tiny difference, still some separation between target and outlier objects is possible.

What is not visible in these plots (due to the overlap of the curves of different methods or because some curves are collapsed onto one point in the ROC plot) is that some of the methods do not perform well either. For instance, the auto-encoder network suffers from the fact that the number of training objects is not very large and thereby it cannot be trained well. It accepts the complete feature space and thus its complete ROC curve is in the lower right corner ($f_{O-} = 1$ and $f_{T+} = 0$). Other methods, such as the k-center method, the SVDD and the SOM perform poorly when a large target acceptance rate is required. It appears that their ROC curves stop at about 80% acceptance rate. When an error integration rate up to 95% is considered, this will cause a large error.

In this artificial data distribution the probability densities of the target and outlier data are known and it is, therefore, possible to compute the optimal boundary for different target rejection rates.[3] For objects coming from a Gaussian target distribution (with known mean and covariance), the Mahanalobis distances are χ^2_d distributed and the optimal threshold can therefore be determined; see formula (3.12). In the right subplot in figure 4.4 the theoretical optimal ROC curve is shown. Most methods approach this optimal curve very closely.

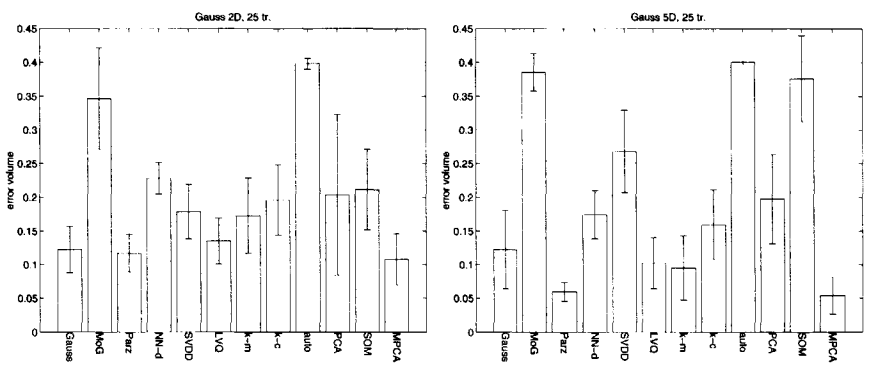


Fig. 4.5: Histograms of the integrated error over the ROC curves for the (left) 2-dimensional and (right) 5-dimensional `Gauss` dataset. The errorbars give the standard deviations over 25 runs.

A histogram of these integrated error areas, as defined in section 3.1.2 is given in figure 4.5. The results for 2-dimensional data are shown in the left plot, for the 5-dimensional data in the right plot. The target data is normally distributed. The worst case error is $\mathcal{E}_{\text{worst}} = 0.45$ for an integration range of 0.5 to 0.95.

[3] Note that the outlier distribution is not completely known when the box procedure from section 4.2.1 is applied. The size of the box is determined by the most remote objects from the random target sample. To avoid the influence of these objects, for the experiment with 25 target objects, the box is fixed between -4.0 and 4.0 in all dimensions.

The histograms show that the Parzen density estimation and the mixture of PCA's perform very well. The results are comparable to and sometimes even better than the results of the fitted Gaussian method. The data is generated from a Gaussian distribution with unit variance in all directions and we can, therefore, expect that the Gaussian model would fit the data best. The Gaussian model, however, has to estimate a complete covariance matrix. This can become problematic in small sample size situations. In this case the Parzen density has the advantage that the width parameter is equal for all directions in the feature space. The mixture of PCA's takes advantage of the fact that a very large regularization parameter has to be used to compute the inverse covariance matrices (matrix C in formula (3.36)). Without this regularization parameter, the mixture cannot be trained at all, but when it is set, an almost spherical description is obtained.

The mixture of Gaussians, nearest neighbor method (NN-d) and the PCA perform very poorly, which was already observed from the ROC curves in figure 4.4. The performance of the auto-encoder network is also poor. This is caused by the fact that the network could not accept more than 80% of the target class. For the larger target acceptance rates it therefore shows very large errors. But when a reasonable integration range is chosen, the integrated error gives an indication comparable to the ROC curves.

In the 2 plots in figure 4.5 the difference in performance is clearly noticeable. In both the left and right problem the target data is normally distributed, but the dimensionality differs. Data is 2-dimensional in the left and 5-dimensional in the right subplot. With increasing dimensionality the volume of the bounding box of the outlier data and the volume of the target distribution diverge (we already mentioned this in sections 3.1.1 and 4.2.1). This means that the volume covered by the target class within the outlier distribution becomes very small and, consequently, the error decreases. It appears that the theoretical optimal performance for 25 objects in 2 dimensions is $\mathcal{E}_{\text{theor}} = 0.105$, while for 5-dimensional data the error decreases to $\mathcal{E}_{\text{theor}} = 0.045$. This also means that for different dimensionalities the problems basically change and therefore cannot be compared. When just one fixed dimensionality is considered, it is, of course, still possible to compare the performances of the different one-class methods.

4.4 Sample size

To investigate the influence of the sample size, the **Gauss** data is drawn from a Gaussian distribution. For three different dimensionalities (2-, 5- and 10-dimensional) samples with different sizes (10, 25, 50, 75 and 100 training objects) are created. In the left plot of figure 4.6 the results of the fitted Gaussian (or normal) method are shown. For increasing sample sizes the error decreases, but for increasing dimensionalities the performance for small sample sizes decreases. This is mainly caused by problems in the estimation and in the inverse operation of the covariance matrix on which the method heavily relies. For small samples sizes the matrix becomes singular and can not be inverted. In these cases, the description by the normal method mainly relies on the regularization used in the covariance matrix (see section 3.3.1). For 10 objects in 10 dimensions the regularization parameter λ

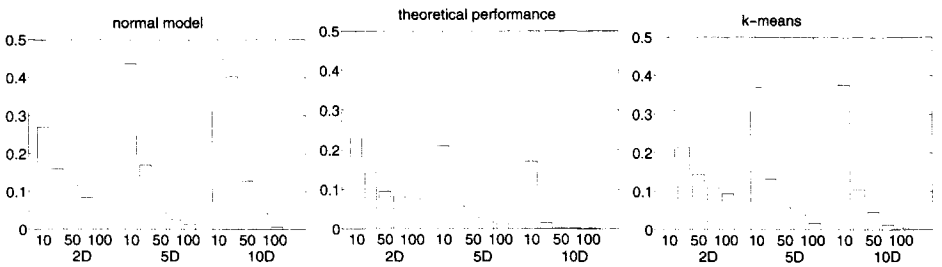


Fig. 4.6: The integrated error for the `Gauss` data of different dimensionalities (2, 5 and 10) and sample sizes (10, 25, 50, 75 and 100 training objects) for the Gaussian model (left) and k-means model (right). As a reference the performance of the true target data distribution (with the correct mean and covariances) is shown in the middle.

is not optimized and the worst case error is approached. This effect is stronger for larger dimensionalities since the required number of objects exponentially increases. For larger sample sizes, reasonable performances are achieved.

When we compare these performances with the theoretically possible performance, shown in the middle plot, we see that a large error is still present (especially for small sample sizes). When the k-means method is applied, it is implicitly assumed that a stronger spherical description is suitable and the problem of the estimation of the covariance matrix is avoided.[4] The performance is better, but now it appears that the estimation of the (empirical) threshold is problematic for small sample sizes. It appears that when 100% of the training data is accepted, on average about 60% of the testing target data is also accepted. This severely increases the error, see formula (3.8). For larger sample sizes, the difference between the normal method and the theoretical model decreases. For 100 objects in two dimensions both methods approach the theoretical performance.

In figure 4.7 the performances of all one-class methods are shown on `Gauss` data. In the upper left corner the dataset contains 10 2-dimensional objects and in the lower right corner 100 10-dimensional objects. In each individual plot the performance of the 12 methods is shown in the following order: the Gaussian method, mixture of Gaussians, Parzen density, NN-d, the SVDD, LVQ and k-means, k-centers, auto-encoder network, the PCA, the self-organizing map and the mixture of PCA's. The diabolo networks is left out of the plots. The number of free parameters was too large for this small sample size data and in all situations poor results are obtained.

These plots show that the performances on a training set consisting of 10 objects are often poor. In particular, the mixture of Gaussians, the SVDD, the k-center method and the SOM do not work at all. Clearly, the sample size of 10 objects in 2 dimensions is insufficient in all these cases. Surprisingly, the Parzen density still works best. Of course, we have to mention that the target distribution has equally scaled feature values,

[4] This is an example of the bias-variance dilemma: we include extra bias in our method to reduce the variance.

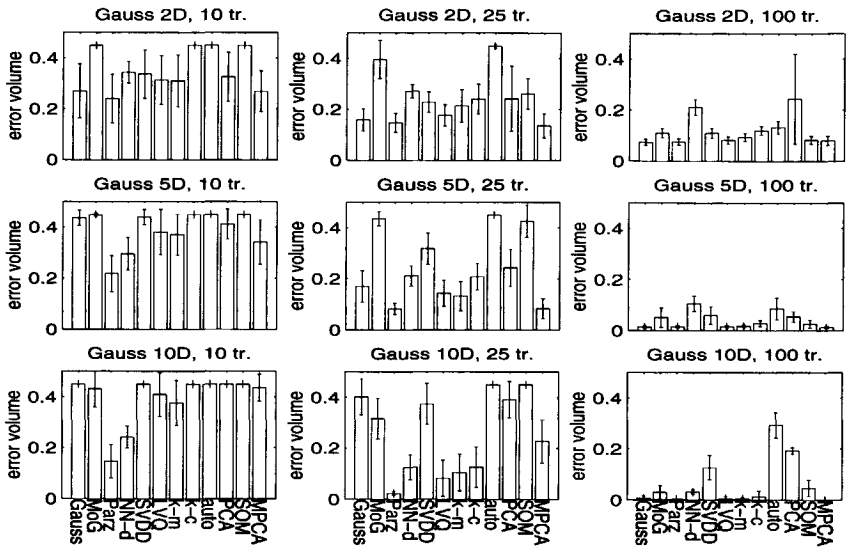


Fig. 4.7: Performance of one-class classification on the Gauss dataset.

and therefore, that the assumptions of the Parzen density estimation are fulfilled (see section 3.3.3, page 67). The Parzen estimator only has to estimate one single smoothing parameter h for all feature directions. Because the assumptions are fulfilled and just one free parameter has to be estimated, the Parzen fits very well with this data.

By increasing the sample size, the performance of all methods improves. This can already be observed for 25 training objects, except for the auto-encoder which requires the largest sample sizes; even on 25 training objects it does not train properly. There are small errors for almost all methods for sample sizes of 100 objects. For 2-dimensional data only the PCA accepts the complete feature space. In 5 dimensions, even the PCA finds a good description (although it retains 5 features), the numerical imprecision in the computation of the reconstruction error still cause some good performance, see section 4.3.

With increasing dimensionality, the errors slightly decrease. This is caused by the fact that the volumes of the target and outlier sets exponentially diverge in these cases (see section 3.1.1). Due to the relatively small target volume, the inherent overlap between the target and outlier class decreases. With increasing dimensionality the NN-d method improves most, while the SVDD suffers from the lower bound on the number of support objects required for its description (when a SVDD is trained on 10 training objects and 4 objects become support vector, the estimation of the error on the target set, formula (2.47), then indicates that the expected error on the target set will be about 40%. This will result in a large error contribution in the integrated ROC error, formula (3.8) for high values of f_{T+}.). For higher dimensional data sets this number can become substantial, especially when smaller sample sizes are considered. The PCA does not fit the data at

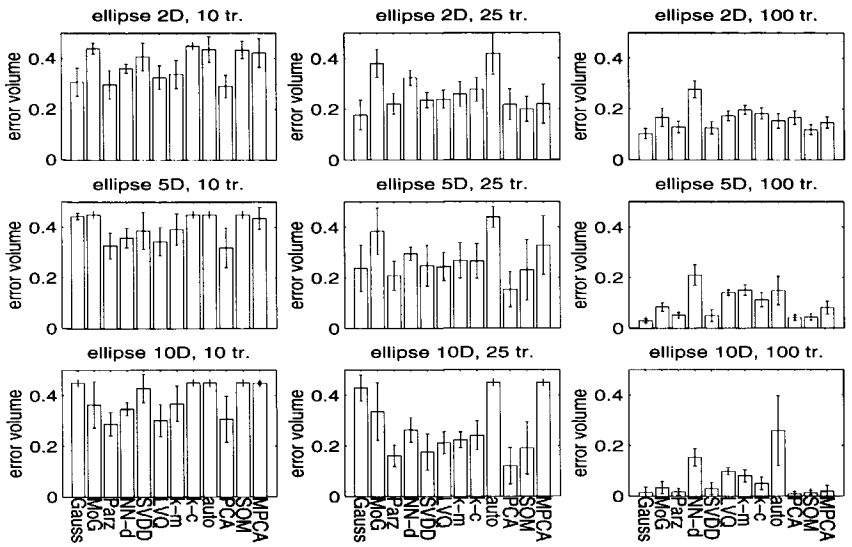


Fig. 4.8: Performances on the `ellipse` data distribution for varying sample sizes and dimensionalities.

all: there is no subspace present and therefore PCA cannot distinguish between target and outlier objects. In this data, with one cluster, the k-centers, k-means and the LVQ show good performance, comparable to the best Parzen density estimation.

4.5 Scaling

In the next dataset, the `ellipse` dataset, we investigate the influence of the scaling of the individual features. The errors on the `ellipse` dataset are shown in figure 4.8. Comparable results, especially for small sample sizes when 10 training objects are used, can be observed. The main difference in performance between the `Gauss` and the `ellipse` datasets can be observed in the PCA. Especially for small sample sizes in high dimensional feature spaces, 10 or 25 training objects in 10 dimensions, the PCA can find a reasonable data description. But also for large sample sizes, the PCA shows much smaller error.

The Parzen method works still well, although it does suffer from poorly scaled features in this experiment. The performance is somewhat worse than in the `Gauss` experiment (figure 4.7). The NN-d also deteriorates, although not as much as the Parzen density estimator. The SVDD performs better in comparison with the `Gauss` data and achieves performance comparable to the Parzen and the PCA. The elongated `ellipse` data makes it possible for the SVDD to use just a few support vectors to describe the complete dataset. The weak performance of the SVDD in the low target rejection rates is, therefore, avoided

and the total performance increases (this is clearly visible in the experiments using 100 training objects). The SOM also suffers from the scaling, especially when it is trained on about 25 training objects. When a large number of samples is available, however, the performance does not deteriorate.

The clustering methods all suffer from the poorly scaled data. In the small sample size situation (10 training objects) performance was already very poor. In the larger sample size situations (25 and 100 training objects), it is visible that the k-means, k-centers and LVQ deteriorate more than the other one-class classifiers. Good performance can be observed in the Gaussian model. The Gaussian model has rescaled its covariance matrix and the performance is about equal to that in the `Gauss` dataset. The auto-encoder again require large sample sizes for training. It shows very poor performance, except for 100 training objects in 2-dimensional data. Finally, the mixture of Gaussians relies on the estimation of a diagonal covariance matrix for each of the components. This makes the method sensitive to the scaling and its performance drops with respect to the original `Gauss` data.

4.6 Multimodality

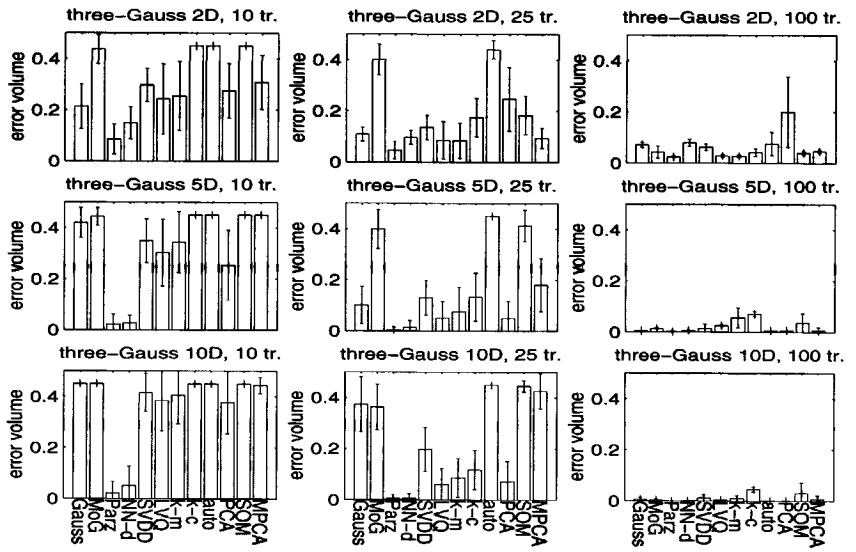


Fig. 4.9: Performances on the `three-gauss` data distribution for varying sample sizes and dimensionality.

To investigate the influence of the number of clusters in the data and the length of the boundary, we apply the one-class classifiers to the `three-gauss` (see figure 4.2). In figure 4.9 the error on the `three-gauss` dataset is shown for all methods and all sample

sizes considered. The characteristics of the `three-gauss` dataset differ significantly from the previous `Gauss`. Since the `three-gauss` data consist of three Gaussian clusters with a distance of $10\sqrt{d}$ between the centers, the uniform outlier distribution will cover a large volume. The fraction which will be occupied by the target data will be relatively small in comparison with the first `Gauss` distribution (compare the two plots in figure 4.2, page 88). The best performances on the `three-gauss` data is therefore better than the best performances of the `Gauss` data (this is especially clear in the experiment with a sample size of 10 and 25 objects).

Some similarities and dissimilarities between the results of the `Gauss` and the `three-gauss` datasets can be observed. Most methods which perform poorly on the `Gauss` data also perform poorly on the `three-gauss` data. A sample size of 10 objects gives results comparable to that of the `Gauss`, indicating that the clustering characteristics of the data is completely hidden by the noise in the small sample size. The clustering of the data only becomes apparent in the large sample size situation. The performance of the NN-d is improved with respect to the `Gauss` due to the increased complexity of the data and therefore the decreased 'effective' sample size. For 100 training objects in 2 dimensions the performance of the (unimodal) Gaussian method deteriorates with respect to the other methods. The mixture of Gaussians, on the other hand, performs well in this case.

For 2-dimensional data, the PCA still preserves the complete feature space. For larger dimensionalities the performance increases. The three centers of the Gaussians span a 2-dimensional subspace which is easily described by the PCA in 5 and 10 dimensions. The performance becomes comparable to that of the Gaussian method. The Parzen method still works very well, also for very small sample sizes (10 objects in the training set). For the moderated sample sizes, the SOM has problems to describe the data. The LVQ and k-means show very large variance for small sample sizes (10 training objects). The prototypes sometimes miss a cluster due to bad sampling of the target data, which results in poor performances. On the other hand, when each of the three clusters obtains a prototype, good results are obtained.

Finally, the SVDD still clearly suffers from the fact that a certain number of vectors is required to define the boundary (see inequality (2.47)): for higher dimensionalities, performance seriously decreases. Although the boundary of the data is enlarged, the volume covered by the target set is decreased (in comparison with the outlier distribution) and the average performance on the `three-gauss` data is comparable to that of the `Gauss` data.

In figure 4.10 the same methods are applied but now with different user settings. In the previous experiments the number of clusters and the sizes of the networks were chosen to be large enough to cover all the data (for the exact parameter values, see page 85). In this experiment we change the number of clusters in the mixture of Gaussians, the LVQ, k-means clustering and the k-centers to 2, and the number of hidden units in the auto-encoder to 2. All other methods stay the same. Because the number of clusters in the `three-gauss` is 3, this mismatch introduces a big bias for these methods.

It is expected that, in general, the performance of these methods will deteriorate. Only the mixture of Gaussians and the auto-encoder network should improve their performance. This is caused by the fact that the number of parameters which have to be estimated

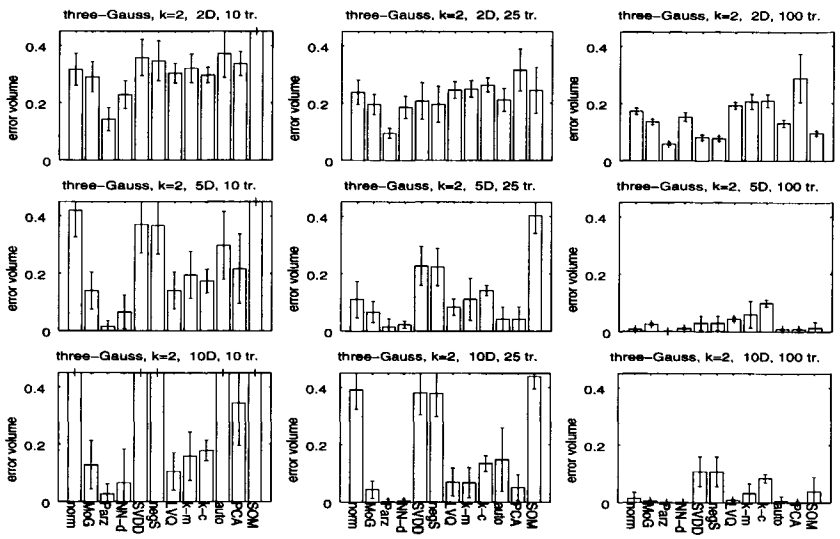


Fig. 4.10: Performances on the **three-gauss** data distribution of the one-class methods which only model two clusters.

significantly decreases in both methods. These fewer parameters can now be estimated with higher precision, which results in a much better description. The performance of the k-means, LVQ and k-centers drastically deteriorates, which becomes especially apparent in the case of a sample size of 100 objects. For smaller sample sizes, not enough data is available to conclude with confidence that three separate clusters are present in the data. Using just two clusters now gives comparable results.

4.7 Non-convexity

Figure 4.11 presents the results for the **banana** dataset. This dataset has a non-convex distribution for the target class. Looking at the scatter plots (page 88) it is obvious that only for larger sample sizes the banana-shape can be distinguished from the **Gauss** data and only then the non-convexity of the data becomes apparent. For smaller sample sizes the **banana** data set just resembles the original **Gauss**. Therefore, we can observe that the performances for the small sample sizes are almost identical to the results on the **Gauss** data (compare figures 4.11 and 4.7). Only for the NN-d method does the performance increase somewhat more with the increase of increasing dimensionality. Most other methods completely fail for small sample sizes (10 training objects).

For larger sample sizes we detect some more differences between the one-class classification methods. The SVDD now also shows poor performance in moderate sample sizes

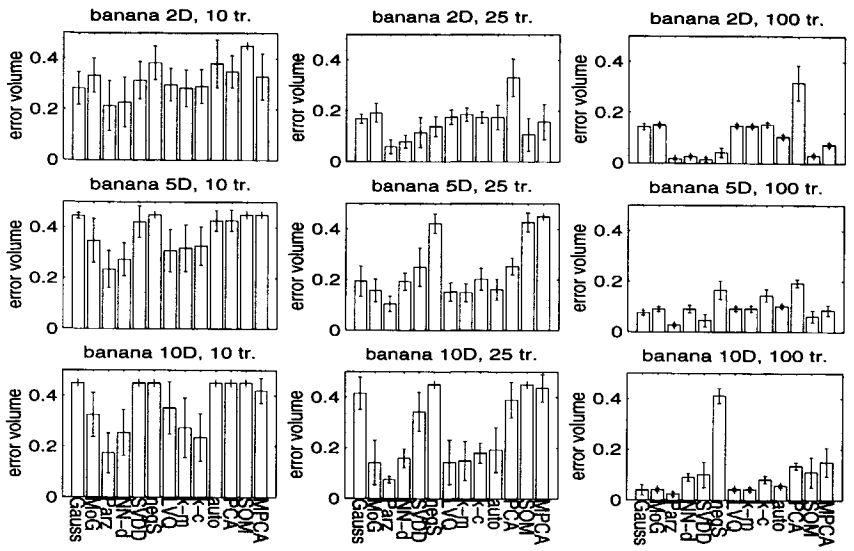


Fig. 4.11: Performances of the one-class classifiers on the banana data distribution for different sample sizes and dimensionalities.

in high dimensional feature spaces in comparison with the Gauss dataset. This is caused by the enlarged boundary of the data. For large sample sizes the SVDD behaves well. For 100 training objects in a 2-dimensional feature space, the SVDD and the Parzen density estimator reach the best accuracy over all other methods. Surprisingly, the normal model works reasonably well, even in the large sample sizes (except perhaps for 100 objects in 2D). The clustering methods, like k-centers, k-means, LVQ and SOM, reach comparable accuracy as in the Gauss dataset. Like the SVDD, the auto-encoder works well when enough data is available, but fails in the small sample size situations. Finally, the PCA fails because no clear linear subspace in the data is present.

The non-convexity of the data seems to be very hard to detect. For small sample size (in high dimensional data) only the crude form of the data can be described. When a large sample is available, just a very small penalty is paid when the non-convexity is not modeled. Even for large sample size data the normal model performs comparably to the other methods.

4.8 Subspace

To investigate which one-class classifiers are suitable for describing a dataset which is distributed in a subspace, we apply the classifiers to the pancake dataset. This is the only artificial dataset in which the outliers are not uniformly distributed around the target

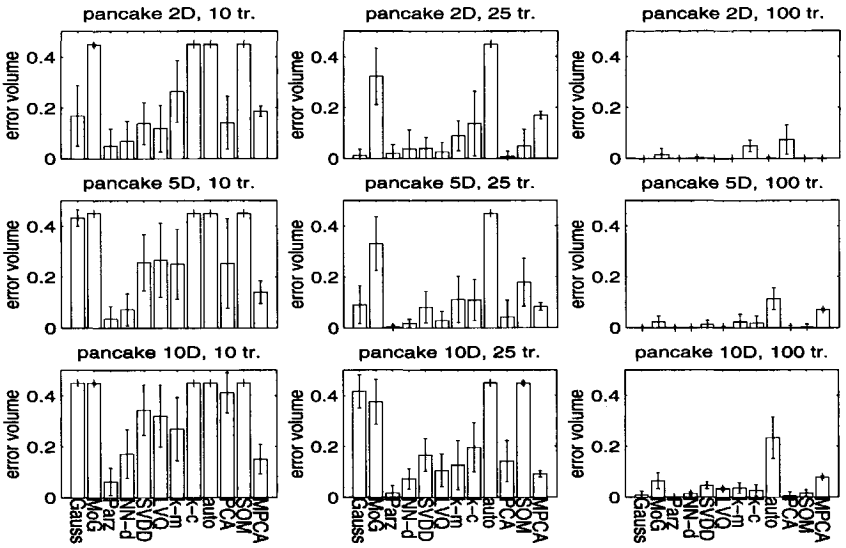


Fig. 4.12: Performances on the **pancake** data distribution for different sample sizes and dimensionalities.

set, but are on both sides of the subspace (see the right subplot in figure 4.2). This outlier distribution distinguishes the **ellipse** and **pancake** dataset; the target sets of both datasets are comparable. In figure 4.12 the error of all one-class classifiers on the **pancake** dataset is shown.

The difference in accuracy for small sample sizes between the **ellipse** and the **pancake** dataset is not very big (compare figure 4.12 and 4.8). The accuracies of the one-class methods are somewhat higher in the **pancake** data, because the overlap between the target and outlier class is (almost) zero in this distribution. But methods which fail in the **ellipse** dataset, also fail in the **pancake** data. The Parzen estimator still works well, and the NN-d slightly improved in comparison with the **ellipse**. For large sample sizes most methods obtain perfect separation, except for the auto-encoder and the mixture of PCA's.

The performance of the PCA in moderate sample sizes (i.e. 25 objects in the training set) is improved, in comparison with the other methods, which more or less perform equally. The performance of the LVQ is surprisingly good. The variance of the k-means and the LVQ is still large for small sample sizes. The SOM works well for large sample sizes, but behaves poorly for the very small sample sizes. Decreasing the size of the map (which is now a 10×10 grid of units) might improve performance again.

In the **ellipse** data set the PCA more or less models a subspace. Because the outliers are drawn uniformly around the target set, the overall performance of this method is not very good. In the **pancake** data, on the other hand, all outlier data is drawn from both sides of the subspace, but not from the subspace itself. The performance of the PCA,

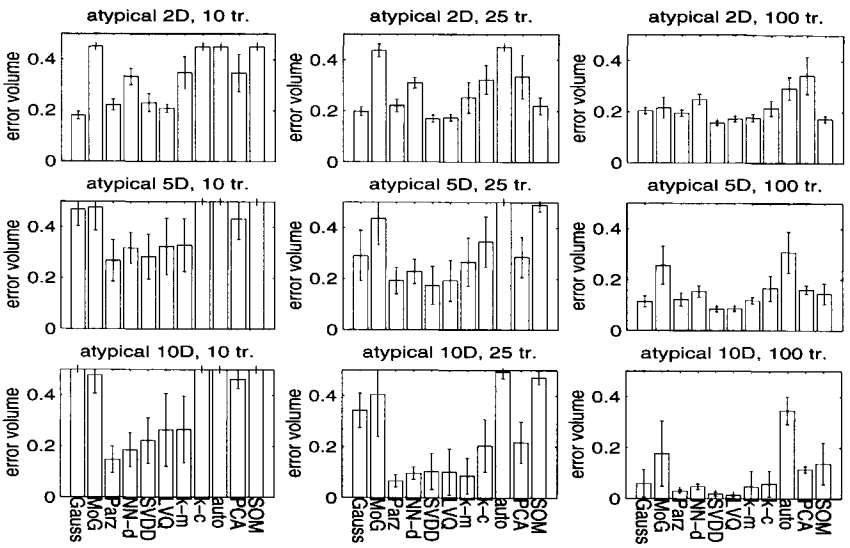


Fig. 4.13: Performances of all one-class classification methods on the `atypical` target data distribution for different dimensionalities and sample sizes.

therefore, significantly improves.

Summarizing, in all experiments in the last five sections, a major constraint was the sample size. The target class can only be characterized well and described well, when enough data is present. For instance, in almost all cases 10 objects in a 10 dimensional feature space is insufficient. Only very obvious characteristics, like the distribution in a subspace or the presence of a few clusters, can be detected. Although different methods perform differently on different types of data, for reasonably well representative data, the Parzen density estimator always obtains an acceptable solution. This will not be the case in the next section.

4.9 Atypical training set

In all previous experiments the training data was a good representation of the distribution we encountered in the testing data. Therefore, the density methods had a large advantage and it appeared that especially the Parzen density estimator was in most cases very apt in finding a good data description. In this section, we investigate the performances on data where the training data distribution differs significantly from the true distribution, only the boundary between the target and outlier data is the same (but the exact target density within the boundary might differ).

In figure 4.13 the results on the `atypical` dataset are given. In this dataset the true

distribution is uniform in a unit square, but the training objects are drawn from a triangular distribution (see figure 4.3 on page 91). The outlier data is generated in a box around the training data, as in all previous artificial datasets.

The results show that for high dimensional data the overlap between outlier and target data decreases and the performance of most methods improves. For very small sample sizes (10 objects) some methods completely fail, for instance, the mixture of Gaussians, k-centers, the auto-encoder and the SOM. The normal model and the PCA also collapse for feature spaces larger than 2-dimensional.

The other methods are more robust and give reasonable results. In particular, the Parzen density, the NN-d and the LVQ work well. The fact that the original target data distribution is square, assumes that all features are equally scaled. In all these methods this assumption is made, and in this small sample size situation this pays off. For this small sample size situation a rough estimate of the mean and variance is already sufficient to find a reasonable description. This holds up to 25 objects in 2 dimensions or 100 objects in 5 dimensions.

For large sample sizes more interesting results are visible. Here the difference between the target and outlier distribution becomes apparent. Although the density methods work well, the best performances are obtained by the SVDD and the SOM. Here, it is more advantageous to describe only a domain in feature space. The density methods suffer, and in particular, the Parzen density performs worse (even worse than the LVQ and k-means).

4.10 The presence of outliers

To measure the influence of outliers in the training set, a dataset containing extra outliers is created. The outliers are randomly drawn from a box 8 times as big as the target set (this box is obtained as explained on page 87). The outliers are not labeled as negative examples and can, therefore, only be rejected on the basis of their large distance to the bulk of the data.

In figure 4.14 the results on a Gaussian target distribution with 5 outliers are shown. The performance of the Gaussian method does not deteriorate with respect to performance on the `Gauss` dataset, because the covariance matrix is already regularized. Sometimes this Gaussian model works even better than in the previous case without example outliers (in the case of 25 training objects in 10 dimensions). The same is observed in the mixture of Gaussians. The few extra objects (although they are remote) improve the estimates of the free parameters in these methods.

As expected, the Parzen method does not change its performance very much. The width parameter is small enough not to accept large parts of the input space. The clustering methods, like the LVQ, k-means and k-centers also do not suffer from outliers in the training target set. In our experiments we already used more prototype vectors $\boldsymbol{\mu}_k$ than was strictly necessary. About 5 prototypes are used to describe the outliers and 5 for the 'real' target set. When 5 prototypes are used to describe the simple Gaussian target class, each prototype covers just for a very small hypersphere in the feature space. The

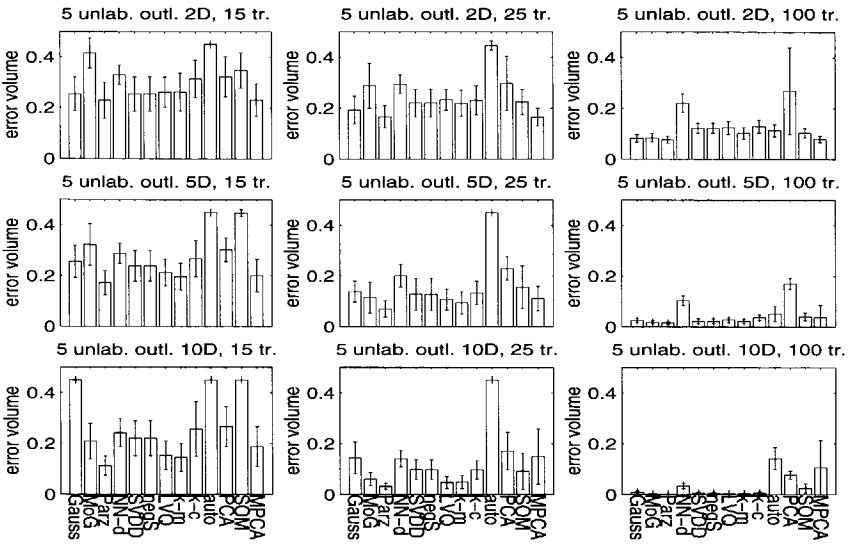


Fig. 4.14: Performances of all one-class classifiers on the Gauss target distribution containing 5 remote outliers. The outliers are labeled as target data.

distance from a test object $\mathbf{z}$ to the closest prototype should be very small, before object $\mathbf{z}$ is classified as a target object (formula (3.29)). The prototypes on the outlier objects therefore capture just small areas in the feature space. Also the support vector data description does not suffer from the outliers, due to its ability to reject objects when they are very remote (formula (2.3)).

Finally, the NN-d seriously suffers from the outliers, which is expected. The performance of the auto-encoder was already poor and no improvements are shown. Finally, the SOM and the PCA were already poor in the original Gauss problem, especially in 2D. The performances deteriorate for larger than 2-dimensional data in comparison with the Gaussian method.

In figure 4.15 the methods are trained again on the Gauss with 5 outlier objects, but now the outliers are labeled -1. We introduce an extra classifier, the neg-SVDD, which uses the information on outliers during its training (using the procedure given in section 2.2). When no example outliers are present, the neg-SVDD reduces to the classic SVDD. Therefore, it was not necessary in the previous experiments to distinguish between the SVDD and neg-SVDD. Now, it is expected that neg-SVDD will be able to obtain a tighter description of the target class. The SVDD will now be trained on the target data and it will ignore all objects which are labeled as outliers.

It is expected that only the Parzen, the neg-SVDD and the LVQ can improve their performance by using these examples. Looking at the performances it appears that only the neg-SVDD gives significant improvements (which now gives higher accuracies than the

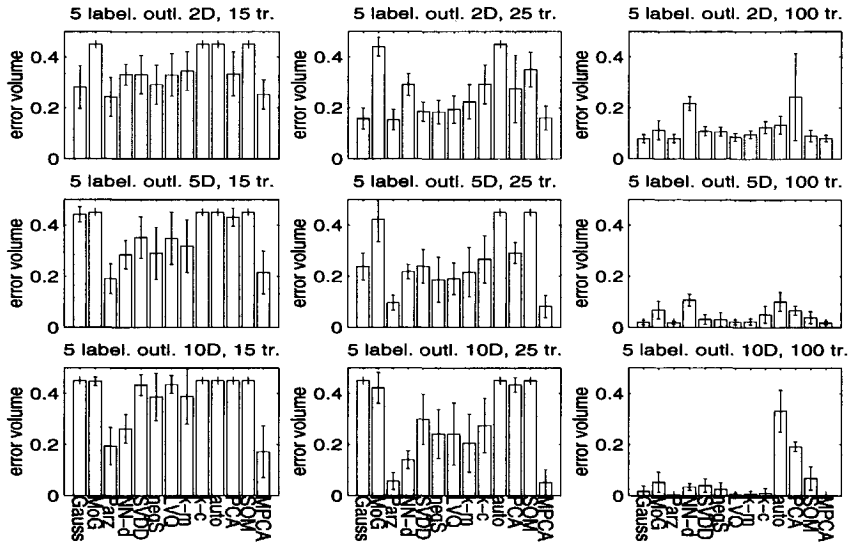


Fig. 4.15: Performance on the **Gauss** distribution with 5 prominent outliers. Outliers are explicitly labeled as outliers.

normal SVDD). The LVQ and Parzen do not show significant improvements. All other methods perform comparably to the original **Gauss** results. From these outcomes we can conclude that it is not easy to include a few example outliers in the training of one-class classifiers. In the construction of the neg-SVDD, one outlier object can significantly change the boundary around the data, and therefore can improve the performance. When a larger number of outlier examples is available, we might consider using a traditional two-class classifier, instead.

4.11 Guaranteed performance and time consumption

In all previous experiments we focused on the error performance of the methods, on how well the target data can be described and be distinguished from the outlier objects. It is also important to see how reliable a method is. In all the methods the user defines beforehand the fraction of target objects f_T which is allowed to fall outside the description. This fraction should be reflected in the acceptance rate of the target data f_{T+} in the testing set. In the following experiments we check this target acceptance rate with the acceptance rate observed in the testing set. To compare the reliability of the one-class methods, we define the error to measure the relative difference between the user defined fraction f_{T+} and the

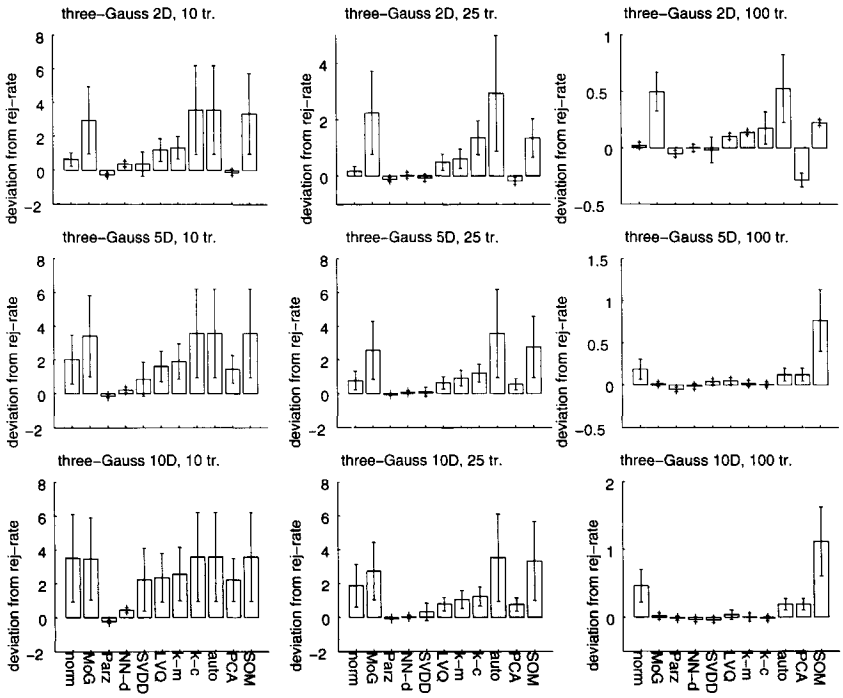


Fig. 4.16: Difference between the user defined target rejection rate and the rate measured on the test set. Experiment was on the **three-gauss** dataset.

fraction observed in the test set:

$$\mathcal{E}_{\text{rel}} = \frac{f_{T+} - \frac{1}{N}\sum_i I\left(p(\mathbf{z}_i) > \theta_{f_{T+}}\right)}{f_{T+}} \tag{4.1}$$

where N is the number of objects in the test set. For error values larger than 0 ($\mathcal{E}_{\text{rel}} > 0$) less target objects are accepted than was requested by f_{T+}, while for negative values ($\mathcal{E}_{\text{rel}} < 0$) more target objects are accepted.

In figure 4.16 the average $\mathcal{E}_{\text{rel}}$ over threshold values from $f_{T+} = 0.99$ to $f_{T+} = 0.5$ is shown. The results are given for the **three-gauss** dataset, but they are characteristic and do not differ significantly for the other artificial data sets. [5]

[5] This only holds for dataset for which the training set is a good representative for the true distribution, so it will not hold for the `atypical` dataset. When the training and true target distributions are different and only the area they cover is comparable, the two fractions will only be comparable when the boundary of the data is modeled (i.e. only for very high target acceptance rates f_{T+}). Integration over f_{T+} will not give sensible results then.

From this plot we see that the Parzen density and the NN-d work well. They do not assume a strong data model and give exactly the target rejection rate which is set by the user. The SVDD (and the neg-SVDD) show the same characteristics for larger sample sizes. For smaller sample sizes, on the other hand, the reliability suffers and is comparable to the LVQ, k-means and k-centers methods.

The mixture of Gaussian, the auto-encoder and the SOM show large deviations between the target rejection rates on the training and the testing sets. This does not improve by increasing the sample size, indicating that these methods do not match the data or are seriously overtrained. For the Gaussian model, the model perfectly matches the data. For small sample sizes the difference between the training and testing rejection rates is significant, but by increasing the sample size, this difference disappears.

Finally, the evaluation time is of interest. In most cases it is possible to train a method off-line and the training time is not of interest. For the practical use of the methods the evaluation time might be critical. In these experiments we investigate what the typical requirements for evaluation objects are.

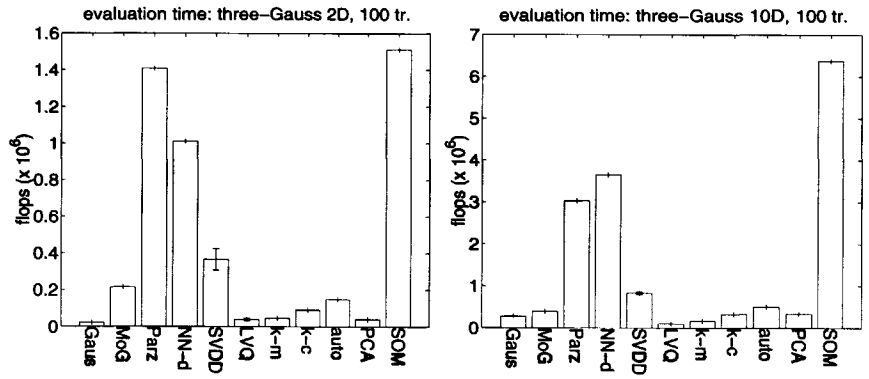


Fig. 4.17: Number of flops used for the evaluation of 1000 outliers in the `three-gauss` dataset.

The number of floating point operations (flops) for the evaluation of a test set by all methods is shown in figure 4.17. It is measured on a test set of 1000 objects. The most expensive methods are the Parzen density, the NN-d and the SOM. The cheapest are the Gaussian model, the LVQ, k-means and the PCA method. An average behavior is shown by the support vector methods, the mixture of Gaussians and the auto-encoder (for evaluation).

4.12 Pump vibration data

In the previous sections we considered data which show one important characteristic, such as unimodality, (non-) convexity or a distribution in a subspace. The performance of the different one-class methods can often be traced back to one of these characteristics. For

data distributions encountered in real life, the data characteristics are far less clear. These datasets are a mixture of different features, for instance, data distributed in a subspace, which is non-convex and which can be multimodal. How the performance of different methods on the data will be, is therefore very hard to predict beforehand. On the other hand, when the results of different methods are known, a profile of the data might be obtained. Here we will consider two real one-class problems, a machine diagnostics problem and a handwritten digit recognition problem.

The machine diagnostics problem has been encountered before in section 2.7, on page 49. It deals with a machine diagnostics: the characterization of the operation of a water pump [Tax et al., 1999]. In a small experimental setup, shown in figure 4.18, several normal and outlier situations can be simulated. The normal operation conditions are situations with different loads (by closing a membrane controlling the water flow) and speeds (from 46 to 54 Hz) of the pump. Also the measurements on faulty situations can be simulated, and consist of loose foundation, imbalance and a bearing failure. To detect dangerous conditions during the working of the pump, a one-class classification method trained on the normal working situation has to detect anomalous signals and raise an alarm.

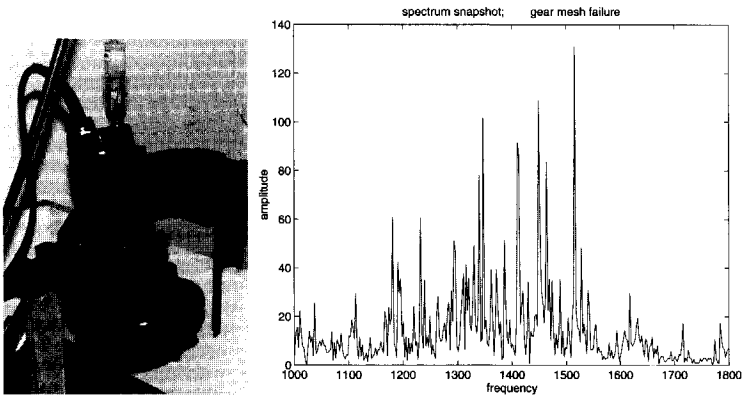


Fig. 4.18: On the left the physical setup of a small water pump. On the right, an example measurement on a pump showing gear mesh failure.

Vibration sensors are mounted on the pump. From the recorded time series (an example frequency spectrum is shown in the right subplot of figure 4.18) subsamples are taken and several features are calculated. It is well known that faults in rotating machines will be visible in the acceleration spectrum as increased harmonics of running speed or presence of sidebands around characteristic (structure-related) frequencies [Mitchell, 1993]. Due to overlap in series of harmonic components and noise, high spectral resolution may be required for adequate fault identification. This may lead to difficulties because of the curse of dimensionality: one needs large sample sizes in high-dimensional spaces in order to avoid overfitting of the train set. Hence we focused on relatively low feature dimensionality (64 spectral bins).

We consider four types of features, the power spectrum, the Auto Regressive model (AR features), the envelope spectrum and a MUSIC spectrum estimation:

power spectrum: standard power spectrum estimation, using Welch's averaged periodogram method. Data is normalized to the mean prior to spectrum estimation, and feature vectors (consisting of spectral amplitudes) are normalized with respect to the mean and standard deviation (in order to retain only sensitivity to the spectrum shape). The dataset contains 157 target objects for training and 143 target and 457 outliers for testing.

autoregressive modeling (AR model): another way to use second-order correlation information as a feature is to model the time series with an autoregressive model (AR-model). For comparison with other features, an AR(64)-model was used (which seemed sufficient to distinguish between the operation modes) and model coefficients were used as features. The dataset contains 151 target objects for training and 149 target and 451 outliers for testing.

envelope spectrum: a measurement time series is demodulated using the Hilbert transform, and from this cleaned signal (probably containing information on periodic impulsive behavior) a spectrum was determined using the above method [Randall, 1987]. Prior to the demodulation a bandpass-filtering in the interval 125 - 250 Hz (using a wavelet decomposition with Daubechies-wavelets of order 4) was performed: it is expected that gear mesh frequencies will be present in this band and impulses due to pitting are expected to be present as sidebands. This dataset contains 157 target objects for training and 143 target and 457 outliers for testing.

MUSIC spectrum: if a time series can be modeled by sinusoids plus a noise term, a MUSIC frequency estimator [Proakis and Manolakis, 1992] can be used to focus on the important spectral components. A statistic can be computed that tends to infinity when a signal vector belongs to the, so-called, signal subspace. When one expects amplitudes at a finite number of discrete frequencies to be a discriminant indicator, MUSIC features may enable good separability while keeping feature size (relatively) small. The dataset contains 91 target objects for training and 89 target and 271 outliers for testing.

The results for the power spectrum are shown in the left subplot in figure 4.19. The differences between the methods are not very large, but the Parzen, the SOM and the k-center method perform best. Using the default settings for the mixture of Gaussians, the cluster algorithms (LVQ, k-means, k-centers, SOM) and the auto-encoder result in worse performance. It appears that by increasing the number of clusters for the LVQ, k-means and k-centers the performance is improved, while for the SOM and the mixture of Gaussians the number of clusters had to be decreased. All in all, the performances could be improved by an error of 0.05 (on the scale of figure 4.19). Given the limited amount of available data, this is quite reasonable performance.

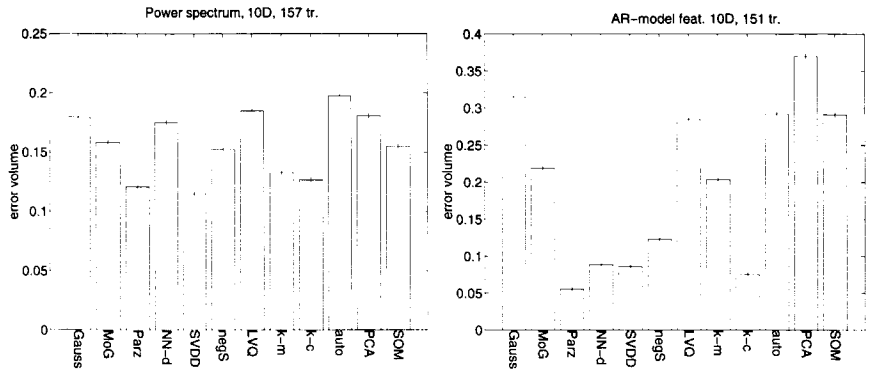


Fig. 4.19: Performance of the one-class classifiers on the pump datasets. On the left 10-dimensional power spectrum data, on the right 10-dimensional AR-model features.

The comparable performance of the normal density, mixture of Gaussians and the Parzen density indicates that we have one cluster. Surprisingly the LVQ and the k-means perform very poorly, which might indicate that different parts of the dataset show different scaling of the features. Furthermore, the LVQ is harmed by the tendency to stop adjusting its prototypes once all training objects are classified well. When target and outlier data are very close and the LVQ is not trained using the outliers, the generalization by the LVQ might be poor.

The overall data signature resembles the signature of the **banana** for large sample sizes (except for the principal component analyzer, which performs here comparably with the other methods). The fact that the performance of the NN-d is not that good, indicates that we do not have very small sample sizes. Combined with the good performance of the PCA and especially the self-organizing map this shows that the data is mainly distributed in a subspace.

The characteristics of the AR model features (right graph in figure 4.19) are completely different from the power spectrum features. Here we have clearly separate clusters, but no clear subspace is available.

In figure 4.20 the results for the envelope spectrum data and the MUSIC features are presented. Comparing them with figure 4.19, similar characteristics are present in the data. The envelope spectrum corresponds to the power spectrum data, again one cluster is available. There appears to be severe overlap between the target and outlier data; the performance of all methods is very poor. These envelope spectrum features are clearly not powerful enough to distinguish between target and outlier situations. The best performance is obtained using the Parzen density method.

The MUSIC frequency estimation can be compared to the AR feature dataset. Here the data is more clustered, but also the performance is somewhat worse than in case of the AR features. Due to overlap between the target and outlier data the SVDD performed very

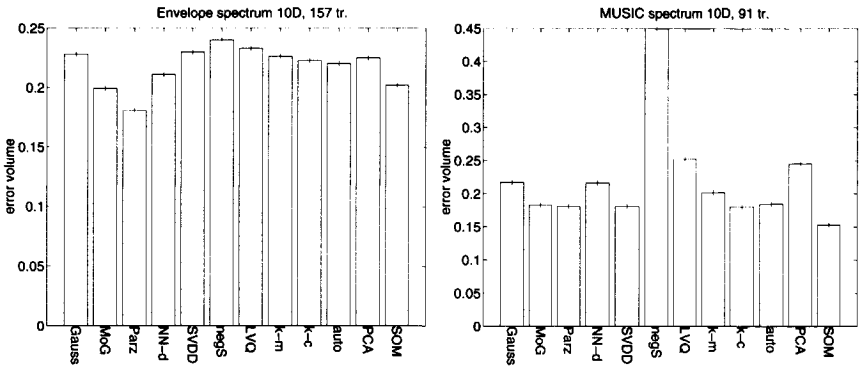


Fig. 4.20: Performance of the one-class classifiers on the pump datasets. On the left 10 dimensional envelope spectrum data, on the right 10-dimensional MUSIC spectrum data.

poorly, especially with the low target rejection rates. For the first time we see (relatively) good performance of the auto-encoder network, which is one of the best methods for this dataset. The Parzen density is now outperformed by the mixture of Gaussians, the SVDD, the auto-encoder and the SOM. This is not very significant though, a large overlap between the target and outlier objects is present and all methods have problems to distinguish between the target and outlier objects.

4.13 Handwritten digit data

At present no standard one-class classification datasets exist. Most repositories only consider multi-class classification problems [Blake et al., 1998]. These traditional classification problems can be changed into one-class problems, when just one class is treated as the target class, and the other classes are considered as outliers. By construction of the one-class classification methods (which mainly focus on the target class) they will perform worse than traditional classifiers which use information of all classes.

For the last experiments in this chapter, a handwritten digit database is used. This dataset contains 4 different feature sets for each object. Digits were extracted from 9 original maps of a Dutch public utility. The maps represent the position of a conduit system with respect to certain landmarks and were hand drawn by a large group of drawing engineers over a period of more than 25 years. The dataset is composed of separate digit images and consist of 10 classes, each having 200 examples (thus assuming equal class probabilities). From the set of 2000 digits 4 types of feature sets were extracted: Zernike moments, Fourier descriptors, profile images and image pixel vectors. For a more elaborate explanation of the features, see appendix C. A few examples of the digits are shown in figure 4.21.

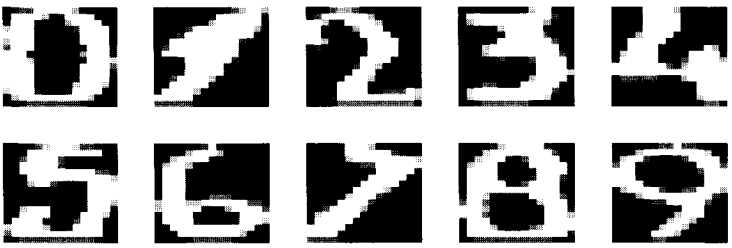


Fig. 4.21: Some examples of the handwritten digits.

Table 4.1: Results of the one-class classification methods on handwritten digit data given by the different feature sets. Class '0' is treated as the target class. Bold numbers indicate the best accuracy for a considered dataset (integrated ROC $\times 100$).

set	Gauss	MoG	Par	NN-d	SVD	LVQ	k-m	k-c	aut	PCA	SOM
Zernike	2.80	1.15	0.45	11.67	4.63	**0.16**	1.98	0.25	2.46	1.47	0.51
Fourier	4.00	0.93	**0.04**	1.60	16.22	0.21	0.79	1.02	0.81	3.60	1.01
Profile	0.65	0.35	**0.08**	2.18	9.58	1.04	2.33	2.14	5.08	4.48	0.35
Pixel	21.60	9.99	**0.09**	2.36	17.22	0.62	1.45	0.86	8.60	26.80	0.39

Treating class '0' as the target class, the performances of the one-class classifiers are presented in table 4.1. Per definition, the outlier class consists of the objects from the other 9 classes. Because we do not create artificial outlier data around the target class, but we are given the same set of outlier objects in all feature spaces, we can compare performances on different feature spaces (and different feature dimensionalities). Thus, we do not consider the volumes of the target set, but the classification performance.

The same abbreviations for the classifiers as in the previous chapter are used (see page 95). In most datasets the Parzen density estimator is clearly the best. In the Profile and Zernike data the '0' class appears to be clustered well, and in these datasets the clustering methods such as LVQ, k-means and k-centers perform comparably to the Parzen method. The performance of the SVDD on the pixel and Fourier data, indicates that the sample sizes in these dataset are not sufficient. Very likely the 0 class is multimodal with a complex boundary. The Gaussian model does not work very well (especially in the pixel dataset), while the performance of the k-means clustering or the k-center method is quite acceptable.

When a good data description of the '0' class is required, the best data set to use is the Fourier dataset. When the storage requirements are important, we can also choose the LVQ on the Zernike dataset (which stores just 10 objects, instead of the complete training set). Note that both the Zernike and the Fourier datasets are rotation invariant, which is a clear advantage in describing the digit 0.

4.14 Conclusions

What can we conclude from the experiments performed in this chapter? We have investigated two types of problems. In the first type the distributions of the target objects in the training set and testing sets are identical. When these distributions are (approximately) equal, the best data description methods are the density methods. In particular, the Parzen density estimator appears to perform well on the considered data. Other density methods sometimes use assumptions which are not satisfied in the data (for instance, the Gaussian model assumes a unimodal density), and a large bias is introduced. In the second type of problems, the distributions of the target objects in the training set and testing sets are different. In such cases, density methods do not work well, and the boundary methods are to be preferred. To provide a more thorough summary, the applicability of the one-class methods on all types of data is summarized in table 4.2. A '+' indicates that good performance can be expected, while a '−' indicates that this one-class classifier is not suited for this data. A '+/−' indicates that with special cares, the method can be applied.

Considering the sample sizes, it is very hard to obtain a data description for very small sample sizes (about 10 objects in 2 or higher dimensional feature spaces). There is not enough information to extract more than a rough estimation of the mean and variance of the data. The Parzen density estimator depends heavily on the continuity assumption and it can extrapolate only in the close neighborhood around the given objects. In most problems with identical distributions for the training and testing sets, this can be sufficient to give reasonable results. This extrapolation is also present in the NN-d method, especially for small sample sizes.

For large sample sizes, about 100 objects in a 2-dimensional feature space, all methods perform well. In such circumstances it becomes important how well the method fits to the data. When a strong model is assumed, for instance, a unimodal Gaussian distribution, the risk of introducing large bias for a particular problem increases. Here, the flexible Parzen model again performs well, but the drawback is that for increasing training sizes, the evaluation times for testing new objects becomes quite large.

The clustering methods: LVQ, k-means and k-centers, often obtain comparable accuracy and never perform very poorly. The only constraint is that a sufficient number of prototypes is chosen, else a large bias is introduced. Given a sufficient number of prototypes, the methods are not only quite robust against varying sample sizes, but also against scaling of features and clustering of data. Only for very small sample sizes the methods break down. On the other hand the auto-encoder requires careful training and adjusting and it is not very robust. It requires quite some data for training, sometimes even 100 objects in a 2-dimensional feature space is not sufficient. In some lucky cases the training algorithm obtains very good results, but subsequent bad results deteriorate the average performance.

Finally, the performance of the the support vector data description is worse on the data with identical training and testing distributions. The SVDD does not show sufficient extrapolation ability to generalize very well in small sample size situations. For larger sample sizes, the performance of the SVDD approaches the Parzen density estimator. When

the training distribution significantly differs from the testing distribution, the situation changes completely. The density methods show now very poor results, but the other methods (especially the SVDD and the clustering methods) achieve good generalization performance. This characteristic of the SVDD is more important in real world datasets than in the artificially created data. In most cases the training objects are not obtained directly from real life situations, but are sampled and recorded especially to construct a set of examples. The operator has to decide which situations are typical and should therefore be included. To estimate the correct probability of occurrence might be very hard, but it might still be possible to sample from the correct area in the feature space. In these cases the SVDD is still able to find a good data description.

As far as the robustness of the methods is concerned, due to the fact that the acceptance threshold is set such that it accepts a certain fraction of the target data (for instance 95%), the methods are already somewhat robust against outliers. Only the NN-d, the PCA and the SOM seriously suffer from a few outliers. Here the model is influenced heavily by the few outliers, and changing the acceptance threshold cannot counter-balance this influence.

In the experiments done on non-artificial data, it appears that real data characteristics are far more complex than these of the artificial datasets. It makes the evaluation of the various schemes more difficult. In many cases we see that the Parzen density and clustering schemes work well for reasonably well-sampled data, while for somewhat atypical training data the SVDD performs well (this is the case of the pump vibration datasets).

Table 4.2: The applicability of all one-class classification methods on data with different characteristics.

one-class classifier	small sample size	scaling insensitivity	clustering	convexity
Gaussian model	-	+	-	-
Mixture of Gaussians	+/-	-	+	+/-
Parzen density	+	+	+	+
NN-distance	+	-	+	+
SVDD	-	+	+/-	+
neg-SVDD	-	+	+/-	+
k-means	+	+/-	+	+/-
k-centers	+	+/-	+	+/-
auto-encoder	-	+	-	+/-
PCA	-	+	-	-
SOM	-	-	-	-

one-class classifier	subspaces	atypical data	robustness	guaranteed performance
Gaussian model	+	+/-	+	+
Mixture of Gaussians	+	-	+	-
Parzen density	+	+/-	+	+
NN-distance	-	-	-	+
SVDD	-	+	+	+
neg-SVDD	-	+	+	+
k-means	+/-	+/-	+	+
k-centers	+/-	+/-	+	+
auto-encoder	-	-	-	-
PCA	+	-	-	-
SOM	+	-	+/-	-

5. COMBINING DESCRIPTIONS

In the previous chapters we focused on the problem of finding a good classifier for one-class classification problems. Depending on the type of data (the sample sizes, the data distribution and how well the true distribution could be sampled), the best fitting data description had to be found. Unfortunately, classifiers hardly ever fit the data distribution optimally. Using just the best classifier and discarding the classifiers with poorer performance might waste valuable information [Wolpert, 1992]. To improve performance, of different classifiers (which may differ in complexity or training algorithm) can be combined. This may not only increase the performance, but can also increase the robustness of the classification [Sharkey and Sharkey, 1995]. Furthermore, it appears to be possible to include prior knowledge in the classification by using specific combining rules; we will see this at the end of this chapter.

Classifiers can be combined in several ways. The first one is to use different feature sets and to combine the classifiers trained on each of the feature sets. The second approach is to train several different classifiers on one feature set and combine these. In this chapter both approaches will be discussed. In section 5.1 some basic properties of combining rules are given, focusing on the product and mean combination rules in conventional classification problems [Tax et al., 2000]. In section 5.2 this is applied to the one-class problems, covering the combining of different datasets and different classifiers. Finally, in section 5.3 the combining rules is used in an image database application where the combining rules can incorporate constraints supplied by the user.

5.1 Combining conventional classifiers

A large number of combining schemes for conventional classifiers exists. In general, three types of situations in combining classifiers have been identified [Xu et al., 1992]. In the first type, each classifier outputs one single class label and these labels have to be combined [Battiti and Colla, 1994]. In the second type, the classifier outputs sets of class labels ranked according to their order of likelihood [Tumer and Ghosh, 1995]. The third type involves the combination of real valued outputs for each of the classes (most often posterior probabilities [Jacobs, 1995], sometimes evidences [Rogova, 1994]).

Commonly a combined decision is obtained by just averaging the estimated posterior probabilities of different classifiers. This simple algorithm already gives very good results [Hashem, 1994, Tanigushi and Tresp, 1997]. This is somewhat surprising, especially considering the fact that the averaging of posterior probabilities is not based on some solid

(Bayesian) foundation. When the Bayes theorem is adopted for the combination of different classifiers, under the assumption of independence, a product combination rule automatically appears: the outputs of the individual classifiers are multiplied and then normalized (this is also called a logarithmic opinion pool [Benediktsson and Swain, 1992]).

Some classifiers immediately offer estimates of posterior probabilities, such as the multilayer perceptron, trained with backpropagation [Ruck et al., 1990] or by maximizing the cross-entropy on the network outputs [Bishop, 1995]. In other classification methods, probabilities are harder to obtain. For instance, the posterior probability in the 1-nearest-neighbor classifier is not defined. When posterior probabilities can be defined, they are often only reliable for large training sets, as for example in the case of the k-nearest neighbor classifier. When posterior probability estimates are not available, these probabilities have to be approached, or the combining has to be applied to the labels or rankings.

In [Kittler et al., 1996] and in [Kittler et al., 1997] a theoretical framework for combining (estimated posterior probabilities from) classifiers is developed. For different types of combination rules (under which the minimum and maximum rules, weighted averages, mean and product rule) derivations are given. It has been shown [Kittler et al., 1996] that when classifiers are applied on identical data representations, the classifiers estimate the same class posterior probability, potentially suffering from the same noise in the data. To suppress the errors in these estimates and overfitting of the individual classifiers, the classifier outputs should be averaged. On the other hand, when independent data representations are available, classifier outcomes should be multiplied to gain maximally from the independent representations.

Assume that each object $\mathbf{x}$ belongs to one of C classes $\omega_j, j = 1...C$.[1] When R measurement vectors $\mathbf{x}^1, ..., \mathbf{x}^R$ from feature spaces $\mathcal{X}_1, ..., \mathcal{X}_R$ are available, the probability $p(\omega_j|\mathbf{x}^1, ..., \mathbf{x}^R)$ has to be approximated to make a classification (see also [Kittler et al., 1996]). In each of the R feature spaces, a classifier is constructed which approximates the true posterior class probability $p(\omega_j|\mathbf{x}^k)$ in $\mathcal{X}_k$:

$$f_j^k(\mathbf{x}^k) = p(\omega_j|\mathbf{x}^k) + \epsilon_j^k(\mathbf{x}^k) \tag{5.1}$$

where $\epsilon_j^k(\mathbf{x}^k)$ is the error made by classifier k on the probability estimate that object $\mathbf{x}^k$ belongs to class ω_j. A combination rule combines these $f_j^k(\mathbf{x}^k)$ to approximate $p(\omega_j|\mathbf{x}^1, ..., \mathbf{x}^R)$ as well as possible.

Two extreme cases can be distinguished, the first in which $\mathcal{X}_1 = \mathcal{X}_2 = ... = \mathcal{X}_R$, the second where $\mathcal{X}_1, ..., \mathcal{X}_R$ are different and assumed to be independent. In the first case, the classifiers all use the same data $\mathbf{x}$: $p(\mathbf{x}^1, ..., \mathbf{x}^R|\omega_j) = p(\mathbf{x}^1|\omega_j)I(\mathbf{x}^2 = \mathbf{x}^1) \cdot ... \cdot I(\mathbf{x}^R = \mathbf{x}^1)$. This trivially leads to:

$$p(\omega_j|\mathbf{x}^1, ..., \mathbf{x}^R) = p(\omega_j|\mathbf{x}^k) \quad 1 \leq k \leq R \tag{5.2}$$

where $p(\omega_j|\mathbf{x}^k)$ is estimated by $f_j^k(\mathbf{x}^k)$. When we assume zero-mean error for $\epsilon_j^k(\mathbf{x}^k)$ (i.e. zero bias), all $f_j^k(\mathbf{x}^k)$'s can be averaged to obtain a less error-sensitive estimation. This

[1] Here $\mathbf{x}$ denotes the object in an unspecified feature space. In different feature spaces, one can obtain representations $\mathbf{x}^k$ for this object $\mathbf{x}$.

leads to the mean combination rule:

$$f_j(\mathbf{x}^1, ..., \mathbf{x}^R) = \frac{1}{R} \sum_{k=1}^{R} f_j^k(\mathbf{x}^k) \tag{5.3}$$

In the second case all feature spaces are different and class conditionally independent. The probabilities can be written as: $p(\mathbf{x}^1, ..., \mathbf{x}^R|\omega_j) = p(\mathbf{x}^1|\omega_j) \cdot p(\mathbf{x}^2|\omega_j) \cdot ... \cdot p(\mathbf{x}^R|\omega_j)$. Using the Bayes rule, we derive:

$$\begin{aligned} p(\omega_j|\mathbf{x}^1, ..., \mathbf{x}^R) = \frac{p(\mathbf{x}_1, ..., \mathbf{x}_R|\omega_j)p(\omega_j)}{p(\mathbf{x}_1, ..., \mathbf{x}_R)} &= \frac{\prod_k p(\mathbf{x}_k|\omega_j)p(\omega_j)}{p(\mathbf{x}_1, ..., \mathbf{x}_R)} = \\ \frac{\left[\prod_k \frac{p(\omega_j|\mathbf{x}_k)p(\mathbf{x}_k)}{p(\omega_j)}\right] p(\omega_j)}{p(\mathbf{x}_1, ..., \mathbf{x}_R)} &= \frac{\prod_k p(\omega_j|\mathbf{x}_k) \prod_k p(\mathbf{x}_k)/p(\omega_j)^{R-1}}{p(\mathbf{x}_1, ..., \mathbf{x}_R)} \\ = \frac{\prod_k p(\omega_j|\mathbf{x}_k) \prod_k p(\mathbf{x}_k)/p(\omega_j)^{R-1}}{\sum_{j'} \left\{ \prod_k p(\omega_j|\mathbf{x}_k) \prod_k p(\mathbf{x}_k)/p(\omega_j)^{R-1} \right\}} &= \frac{\prod_k p(\omega_j|\mathbf{x}^k)/p(\omega_j)^{R-1}}{\sum_{j'} \left\{ \prod_{k'} p(\omega_{j'}|\mathbf{x}^{k'})/p(\omega_{j'})^{R-1} \right\}} \end{aligned} \tag{5.4}$$

In case of equal prior class probabilities ($p(\omega_j) = 1/C$) and negligible errors $\epsilon_j^k(\mathbf{x}^k) = 0$ (!):

$$p(\omega_j|\mathbf{x}^1, ..., \mathbf{x}^R) = \frac{\prod_k p(\omega_j|\mathbf{x}^k)/C^{R-1}}{\sum_{j'} \left\{ \prod_{k'} p(\omega_{j'}|\mathbf{x}^{k'})/C^{R-1} \right\}} \tag{5.5}$$

$$= \frac{C \prod_k p(\omega_j|\mathbf{x}^k)}{\sum_{j'} \left\{ C \prod_{k'} p(\omega_{j'}|\mathbf{x}^{k'}) \right\}} \simeq \frac{\prod_k f_j^k(\mathbf{x}^k)}{\sum_{j'} \left\{ \prod_{k'} f_{j'}^k(\mathbf{x}^k) \right\}} \tag{5.6}$$

So, this approximation results in the product combination rule:

$$f_j(\mathbf{x}^1, ..., \mathbf{x}^R) = \frac{\prod_{k=1}^R f_j^k(\mathbf{x}^k)}{\sum_{j'} \prod_{k=1}^R f_{j'}^k(\mathbf{x}^k)} \tag{5.7}$$

In [Tax et al., 1997] comparisons between the average combination rule and the product combination rule are made. It was confirmed that when independent data sets were available, the product combination rule should be used. Only in the case of poor posterior probability estimates, should the more fault tolerant mean combination rule is to be used.

5.1.1 Differences between averaging and multiplying

From the derivation of the combining rules (5.3) and (5.7), we would expect that the two rules will be useful under different conditions. The mean combination rule will be especially useful in case of identical or very highly correlated feature spaces in which the classifiers make independent errors. The product combination rule is apt for different, class-conditional independent feature spaces where classifiers make small estimation errors. In some situations though, the performances of the mean and product rule will be equal. We will investigate this with a very simple model.

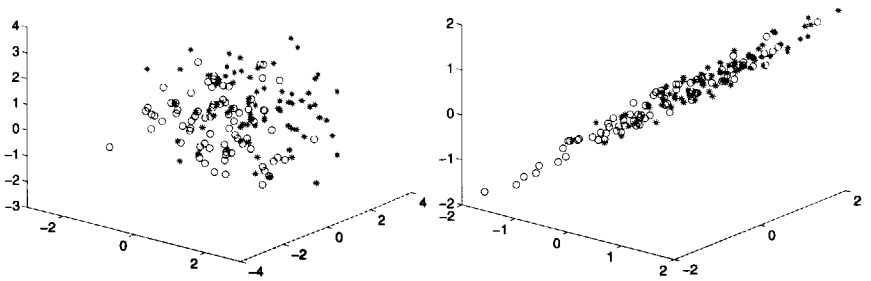


Fig. 5.1: Data distribution in 3 dimensions. On the left the data is uncorrelated, on the right it is correlated. The data consist of two classes where the first one is marked by circles and the second by crosses.

In this model data consists of two classes in an R-dimensional space, each normally distributed (see figure 5.1 for a 3-dimensional example). The first class is centered on the origin, while the second on $(1, 1, .., 1)/\sqrt{R}$. The covariance matrix can be adjusted to change the correlation between the feature values. It can be changed from identity, in which case the data is uncorrelated for each component (see left subplot in figure 5.1), to complete correlation, in which case all data is perfectly correlated for each component (right subplot in 5.1). Each of the R classifiers uses just one of the features, from which it has to estimate the posterior probabilities and a decision boundary. Combining the predictions of the classifiers in the two extremes, perfect correlation and complete independence of the data, will indicate where one combination rule can be preferred over the other.

In case of two class classification problems, we can derive conditions for which the rules behave the same. Assume we have equal class probabilities for the classes. When the product rule classifies some object $\mathbf{x}$ to class ω_j, (j can be 1 or 2) then:

$$\prod_k f_1^k(\mathbf{x}^k) > \prod_k f_2^k(\mathbf{x}^k) = \prod_k (1 - f_1^k(\mathbf{x}^k)) \tag{5.8}$$

We can decompose the output of one of the classifiers $f_j^k(\mathbf{x}^k)$ into two parts. The first part is the average over all other classifiers, $\bar{f}_j(\mathbf{x}) = \frac{1}{R}\sum_k f_j^k(\mathbf{x}^k)$, the second part is a rest term

ζ_j^k. Thus we defined:

$$f_j^k(\mathbf{x}^k) = \bar{f}_j(\mathbf{x}) + \zeta_j^k(\mathbf{x}^k) \tag{5.9}$$

Therefore, per definition holds $\sum_k \zeta_j^k(\mathbf{x}^k) = 0$. This is basically the bias-variance decomposition [Geman et al., 1992] (see chapter 1). The different values for ζ_j^k account for the variance, while $\sum_k \zeta_j^k = 0$ indicates that there is no bias. We can expand the terms in (5.8):[2]

$$\begin{aligned}\prod_k f_j^k &= \bar{f}_j^R \left[1 + \sum_{k,k'} \frac{\zeta_j^k \zeta_j^{k'}}{\bar{f}_j^2} + \ldots\right] \\ &= \bar{f}_j^R + \bar{f}_j^{R-2} \sum_{k,k'} \zeta_j^k \zeta_j^{k'} + \bar{f}_j^{R-3} \sum_{k,k',k''} \zeta_j^k \zeta_j^{k'} \zeta_j^{k''} + \ldots \end{aligned} \tag{5.10}$$

The second term with $\bar{f}_j^{R-1}$ in the summation can be dropped because $\sum_k \zeta_j^k = 0$.

For two class problems $\zeta_1^k = -\zeta_2^k$. All sums over ζ_j^k's in the expansion of $\prod_k f_1^k$ and $\prod_k f_2^k$ will be equal, except for the terms with summations over an odd number of classifier outputs (here the signs are opposite) and for the factors $\bar{f}_j^{R-n}, n = 2, 3, \ldots$. When we want to consider the inequality $\prod_k f_1^k(\mathbf{x}^k) > \prod_k f_2^k(\mathbf{x}^k)$ we therefore only have to consider these terms. Resubstituting (5.10) in (5.8) results in:

$$\begin{aligned}&\bar{f}_1^R + \bar{f}_1^{R-2} \sum_{k,k'} \zeta_1^k \zeta_1^{k'} + \bar{f}_1^{R-3} \sum_{k,k',k''} \zeta_1^k \zeta_1^{k'} \zeta_1^{k''} + \ldots > \\ &(1 - \bar{f}_1)^R + (1 - \bar{f}_1)^{R-2} \sum_{k,k'} \zeta_1^k \zeta_1^{k'} - (1 - \bar{f}_1)^{R-3} \sum_{k,k',k''} \zeta_1^k \zeta_j^{k'} \zeta_1^{k''} + \ldots \end{aligned} \tag{5.11}$$

which can be simplified to:

$$\begin{aligned}&\bar{f}_1^R + K' \bar{f}_1^{R-2} + K'' \bar{f}_1^{R-3} + \ldots > \\ &\quad (1 - \bar{f}_1)^R + K'(1 - \bar{f}_1)^{R-2} - K''(1 - \bar{f}_1)^{R-3} + \ldots \end{aligned} \tag{5.12}$$

where K' and K'' are constants.

When there are no outliers and ζ_j^k is smaller than $\bar{f}_j$, then the term $K'' = \sum_{k,k',k''} \zeta_j^k \zeta_j^{k'} \zeta_j^{k''}$ will stay small. In table 5.1 the size of $\bar{f}_j^{R-3} \sum_{k,k',k''} \zeta_j^k \zeta_j^{k'} \zeta_j^{k''}$ relative to the previous two terms is shown for different number of classes. Three classifiers were trained on the artificial dataset shown in figure 5.1. The correlation was set to 0.5. For classification problems with more than two classes, the mean of a cluster k was placed at $(k, k, k)/\sqrt{R}$. These values are the largest absolute values over all classes.

Especially in the two class problem the third term is very small. The relative large value of the second term does not influence the classification, because the signs of these terms is equal for all classes. This means that when we start with the product combination

[2] For clarity of notation, we drop all $\mathbf{x}^k$ in f_j^k, $\bar{f}_j$ and ζ_j^k.

Table 5.1: Sizes of higher order sums in formula (5.11) relative to the first term. $R = 3$ classifiers trained on data from figure 5.1 were combined.

# classes	term 1	term 2	term 3
2	1.0	0.16	0.00016
3	1.0	0.22	0.00971
4	1.0	0.23	0.01862
5	1.0	0.30	0.02231

rule (given by formula (5.8)) and we apply the approximation given by (5.10), we get the new combination rule: classify object $\mathbf{x}$ to class ω_j, (j can be 1 or 2) when:

$$\left(\bar{f}_1^2 + K'\right) \bar{f}_1^{R-2} > \left((1-\bar{f}_1)^2 + K'\right)(1-\bar{f}_1)^{R-2} \tag{5.13}$$

This is a rescaled version of the mean combination rule for a two class problem. In the product combination rule the output values are just shifted into the direction of the extremes; for $\bar{f}_j < 0.5$ to 0 and for values $\bar{f}_j > 0.5$ to 1. In [Tax et al., 1997] it was shown that for this simple two-class problem the differences between the product and mean combination rule was very small for all correlations (i.e. from no correlation to perfect correlation between the individual feature sets). Only when this artificial classification problem was extended to a multiclass problem, the differences between the mean and the product combination rule appear for the different correlations in the feature sets.

The robustness of the mean combination rule with respect to the product combination rule is shown by [Kittler et al., 1996] in which formula (5.1) is expanded, comparable with the expansion in formula (5.10). It can be shown that the combined classifiers using a product combination rule approximate the error-free classifier up to a factor $\left[1 + \sum_k \frac{\zeta_j^k}{p(\omega_j|\mathbf{x}^k)}\right]$ while in the mean combination rule the factor is $\left[1 + \frac{\sum_k \zeta_j^k}{\sum_k p(\omega_j|\mathbf{x}^k)}\right]$. Note that $p(\omega_j|\mathbf{x}^k) \leq 1$, so errors are amplified by the product rule. In the mean combination rule the errors are divided by the sum of posterior probabilities and in particular, for the winning class, where these probabilities are large, the errors are severely dampened.

5.2 Combining one-class classifiers

In the previous section the combining of a general set of classifiers (using the mean and the product combination rule) is discussed, where it is assumed that the classifiers approximate the posterior probability of an object $\mathbf{x}$ for a class ω_j. When a classifier is not based on some type of density estimation, the posterior probability can be estimated by a heuristic approach, depending on the type of classifier. For one-class classifiers another situation is present. Here the problem is always a two-class problem where only information about

one class, the target class, is known (i.e. $p(\mathbf{x}|\omega_T)$, when the training set is an i.i.d. sample from the true target distribution), but the outlier class $p(\mathbf{x}|\omega_O)$ is not.

Using the Bayes rule, the posterior probability for the target class can be computed by:

$$p(\omega_T|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_T)p(\omega_T)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|\omega_T)p(\omega_T)}{p(\mathbf{x}|\omega_T)p(\omega_T) + p(\mathbf{x}|\omega_O)p(\omega_O)} \tag{5.14}$$

Because the outlier distribution $p(\mathbf{x}|\omega_O)$ is unknown, and even the prior probabilities $p(\omega_T)$ and $p(\omega_O)$ are very hard to estimate, equation (5.14) cannot be used directly. The problem is solved when an outlier distribution is assumed. When $p(\mathbf{x}|\omega_O)$ is independent of $\mathbf{x}$, i.e. it is a uniform distribution in the area of the feature space that we are considering, $p(\mathbf{x}|\omega_T)$ can be used instead of $p(\omega_T|\mathbf{x})$.

To complicate matters further, no i.i.d. sample from the target distribution is available, and only a dataset indicating the boundaries of the target distribution is available. Then the one-class classifiers do not model the complete $p(\mathbf{x}|\omega_T)$, but they only provide a yes-no output: object $\mathbf{x}$ is either accepted or rejected. In one-class classification the binary decision threshold θ_{f_T} is optimized to obtain a certain target acceptance rate f_T on the training set:

$$\theta_{f_T} : \int I(p(\mathbf{x}|\omega_T) \geq \theta_{f_T}) d\mathbf{x} = f_T \tag{5.15}$$

(or $\theta_{f_T} : \int I(d(\mathbf{x}|\omega_T) \leq \theta_{f_T}) d\mathbf{x} = f_T$ for methods using distance d). This means that for a given object $\mathbf{x}$ from the target set, the one-class classifier only approximates

$$p(\text{accepting } \mathbf{x}|\omega_T) = f_{T+} \tag{5.16}$$

$$p(\text{rejecting } \mathbf{x}|\omega_T) = 1 - f_{T+} = f_{T-} \tag{5.17}$$

as well as possible.

The different measures for the posterior probabilities are shown in figure 5.2 for an artificial example where the true target and outlier distributions are known. The target distribution is the `Gauss` distribution with unit variance ($\sigma = 1$) and the outliers are uniformly distributed. Both distributions are shown in the left subplot. The prior probabilities are chosen $p(\omega_T) = 0.85 = 1 - p(\omega_O)$ (so 15% of the data are outliers). In the middle subplot the posterior probability $p(\omega_T|\mathbf{x})$ is shown, computed by the Bayes rule (5.14) using both known probabilities. The posterior probability becomes a more 'blocked' version of the target distribution, with sharper boundaries between the target and outlier classes. The decision boundary is put at $p(\omega_T|\mathbf{x}) = 0.5$.

In the rightmost subplot the output of a one-class classifier is shown. The 2σ boundary of the Gaussian distribution is shown, thus accepting about 95% of the target data. For this low prior probability of outlier objects, the boundary of the data tends to be underestimated by $p(\mathbf{x}|\omega_T)$ (i.e. the boundary is more tight than the optimal Bayes solution[3]). In this

[3] Of course this is an unfair comparison. The Bayes rule uses the true probability distributions of both the target and outlier data and thus finds the optimum boundary for the problem. The one-class classifier can never beat that.

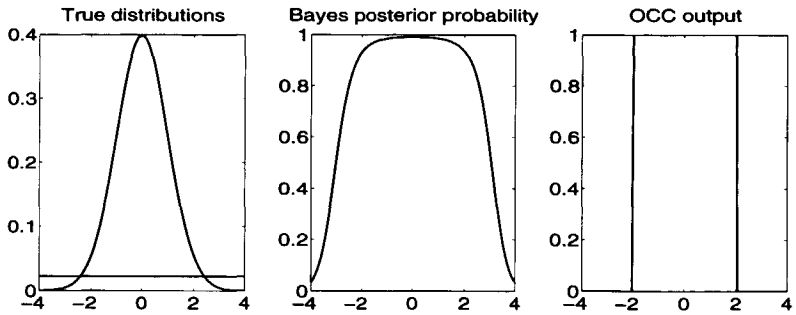


Fig. 5.2: Example of the posterior probability using Bayes rule (middle) when the true target and outlier distributions are known (the Gaussian shaped distribution and the uniform distribution respectively in the left subplot), and the results of a one-class classifier (right).

example the optimal boundary is located at about 3σ, which results in a target rejection rate of 0.3%. For limited number of target examples and for low outlier density, the boundary is located far away in the tails of the distribution. When the boundaries have to be estimated with some confidence, a higher target rejection rate should be used.

Note that when the prior probability of outliers increases, the boundaries of the Bayes rule will also tighten. The fact that the outlier prior probability (and distribution) is not known, forces the use of just $p(\mathbf{x}|\omega_T)$.

5.2.1 Combining rules

When one-class classifiers are to be combined based on posterior probabilities, an estimate for $p(\omega_T|\mathbf{x})$ has to be used. For all types of one-class classifiers (density estimators, boundary estimators and reconstruction models) estimates for the chances of accepting and rejecting target objects, $p(\text{accepting}\,\mathbf{x}|\omega_T)$ and $p(\text{rejecting}\,\mathbf{x}|\omega_T)$, are available. The $p(\omega_T|\mathbf{x})$ is approximated by just two values, f_{T+} and $1 - f_{T+}$. The binary outputs of the one-class methods can be replaced by these probabilities. When just this binary output is used (accept or reject) the different one-class methods can only be combined by a form of majority voting.

From the previous section we know that by assuming the uniform distribution for the outlier objects, $p(\mathbf{x}|\omega_T)$ can be used as an estimate of $p(\omega_T|\mathbf{x})$. For methods which estimate a distance $d(\mathbf{x}|\omega_T)$, this $p(\mathbf{x}|\omega_T)$ is not available and the distance should be transformed in to a probability. Therefore, some heuristic mapping has to be applied. Two possible transformations are:

$$\tilde{P}(\mathbf{x}|\omega_T) = \begin{cases} \frac{1}{c_1}\left(c_2 - d(\mathbf{x}|\omega_T)\right) & \text{for} \quad d(\mathbf{x}|\omega_T) < c_2, \\ 0 & \text{for} \quad d(\mathbf{x}|\omega_T) \geq c_2. \end{cases} \tag{5.18}$$

(which resembles the 'tent-shaped' or lambda-type fuzzy membership function [Zadeh, 1965]) or

$$\tilde{P}(\mathbf{x}|\omega_T) = \frac{1}{c_1} \exp\left(-d(\mathbf{x}|\omega_T)/c_2\right) \tag{5.19}$$

(which models a Gaussian distribution around the model if $d(\mathbf{x}|\omega_T)$ is a squared Euclidean distance). The parameters c_1 (normalization constant) and c_2 (scale parameter) can be fitted to the distribution of $d(\mathbf{x}|\omega_T)$ of the training (target) objects. In both definitions the probability estimate becomes maximal when the distance d decreases to zero, while for very large distances the estimates drop to zero.

For R one-class classifiers, with estimated probabilities $P(\mathbf{x}_k|\omega_T)$ and threshold θ_k, this results in the following set of combining rules:

1. First the *mean vote*, which combines the binary (0-1) output labels:

$$y_{mv}(\mathbf{x}) = \frac{1}{R} \sum_k I\left(P_k(\mathbf{x}|\omega_T) \geq \theta_k\right) \tag{5.20}$$

   Here, $I(\cdot)$ is the indicator function. Of course, when one of the heuristic methods for computing a probability $P_k(\mathbf{x}|\omega_T)$ from a distance $d(\mathbf{x}|\omega_T)$ is used (formula (5.18) or (5.19)), the original threshold for the method should also be mapped. When a threshold of 0.5 is applied to $y_{mv}(\mathbf{x})$, this rule becomes a majority vote in a 2-class problem.

2. The second combining rule is the *mean weighted vote*, where the weighting by f_{Tk} and $1 - f_{Tk}$ is introduced. Here f_{Tk} is the (often predefined) fraction of the target class that is accepted by method k.

$$y_{mwv}(\mathbf{x}) = \frac{1}{R} \sum_k \left(f_{T,k} I(P_k(\mathbf{x}|\omega_T) \geq \theta_k) + (1 - f_{T,k}) I(P_k(\mathbf{x}|\omega_T) < \theta_k)\right) \tag{5.21}$$

   This is a smoothed version of the mean vote, (5.20), but it gives identical results when a threshold of 0.5 is applied.

3. The third combining rule is the *product of the weighted votes*:

$$y_{pwv}(\mathbf{x}) = \frac{\prod_k f_{T,k} I(P_k(\mathbf{x}|\omega_T) \geq \theta_k)}{\prod_k f_{T,k} I(P_k(\mathbf{x}|\omega_T) \geq \theta_k) + \prod_k (1 - f_{T,k}) I(P_k(\mathbf{x}|\omega_T) < \theta_k)} \tag{5.22}$$

4. The fourth combining rule is the *mean of the estimated probabilities*:

$$y_{mp}(\mathbf{x}) = \frac{1}{R} \sum_k P_k(\mathbf{x}|\omega_T) \tag{5.23}$$

5. and, finally, the fifth combining rule is the *product combination of the estimated probabilities*:

$$y_{pp}(\mathbf{x}) = \frac{\prod_k P_k(\mathbf{x}|\omega_T)}{\prod_k P_k(\mathbf{x}|\omega_T) + \prod_k \theta_k} \tag{5.24}$$

where we have used the approximation that $P_k(\mathbf{x}|\omega_O) = \theta_k$. This means that the outlier object distribution is independent of $\mathbf{x}$ and thus uniform in the area of feature space we are interested in.

The output $y(\mathbf{x})$ of the combining rule can now be treated as a standard one-class classifier output. The accuracy on a testing set can be estimated by computing the ROC curve and integrating the error under the curve (error (3.8)). All these combining rules will be compared in a real world one-class problem in the next section.

5.2.2 Combining experiments

To investigate the performance of the different combining rules, a classification problem with several feature sets is used. It is the handwritten digit recognition problem from section 4.13 (or appendix C), for which the following features are available: digits, Zernike, Fourier and profile. For the one-class combining problem one class of handwritten digits is described by one or more data descriptions and distinguished from all other classes. For training, 100 objects are drawn from the target class (no negative examples are used) and for testing 100 objects per class are used (which thus gives a total of 900 outlier objects). In appendix C the integrated ROC errors of all the *individual* one-class methods on all datasets are shown. The results on class '0' are also shown on page 113.

In the combining experiments, 11 of the methods discussed in chapters 3 and 4 are used. These are the Gaussian model, mixture of Gaussians, Parzen density, NN-d, SVDD, LVQ, k-means, k-centers, auto-encoder network, PCA and SOM. The one-class classifiers are optimized for one threshold value, all methods have to reject 10% of the target set. The outputs of the methods are combined using the combining rules mentioned in the previous section. For some of the methods their resemblance measure had to be mapped to a probability estimate. These methods are the NN-d, SVDD, the LVQ, k-means, k-centers and SOM. In all these one-class classifiers a distance larger than zero can be defined (most often just the squared Euclidean distance to the nearest prototype), therefore the transformation (5.19) was chosen. Parameter c_1 was not optimized ($c_1 = 1$), but parameter c_2 was set such that the average $\frac{1}{N}\sum_i d(\mathbf{x}_i|\omega_T)/c_2 = 1.0$.

The ROC curves of the 5 combining rules are shown in figure 5.3 for the target classes '0' (left subplot) and '1' (left subplot). The ROC curves of the 5 combining rules are shown. Because the individual one-class methods are optimized for one specific f_T, the performance of the classifiers is given as a single point in the graph (marked by the stars). Most methods show somewhat higher target rejection rates on the testing set. The ROC curves make clear that most combining rules perform an averaging over the individual classifiers. Only the product combination rule on the estimated probabilities improves

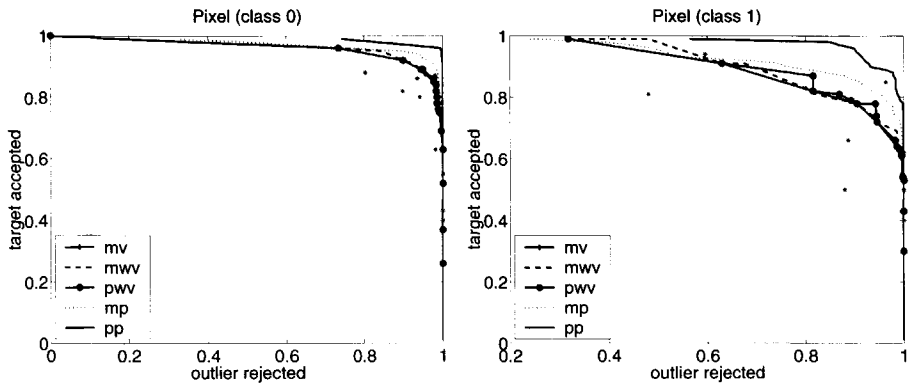


Fig. 5.3: The ROC curves of the 5 combining rules on 11 one-class classifiers. The performance of the individual one-class classifier is given by the stars. In the left plot class '0' of the pixel data is the target class, in the right plot, it is class '1'. The legend uses the abbreviations of the combining rules from defitions (5.20) to (5.24).

over all the individual classifiers and produces an ROC curve which exceeds the other classifiers (this is especially clear in the accuracies on class '1').

Table 5.2: Integrated ROC errors (×100) for the 5 combining rules, applied to the one-class classifiers trained on the Zernike dataset. Numbers in bold indicate that this performance is an improvement over the best individual performance by any of the classifier.

combining	target class nr.									
method	0	1	2	3	4	5	6	7	8	9
mv	**0.01**	**0.21**	**0.55**	6.69	0.89	**1.51**	**3.69**	0.49	**0.01**	4.16
mwv	**0.01**	**0.20**	**0.55**	6.71	0.87	**1.49**	**3.67**	0.45	**0.01**	4.12
pwv	**0.01**	**0.21**	**0.54**	6.58	0.89	**1.50**	**3.67**	0.47	**0.01**	4.15
mp	**0.01**	**0.16**	**0.59**	5.66	**0.65**	1.72	**3.77**	**0.22**	**0.00**	4.22
pp	**0.00**	**0.09**	**0.40**	4.45	**0.48**	**1.01**	**3.83**	**0.16**	**0.00**	4.13

The combining results on all one-class classifiers for all 4 datasets are shown in tables 5.2 to 5.5. We start the discussion with the integrated ROC errors (see (3.7)) for the Zernike dataset are shown in table 5.2. Again the integration boundaries are from $f_{T+} = 0.5$ up to $f_{T+} = 0.95$. Numbers in bold indicate that the performance is improved over the best individual one-class classifier. In the Zernike dataset, the accuracy on almost all classes improves. Only for target class '3' and '9' the combining rules do not improve over the best individual performance. Furthermore, the performance of the mean and product of

Table 5.3: Integrated ROC errors (×100) for the 5 combining rule, applied to the one-class classifiers trained on the Fourier dataset. Numbers in bold indicate that this performance is an improvement over the best individual performance by any of the classifier.

combining	target class nr.									
method	0	1	2	3	4	5	6	7	8	9
mv	0.30	8.82	4.43	19.80	28.95	28.37	23.09	4.45	**0.72**	26.76
mwv	0.30	8.72	4.28	19.25	28.33	27.80	22.94	4.23	**0.72**	26.34
pwv	0.30	8.76	4.27	19.73	28.67	28.12	23.07	3.76	**0.70**	26.68
mp	**0.00**	6.85	2.33	15.86	23.12	**22.05**	20.89	2.49	45.00	24.85
pp	**0.00**	5.83	1.28	9.94	17.34	**13.88**	**13.47**	**1.30**	45.00	17.02

the estimated probabilities (rules 'mp' and 'pp') are often better than the first three rules, indicating that the estimated probabilities contain valuable information for combining.

In the combination of the methods on the Fourier dataset (see table 5.3) the performance improvements are much smaller than in the Zernike dataset. Combining the different classifiers is only useful for classes '0', '5' and '8' (and perhaps classes '6' and '7'). This may be explained by the fact that in this dataset, in general, the performance is quite poor. For class '5', for instance, the best individual performance is 26.53. Therefore, by combining these noisy conclusions not much can be gained. Reasonable results might be expected from classes '0', '2' and '7'. In this dataset it is even more clear that just the mean and the product of the estimated probabilities improves the performance. The exception here is class '8', where several probability estimates become 0. These estimates ruin all good estimates from other methods.

Table 5.4: Integrated ROC errors (×100) for the 5 combining methods, combining the one-class classifiers trained on the profile dataset. Numbers in bold indicate that this performance is an improvement over the best individual performance by any of the classifier.

combining	target class nr.									
method	0	1	2	3	4	5	6	7	8	9
mv	**0.07**	1.42	0.10	**0.40**	0.28	1.86	0.47	**0.02**	1.20	0.39
mwv	**0.06**	1.59	0.09	**0.40**	0.27	1.88	0.33	**0.02**	1.15	0.37
pwv	**0.07**	1.37	0.10	**0.39**	0.28	1.70	0.44	**0.01**	1.14	0.37
mp	**0.05**	0.37	0.09	**0.27**	0.17	1.00	0.42	**0.00**	0.87	**0.19**
pp	**0.07**	**0.09**	**0.06**	**0.25**	**0.12**	**0.65**	0.12	**0.03**	0.85	0.21

The results on the profile dataset (table 5.4) show similar characteristics as in the case of the Fourier dataset. For some classes all combining rules improve their performance, for

Table 5.5: Integrated ROC errors (×100) for the 5 combining methods, combining the one-class classifiers trained on the Pixel dataset. Numbers in bold indicate that this performance is an improvement over the best individual performance by any of the classifier.

combining	target class nr.									
method	0	1	2	3	4	5	6	7	8	9
mv	1.08	5.61	0.94	6.29	2.13	5.80	3.57	0.10	8.69	2.47
mwv	0.96	5.14	0.88	5.97	1.76	5.37	3.23	0.11	8.37	2.11
pwv	1.05	4.94	0.83	6.15	1.79	5.67	3.13	0.07	8.57	2.31
mp	0.35	3.11	0.48	4.70	0.63	1.83	1.69	0.03	4.39	1.08
pp	**0.01**	0.62	0.15	1.99	0.11	**0.28**	0.17	**0.01**	1.97	0.31

some classes only the mean or product rules on the estimated probabilities are useful, and for some classes no improvement can be observed.

In the case of the pixel dataset, hardly any improvement can be obtained (see table 5.5). What was already apparent in the ROC curves of figure 5.3 for classes '0' and '1', it is also noticeable in the performances: the product combination rule (5.24) results in the biggest improvement. In most cases the other combining rules are also useful (they achieve higher accuracies), only in classes '3', '4', '7' and '9' individual classifiers may perform better.

The surprising fact is that the product combination over the posterior probabilities works well. In many cases the posterior probability is estimated very poorly (especially keeping in mind that in many cases the distance measure $d(\mathbf{x})$ has to be mapped to a probability $p(\mathbf{x})$). Combining these poor probability measures by averaging does not lead to improvements, but the more noise sensitive multiplication does. Only by the fact that the one-class classification methods make highly uncorrelated errors, is it possible that the performance increases. The results for the target data give very small output values (for class '1' in the profile dataset, it is on the order of 10^{-9}), but the output for the outlier classes are even smaller (on the order of 10^{-31}).

The results in this section show, that combining different classifiers trained on the same data, does not always result in better performances. In some cases one individual classifier already achieves very high accuracy, and this cannot be improved by combining. When the classifiers are combined, the best combining rules are the mean or the product combination of the estimated probabilities. Using the mean vote, mean weighted vote or the product of weighted votes, approximates the real valued outputs of the individual classifiers for the combining by binary values. The results in this section show that by this approximation procedure useful information for the final classification is lost.

5.2.3 Combining different feature sets

The second experiment concerns the combination of one-class classifiers which are trained on different feature sets. On the 4 sets: profile, Fourier, pixel and Zernike, one-class classifiers of the same type are trained and the outputs are combined again with the same combining rules. It is expected that the extra information contained in the other dataset will improve the final performance. Again the ROC curves are calculated and the integrated error for the classifiers on all datasets are shown in table C.11.

Table 5.6: Integrated ROC errors ($\times 100$) for the 5 combining methods over the 4 feature sets for class '0' as target set. Numbers in bold indicate that the performance is better than that of the best individual classifier.

combining	method nr.										
method	nor	MoG	Par	NN	SVD	LVQ	k-m	k-c	aut	PCA	SOM
mv	**0.00**	**0.01**	**0.00**	**0.00**	**0.00**	**0.00**	**0.01**	**0.00**	**0.00**	**0.00**	**0.00**
mwv	**0.00**	**0.01**	**0.00**	**0.00**	**0.00**	**0.00**	**0.01**	**0.00**	**0.00**	**0.00**	**0.00**
pwv	**0.00**	**0.01**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
mp	**0.01**	**0.03**	**0.00**	**0.02**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
pp	**0.00**	**0.00**	**0.00**	**0.01**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**

The results of combining the classifiers for class '0' is shown in table 5.6. It appears that, using all 4 feature sets, it is possible to separate this target class from all other classes almost without errors. In most cases and for all possible one-class classifiers, the error almost completely vanishes on the independent test set. Note that the best performance of the individual classifiers on class '0' is 0.04 using the Parzen density on the Fourier dataset. This already indicates that very good performances can be obtained for this target class.

In table 5.7 the same results are shown for class '1'. Here the best performance of the individual classifier is 1.07 (again the Parzen density on the profile dataset). Now only some combining rules and one-class methods match and give higher accuracies than the best individual classifier. In particular; the SVDD reach low accuracies with all combining rules. It appears that it extrapolates so poorly for high f_{T+} that even by combining the performance does not improve significantly. Density estimators work well in this problem, which is caused by the fact that the training set is a good sample from the true target distribution. For the combination of the Gaussian model and the auto-encoder network small errors are achieved. Also in these experiments, we see different behavior for the first three (voting type) rules and the last two (probability type) combining rules.

The other combining results are shown in table C.11 (page 167). For the combining of class '2' (best 0.98) and '3' (best 1.63) combining rules almost always improve performance. Only combining the NN-d and the SVDD can give larger errors.

Table 5.7: Integrated ROC errors (×100) for the 5 combining methods over the 4 feature sets for class '1' as target set. Numbers in bold indicate that the performance is better than that of the best individual classifier.

combining	method nr.										
method	nor	MoG	Par	NN	SVD	LVQ	k-m	k-c	aut	PCA	SOM
mv	**0.04**	**0.28**	0.05	3.40	**5.79**	**4.13**	0.63	0.35	**0.04**	**0.25**	**0.33**
mwv	**0.04**	**0.28**	0.05	3.32	**5.78**	**3.98**	0.64	0.34	**0.04**	**0.25**	**0.33**
pwv	**0.04**	**0.38**	**0.04**	3.39	**5.60**	**3.46**	**0.38**	**0.20**	**0.05**	**0.25**	0.57
mp	**0.09**	**0.21**	0.15	**0.50**	**2.17**	**0.28**	**0.01**	**0.01**	**0.04**	**0.06**	**0.00**
pp	**0.00**	**0.01**	**0.00**	**0.57**	**2.17**	**0.28**	**0.01**	**0.01**	**0.00**	**0.00**	**0.00**

5.3 Combining to include prior knowledge

In the previous sections the results of the one-class classifiers are combined to improve the overall performance of the classification problem. It is assumed that in each feature set the probability $P(\omega_T|\mathbf{x}^k)$ for the target set is well defined. When several separate feature sets are available, not only the extra information $P(\omega_T|\mathbf{x}^k)$ can be used to improve performance. The fact that independent $P(\omega_T|\mathbf{x}^k)$ are available makes it possible to *exclude* (or suppress the influence of) a certain $P(\omega_T|\mathbf{x}^k)$.

Let us consider an adapted version of the example from chapter 1 where only apples should be distinguished from all other possible objects. Assume that from all objects two types of features are measured, such that the objects are characterized in, for instance, a 'flavor' and a 'shape' feature space. Then a probability density should be modeled in both feature spaces. An apple should then both smell like an apple and look like one. There are situations, however, where only one feature is measured reliably. It might happen that there is some mud on the apple, which preserves the shape characterization, but might destroy the flavor. Or the apple might have some leaves which do not disturb the flavor, but harm the shape of the object. Using appropriate combining rules can then make the classification more robust.

The average and multiplication rules can be used for this purpose. The mean and product combination rules can be interpreted as being an OR combination and an AND combination, respectively. When it is required that the new object is accepted by all one-class classifiers, a product combination rule gives high output when all classifiers agree that the object is acceptable, but immediately gives low output when one classifier disagrees. In a mean combination all outputs are averaged and one poor output of one of the classifier can be counterbalanced by another high output. This results in an OR combination. By using the mean combination rule, a muddy apple can still be acceptable. If the user wants to reject objects like that, then he[4] should apply the product combination rule.

[4] or she

5.3.1 Image database retrieval

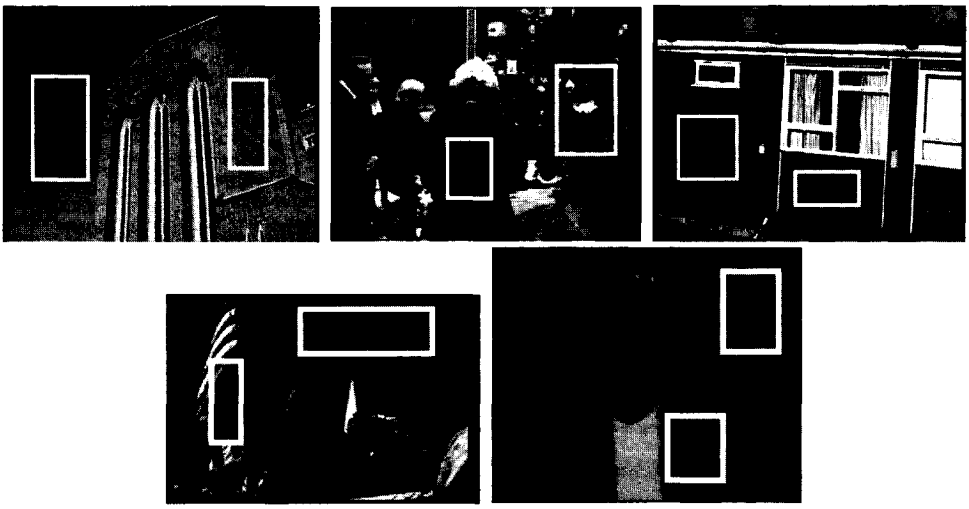


Fig. 5.4: The five defined queries, Cathedral, Queen, Hut, Flag, Newsreader, each with two user-defined regions.

To investigate the use of these rules in a real application, the SVDD will be applied to an image database retrieval problem. In this application the user specifies some interesting and desired image regions, and the application should retrieve the resembling images (images with the same characteristics as in the user-defined regions) from the database. Five test queries are shown in figure 5.4, each with 2 user-selected regions.

In the literature it is well known that color features perform very well in distinguishing the target images from the rest of the database [Antani et al., 1998]. Some very advanced techniques for image database retrieval have been employed to use both color and texture features. A well optimized retrieval procedure, including automatic feature selection and extraction is also found in [Messer, 1999]. In the latter case also (random) images from the database are used as negative examples.

We will use the image database from Messer [Messer, 1999], which contains 3483 images, ranging from captured television images to images from the MPEG-7 test set (the 5 test queries defined in figure 5.4 are already defined in [Messer, 1999]). For each image in this database 33 color and texture features are computed, thus each image can be represented as a large cloud of objects (pixels) in a 33-dimensional feature space. The set of pixels from the user-selected regions are now the target set. To perform a query, a one-class classifier should be trained on these pixels and all other pixels of all other images should be classified by the classifier. The fraction of pixels from a certain image that is accepted, then gives an indication how well that image resembles the user-selected regions in the query image. We will rank all images in the database based on the resemblance between the image and

the user-selected regions (thus the image with rank 1 fits best). There are several ways to rank the images, we will discuss them in sections 5.3.3 and 5.3.4.

To test the performance of the one-class classifier, 5 to 8 other target images, which resemble the target images, are defined beforehand. For a successful retrieval, these target images should appear high in the final image ranking (preferably within the first 100 images). We will discuss this in more detail in section 5.3.2.

To accelerate the query, the color and texture features of the image are calculated beforehand. For each pixel in each image 33 features are computed. They can be separated in 7 feature sets, 3 texture feature sets (Discrete Cosine Transform, Gabor, wavelet) and 4 color feature sets (energy, entropy, mean, variance). The features are listed in table 5.8. The values in the rightmost column are the average standard deviations of the different features sets. The values are measured on the second user-selected region in the first image from figure 5.4. These values show that the different features have significantly different characteristics, and it might be profitable to train 7 separate one-class classifiers in each of these feature spaces. We will consider this in section 5.3.4.

Table 5.8: List of the 33 features, divided in 7 feature sets. The last column gives the average standard deviation of the data set obtained from the brick wall region from the Cathedral image. The large differences show that the data characteristics significantly differ in the different feature sets.

feature set	type of features	dimensionality	σ
1	discrete cosine transform	9	$1 \cdot 10^4$
2	Gabor filters	8	$2 \cdot 10^3$
3	energy (green/int/red)	3	$3 \cdot 10^{-2}$
4	entropy (green/int/red)	3	$6 \cdot 10^{-2}$
5	mean (green/int/red)	3	$2 \cdot 10^0$
6	variance (green/int/red)	3	$6 \cdot 10^2$
7	wavelet transform	4	$1 \cdot 10^3$

To reduce the training time, from the user-selected image regions just a subset of 200 pixels is randomly drawn, and on this set of objects a one-class classifier is trained. Here, the SVDD is used. Using larger training sets sometimes gives somewhat better results, but requires long training times. In all experiments, the SVDD is trained on the user-defined regions such that about 10% of the training pixels are expected to fall outside the description (estimated by (2.47)).

The evaluation of all pixels in all images from the database is very time-consuming. To reduce the computational burden in the evaluation of the complete image database, all images are segmented first by clustering the set of pixels in the 33-dimensional feature space. On average about 30 to 50 cluster centers are obtained for each of the images. These cluster centers will be called indices. An example of a clustered image is given in figure 5.5. This reduces the size of the representation of the images from about 300 × 200 pixels

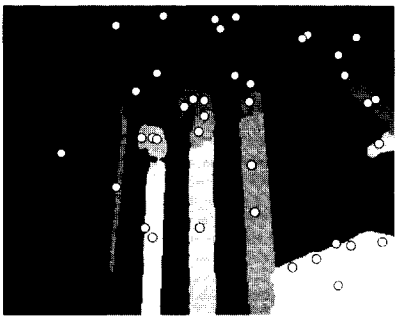


Fig. 5.5: Clustered (or segmented) Cathedral image, containing 38 clusters. The cluster centers (indices) and their approximate positions are given by the dots.

to about 50 indices. The drawback is that this only considers an average of the cluster and the variance structure of the cluster is lost.

5.3.2 Evaluation performance

Thus, on a set of pixels drawn from the user-selected regions, a SVDD is trained. All indices of all images are classified by this SVDD and the images are ranked using a ranking measure $\mathcal{E}_{\text{rank}}$ (definitions of ranking measures will be given in sections 5.3.3 and 5.3.4). To judge how well the 5 (or 8) test images are retrieved, a retrieval error $\mathcal{E}_{\text{ret}}$ should be defined. The retrieval error is defined as the chance that a random method (a method which just randomly ranks the images) shows the same or a better ranking as the method under investigation. With a better ranking we mean a ranking in which the average rank of the test images is lower (more images are ranked at the top).

The exact computation of the chance that a certain average rank is obtained by a random method, is expensive, even if we avoid the discrete nature of the rankings. Assume in the continuous case we have n ranks $r_i, i = 1, \ldots, n$ uniformly distributed between 0 and 1.[5] Using the central limit theorem we approximate the distribution of the average ranking $\overline{m} = \frac{1}{n}\sum_{i=1}^{n} r_i$ by a normal distribution with the mean of the uniform distribution $\mu = \mu_{\text{unif}} = \frac{1}{2}$ and the variance of $\sigma^2 = \frac{1}{n}\sigma^2_{\text{unif}} = \frac{1}{12n}$. The chance of finding an average

[5] When we have n independent uniform random variables, the rankings r_i, between 0 and 1, and we order the random variables such that $r_1 < r_2 < \cdots < r_n$, the k^{th} variable is distributed as [Grimmett and Stirzaker, 1982]:

$$f_k(r) = n\binom{n-1}{k-1} r^{k-1}(1-r)^{n-k} \qquad \text{for } r \text{ between 0 and 1} \tag{5.25}$$

The joint probability for n variables becomes $f(r_1, r_2, \ldots, r_n) = f_1(r_1)f_2(r_2)\ldots f_n(r_n)$. This is the distribution of the set of n values. We are interested in any set equal or *better* than this. A better set is defined as one having an average ranking higher at the top. Then, the mean rank $\overline{m} = \frac{1}{n}\sum_{i=1}^{n} r_i$ is distributed

rank $\overline{m}$ or better, is than given by $\int_0^{\overline{m}} \mathcal{N}(m; \frac{1}{2}, \frac{1}{12n})dm$.

In the case of the image database, we have n images ranked between 1 and M. The distribution of *average rank* m of the n images will be distributed as:

$$p(\overline{m}) = \mathcal{N}\left(\overline{m}; \mu = \frac{M+1}{2}, \sigma^2 = \frac{(M-1)^2}{12n}\right) \tag{5.27}$$

where $\mathcal{N}(\mathbf{x}; \mu, \sigma^2)$ is the Gaussian distribution [Ullman, 1978]. In general, the average ranking obtained by a non-random method will be better. Integration over m up to this average rank $\overline{m}$ of (5.27) gives an indication how (un-)likely the results would be in the case of a random method:

$$\mathcal{E}_{\text{ret}}(\overline{m}) = \int_{-\infty}^{\overline{m}} p(m)dm \tag{5.28}$$

To get a feeling for what this means, assume that we have a database containing $M = 10$ images and a one-class classifiers ranks the $n = 3$ target images on ranks $1, 2$ and 5. This gives $\overline{m} = 2.67$, and results in $\mathcal{E} = 0.029$ by formula (5.28). A more realistic image database will contain $M = 3500$ images, and assume that a query with $n = 5$ target images is defined. When the images are ranked $(1, 2, 3, 4, 5)$, $\overline{m} = 3$, the error becomes $\mathcal{E}_{\text{ret}} = 5.09 \cdot 10^{-15}$, while for ranking $(1500, 1550, 1600, 1650, 1700)$ $\mathcal{E}_{\text{ret}} = 0.2526$. To avoid printing the huge differences in orders, the results of the experiments will be shown in a more readable format as $\log_{10}(\mathcal{E}_{\text{ret}})$, giving -14.29 and -0.59 respectively.

Although this measure does not give an exact estimate of the performance of a one-class classifier with respect to the random method, it does give a possibility to compare different classifiers. This measure is therefore only used to give an impression of the relative performance. Is not used to give precise, absolute errors of each of the individual classifiers.

5.3.3 Original features

First a SVDD is trained on the original 33 features, without preprocessing, to give an indication of difficulty of the problem. In table 5.9 four definitions of the ranking measures $\mathcal{E}_{\text{rank}}$ are shown. First, it is determined which indices are accepted by the SVDD. Second, the accepted indices are ranked according to distance to center of the SVDD, the region size of the index or a combination of both. Finally, these values are summed (or multiplied in case of ranking measure 4) over all indices to obtain a final score or rank for the image.

---

as:

$$f(\overline{m}) = \int_{r_1} \int_{r_2} \cdots \int_{r_n} f(r_1, r_2, \ldots, r_n) I\left(\frac{1}{n}\sum_{j=1}^{n} r_j \leq \overline{m}\right) dr_1 dr_2 \ldots dr_n \tag{5.26}$$

This is very hard to compute, and we therefore use the central limit theorem which states that $\overline{m}$ is approximately normally distributed when n is large enough [Weisstein, 1998] (all r_i's originate from the same uniform distribution). The integration in (5.26) is than replaced by an integration over a normal distribution.

Table 5.9: Ranking measures $\mathcal{E}_{\text{rank}}$ defined for queries where all features are used in one SVDD.

ranking measure	error
1	sum over region sizes of accepted indices
2	sum over distances from accepted indices to centers
3	sum over distance × (1+region size)
4	product over region sizes

Table 5.11 (page 139) shows the ranking performances for the 5 query images, Cathedral, Queen, Hut, Flag and Newsreader, on the complete image database for all retrieval methods used in this chapter. In the third and fourth column of table 5.11 (page 139) the results are shown for the case in which the SVDD is trained on all features. Because the database consists of almost 3500 images, a worst case ranking of, for instance, 3200 means that the SVDD on that image is far worse than random guessing which would give a ranking around 1700. When one image out of 5 testing images is ranked like this, an ranking measure on the order of -1.0 to -2.0 occurs (using formula (5.28)), depending on how the other testing images are ranked. The results show that the four different ranking measures $\mathcal{E}_{\text{rank}}$ (table 5.9) give about the same ranking for the images. Results on the Cathedral, Queen and Newsreader queries are acceptable. Although it appears that the desired images are not within the first 100, the SVDD clearly performs better than random guessing. The results on the Hut and Flag queries on the other hand are disappointing. It appears that in the Hut query only the training image is recognized well (it is ranked $91, 4, 5, 43$ in the four ranking measures). All other images are either ranked very low or are completely rejected.

In the Flag query, most of the desired images are ranked very high. Only two images are not retrieved well. These images are shown in figure 5.6. These images have been rotated over 90 degrees. This transformation is not considered irrelevant (the texture features are not rotation invariant), and therefore the images are rejected.

Fig. 5.6: Images with very poor ranking in the Flag query

The last column in table 5.8 shows the standard deviations of the data extracted from

the wall region from the Cathedral image. Clearly, the large difference in scale of the features can deteriorate the performance of the SVDD. Therefore, the data is rescaled to unit variance in the target set and the experiments are repeated. The results are shown in the 4th column in table 5.11. The results on the Cathedral, Queen and Newsreader queries are very good. When the second worst rank is considered in the Queen query, all ranks are lower than 32. Hut and Flag suffer from the same problems as before. Even more so, the only image in the Hut query which is accepted by the SVDD is the original target image: *all* other images are rejected.

5.3.4 Separating the feature sets

To improve performance, especially in the Flag query, separate SVDD's are trained on the individual feature sets (see table 5.8). It is hoped that it will not only make training of a SVDD easier and more efficient, but it will also give the possibility to use several feature combination rules. In the Flag query this means that the one-class classifier can be forced to use the texture information.

Table 5.10: Ranking measures $\mathcal{E}_{\text{rank}}$ defined for image ranking in case of SVDD on separate feature sets.

ranking measure	distance to center	region size	weighted by training perform.	feature sets	indices
1	X			sum	sum
2		X		sum	sum
3			X	sum	sum
4	X			prod	sum
5		X		prod	sum
6			X	prod	sum

Table 5.10 shows the definitions of the ranking measures in this new problem. The fourth column, called 'weighted by training perform.', indicates that the ranking is also weighted by how well the extracted indices from the training image are accepted by the one-class classifiers. When all indices of the training image are rejected by the SVDD's, the corresponding feature set obtains a low weight. The difference between a sum and a product combination over feature sets is the difference between 'blue sky OR brick wall' and 'blue sky AND brick wall'.

In the fifth and sixth column of table 5.11 the results for the separate SVDD's without and with scaling are shown. The separate feature sets are scaled well and the rescaling does not improve performance very much. The results are somewhat worse for the Cathedral, Queen and Newsreader with respect to the original results. Only when the ranking measure is based on a product over the feature sets (measures 4,5,6), the performance is comparable.

Fig. 5.7: An example of a Hut test image, where one of the panels is partly occluded by an antenna.

For the Hut image the OR-operation over the feature sets performs very well. It appears that in some of the desired images one of the panels is partly obscured by an antenna, thus destroying the texture characterization of the original panel. An example is shown in figure 5.7. By using the OR operation, the influence of the "wrongly" characterized panels can be reduced.

Finally, the Flag query still suffers from the same problems as before. The two desired images, which are rotated 90 degrees, are not ranked very high. Best performance is reached using the AND operation on the feature sets, while only considering the distance to the centers of the SVDD's.

It appeared that in this application the color feature is important and that it is well clustered. Scaling both color and texture features to unit variance does not decrease performance for image queries in which texture features can be ignored. For queries where texture features are more important, the different color and texture features are better separated and treated separately. Combining the descriptions from the different feature sets opens the possibility of applying AND and OR operations on the different features. Then a distinction between 'blue sky AND brick wall' and 'blue sky OR brick wall' can be made. Of course this will require user input.

A normal operation procedure might therefore be, that a SVDD is trained on all 33 features, which are scaled to unit variance. In most cases this gives very acceptable results, especially when queries are focused on color. When the user is not satisfied, data descriptions in separate feature sets should be trained. The user then has to indicate how they should be combined (by AND or OR operations).

5.4 Conclusion

Inspired by the gains in performance which have been obtained when conventional classifiers are combined, we looked at the possibilities of combining one-class classifiers. In the combining of conventional classifiers several combining rules can be considered, depending on the type of output the individual classifiers give. It appears that when the classifiers

Table 5.11: Query results for one SVDD using all features (original and scaled to unit variance) and the SVDD's trained on separate feature sets (also on the original and rescaled sets). The results are presented as logarithmicly scaled retrieval error, $\log_{10}(\mathcal{E}_{\text{ret}}(\overline{m}))$.

	ranking	all features		separate features		outlier objects	
name query	type	no sc.	scaling	no sc.	scaling	no sc.	scaling
Cathedral	1	-13.01	-12.98	-9.23	-12.31	-12.94	-14.23
	2	-13.15	-13.18	-10.57	-7.84	-13.27	-14.23
(5 target	3	-12.87	-12.99	-10.07	-9.88	-12.82	-14.23
images)	4	-13.01	-12.99	-14.03	-13.95	-12.95	-14.23
	5			-14.01	-13.95		
	6			-14.04	-14.11		
Queen	1	-12.31	-11.82	-8.64	-12.17	-13.32	-13.49
	2	-12.46	-11.11	-13.01	-13.47	-12.45	-13.74
(5 target	3	-12.33	-11.05	-6.86	-7.13	-12.47	-13.47
images)	4	-12.84	-11.68	-12.63	-13.96	-13.51	-13.59
	5			-12.24	-13.72		
	6			-12.69	-13.96		
Hut	1	-4.31	-10.01	-10.28	-12.23	-4.99	-6.86
	2	-4.57	-10.04	-2.93	-4.89	-5.10	-6.86
(5 target	3	-4.40	-10.04	-13.23	-13.15	-5.03	-6.86
images)	4	-4.34	-10.02	-8.17	-6.60	-5.02	-6.86
	5			-8.12	-6.57		
	6			-8.22	-6.60		
Flag	1	-4.30	-8.90	-0.09	-7.18	-6.34	-2.15
	2	-2.11	-8.54	-7.10	-15.21	-4.41	-2.14
(8 target	3	-3.04	-8.64	-0.00	-0.00	-5.69	-2.15
images)	4	-4.41	-8.64	-1.99	-20.84	-6.59	-2.15
	5			-2.10	-17.57		
	6			-1.86	-10.07		
Newsreader	1	-10.16	-12.22	-1.24	-13.68	-8.69	-14.05
	2	-13.19	-13.62	-11.70	-12.98	-12.44	-14.27
(5 target	3	-12.15	-13.65	-0.08	-0.37	-11.47	-14.26
images)	4	-10.46	-13.51	-14.27	-14.29	-9.41	-14.13
	5			-14.12	-14.18		
	6			-14.24	-14.27		

estimate class posterior probabilities, the Bayes rule shows that the product combination rule is optimal for classifiers which estimate from independent data. On the other hand, when data is highly correlated, or the posterior probability estimates are very noisy (large errors occur) the mean combination rule is more robust and shows better performance.

In the combination of one-class classifiers, the situation is different. In most cases, only a class label is estimated. It is even impossible to estimate a posterior probability because the density of one of the classes (the outlier class) is unknown. Only when it is assumed that the outliers are uniformly distributed, can we estimate the posterior probability. Furthermore, in some one-class classification methods no probability is estimated, but a distance. When we have a combination of distance and probability outputs, the outputs should be standardized before they can be combined. To use the same type of combining rules as in conventional classification combining, the distance measures must be transformed into a probability measure.

When this is done, we observe that, as in combining conventional classifiers, combining often improves performance, especially when the probability estimates are combined by using the product rule. Surprisingly, this rule still performs well also when the resemblances are mapped to probabilities by a, more or less ad hoc, transformation. Although the product outputs are, in general, very small, significant outputs for the target and outlier outputs can be reported.

The results of combining one-class classifiers confirm what has already been observed in the combining of conventional classifiers, i.e. that combining different feature sets is more useful than combining different classifiers. This is a natural implication of the fact that the different datasets contain more information than the different views of the individual classifiers on one dataset. To be more specific, for the individual one-class classifiers, it appears that although the Parzen density is often the best classifier, combining is often best using a Gaussian model.

Finally, combining rules are not only useful in improving the classification performance over one classifier. Using different combining rules can also give the opportunity to include prior knowledge and user interaction into the classification process. In the application of the image database retrieval, for instance, the choice between the product and mean combination rule opens the possibility of applying AND and OR operations on the different features. In such a case, the user can make a distinction between the queries 'blue sky AND brick wall' and the query 'blue sky OR brick wall'.

6. CONCLUSIONS

In this thesis we have have investigated one-class classifiers. We tried to answer the following questions:

- How can a classifier be constructed, trained and evaluated based on samples from just one of the classes (called the target class)?
- What should be the basis for a decision to classify an object as belonging to the target class?
- How can the resemblance of an object to a set of training objects be defined?
- What should be the measure to minimize the chance of accepting non-target (or outlier) objects?
- How can a one-class classifier be evaluated when no example outlier objects are available?

One-class classification appears when in a conventional classification problem classes in the (training) data are poorly balanced, i.e. one of the classes is severely undersampled due to the measuring costs for that class caused by the low frequency of occurrence. It might also occur that it is completely unclear what the representative distribution of the data is. In that case only the boundary of the data might be known (or approximated by the user), but a complete density cannot be modeled with high confidence. In these situations conventional classification methods should be replaced by one-class classifiers (see the experiments in chapters 2 and 4).

A complication emerges when one-class classifiers have to be evaluated. When only objects from one class are available, only the number of false negatives can be estimated (target objects which are rejected). Unfortunately, the false positive fraction (the fraction of outlier objects which is accepted) cannot be estimated, because negative examples are (per definition for the one-class classification problem) absent or very scarce. To get an estimate for the fraction of false negatives, artificial outliers have to be created. When these artificial outliers are uniformly distributed in and around the one-class classifier, the volume occupied by the one-class classification can be estimated. It is hypothesized that a smaller volume implies a smaller chance of accepting outliers. The total error is then a combination of the classification error on the target set and the volume of the one-class classifier. In practice, the user has to give the tradeoff between these two.

6.1 What has been done in this thesis?

In chapter 2 we presented the derivation of the support vector data description (SVDD), a one-class classification method which avoids a complete density estimation in the feature space. Instead of estimating a complete probability density, it estimates a closed boundary around the data set such that it encloses a volume as small as possible in the feature space. This method is, therefore, eminently suited for solving the one-class problems mentioned above. The boundary of the basic method is given by a hypersphere and it appears that this boundary can be described by a few training objects, the support vectors. Furthermore, in this formulation the feature vectors always form an inner product with another feature vector.

Although the SVDD uses a rigid hypersphere model for the boundary around the data, it has the ability to obtain more flexible data descriptions. This is done by using kernel functions instead of the original inner products. These kernel functions implicitly map the data to another, possibly very high dimensional, feature space. The mapped data in this feature space may fit the hypersphere model better than the original data. Two important kernel functions have been investigated, the polynomial and the Gaussian kernels. In contrast to the support vector classifier, the support vector data description with a polynomial kernel suffers from the influence of large norms of the object vectors, but it shows promising results for the Gaussian kernel. Using the Gaussian kernel, descriptions comparable to the hyperplane solution of Schölkopf et al. [Schölkopf et al., 1999] are obtained.

When it is assumed that the training set is a representative sample from the true target distribution,[1] the fraction of the target objects which become support vectors is an estimate of the error which the description will make on the target set. When the maximum acceptable error on the target set is known beforehand, the width parameter (which is still a free parameter in the support vector classifier) can be set to give the desired number of support vectors. Because the SVDD cannot extrapolate very well, the performance of the SVDD for high target acceptance rates (low false negative rates) is poor. When the number of training objects is insufficient, the fraction of objects which become support vectors remains high whatever width parameter is used. This is an indication for the SVDD that more data is necessary or that the data dimensionality should be reduced. Extra data in the form of outlier objects can also be used to improve the support vector data description.

In chapters 3 and 4, we discussed the performance of a large set of (relatively simple) one-class classifiers. We distinguished 3 types of one-class classification models: based on density estimation, based on a boundary description and based on data reconstruction. Some methods assume strong models for the data (especially the reconstruction methods) while others rely almost completely on the data (which is especially true for the Parzen density estimator and the NN-d). The influence of these assumptions and the ease of optimization of the methods have been investigated by applying them to a set of artificial and

[1] In this thesis we do not automatically assume that this is the case. In most practical one-class classification problems this will not hold, and therefore the proposed estimates cannot be used.

real world data. The artificial datasets were constructed such that they showed different characteristics concerning sample size, scaling of the features, clustering, convexity and the presence of subspaces. The applicability of the methods to all types of data was measured. The fit of the one-class methods to the data was used to characterize some real world data.

Some conclusions about the one-class classifiers and their fit to the different types of data, can be drawn:

1. For small sample sizes it is, in general, impossible to make a good description of the data. The few objects will contain too much variability to provide a description with high confidence. On the other hand, for high dimensional data and small sample sizes the problem of distinguishing between target objects and outlier objects becomes quite easy for most methods. In these high dimensional feature spaces there is much freedom and space so a solution can easily be found.

2. The Parzen density estimator is often a good choice for describing the data. Because the width parameter in the Parzen density can be found using maximum likelihood optimization, even with small sample sizes reasonable solutions can be reached. This density approach fails when the data is very scarce and poorly scaled, or when the training distribution differs significantly from the true target distribution. A further drawback of the Parzen density estimator is that the evaluation time is considerable.

3. Evaluation time for the NN-d is also considerable, but fortunately this method performs best in small sample size areas where computational efforts are low. For larger sample sizes and noisy data the performance is very poor. Here the nearest neighbor method seriously suffers from remote objects in the training set. For small sample sizes and data distributions with sharp boundaries, performance of the nearest neighbor method is good.

4. The clustering methods, LVQ, k-means and k-centers often have comparable performance. Most often the assumption that the objects from one class are clustered or nearby in feature space (continuity assumption), is satisfied. The only constraint for these methods is that a sufficient number of prototypes is used. When this number is too low, a large bias is introduced and the model does not have enough flexibility to adapt to the data.

5. The support vector data description performs poorly when an insufficient amount of data is available. Otherwise, performance is comparable to the Parzen density estimator. Fortunately, the performance of the SVDD on the training set is a good indication of the performance on a test set, so poor performance is already expected when the error is high on the training set. Furthermore, the performance of the SVDD is especially good when the training distribution is different from the target distribution (where in the training data mainly covers the target class area, but does not model the density). Most methods will overtrain on the training data, but the SVDD is very apt at finding a good boundary. Finally, the evaluation time is

moderate, as the SVDD only depends just on the number of support vectors. In normal cases, this number of support vectors is a small fraction of the number of training objects, and the evaluation time stays small (especially in contrast to the Parzen density estimator).

6. The auto-encoder (and the diabolo) networks require careful training and adjusting and is not very robust. It also demands larger sample sizes than the other one-class classification methods. When enough data is available, and some effort is put into good optimization, very good results can be obtained. When the auto-encoder is expected to be a fully automatic procedure, the results can be very disappointing.

7. Finally, the SOM and the PCA use strong assumptions about the data. For the SOM it is assumed that the data is distributed in a low dimensional manifold (1, 2 or 3-dimensional, depending on the definition of the grid). For the PCA the data should be located in a linear subspace. When the data satisfies these assumptions, good results can be obtained.

In the last chapter we investigated the possibilities of combining one-class classifiers. In conventional classification problems, it often appears that one single classifier is not capable of obtaining the optimal results from the dataset. By combining several classifiers particular weaknesses of individual classifiers can be hidden, and the final performance can be significantly improved. It is hoped that this boost in performance can also be observed when one-class classifiers are combined.

It appears that combining one-class classifiers is more complicated than combining conventional classifiers. Much of the theory of combining conventional classifiers is based on the assumption that the classifiers estimate posterior class probabilities. When each classifier outputs posterior probabilities, it is possible to combine them using the Bayes rule. In combining one-class classifiers, the classifiers do not have to output a probability estimate, but they can also output a distance to a model. To combine these types of classifiers, the distance measure has to be transformed into a type of resemblance. This transformation introduces extra assumptions in the combining procedure.

Fortunately, it appears that combining one-class classifiers often improves performance, like in the case of combining conventional classifiers. Somewhat surprising is the result that combining is especially useful when the probability estimates are combined using the product rule. In conventional classification combining, the product combination rule works best when the error in the probability estimates is limited. In one-class combining, resemblances are mapped to probability estimates by a, more or less ad hoc, transformation which might introduce a significant amount of noise in the estimates. Still, the product combination often gives very good results. The noise introduced by the mapping from resemblance to probability does not seem to harm the final classification results very much.

Finally, we have observed that combining different feature sets is more useful than combining different classifiers. This is to be expected and it is also noticed in the combination of conventional classifiers. Because different feature sets contain more information than different classifiers trained on one dataset, better generalization can be obtained.

In the second part of chapter 5 combining is not merely used to improve performance directly, but to include prior knowledge into the classification problem and to steer the solution. In the image database retrieval problem it was the task to retrieve all images resembling an image which was given by the user. The user could indicate sub-regions in the original image which are important (for instance, a part of a blue sky and a red brick wall). In some cases it turns out to be useful to search for AND and OR combinations of these sub-regions. Then the user can search for 'blue sky' AND 'brick wall' or the user can search for 'blue sky' OR 'brick wall'. This can be done by applying the product or the mean combination rule, respectively.

6.2 What can be concluded from this thesis?

First, the best overall one-class classifier for well sampled data is the Parzen density. Well sampled data means that the sample size is reasonably high (as a rule of thumb, the number of training objects is at least five times the data dimensionality), and that the distribution of the training data is the same as the testing (or 'true') data distribution.

Secondly, the support vector data description performs best when the training distribution is not identical to the testing distribution. It only estimates the boundary of the data. Because the extrapolation ability of the SVDD is poor, the SVDD does not perform well when the sample size is very small. SVDD works fine when a prespecified target acceptance rate is required. Also the evaluation costs are small (although the training costs are not!).

When more is known about the data distribution, a more specific model can be fitted to the data. PCA and SOM are good candidates to use when data is distributed in a subspace, but they require non-trivial sample sizes. For clustered data, methods like k-means, LVQ and k-centers are well-suited.

Finally, combining one-class classifiers is possible, and it is especially useful when classifiers trained on different feature sets are combined. Combining by multiplying the outputs (maybe mapped to probabilities) of the one-class classifiers often shows the best final performance.

In figure 6.1 a very coarse overview is given which shows the applicability of different methods to different situations. For well sampled data it is assumed that the training distribution closely resembles the 'true' distribution and here density estimators work well. For small sample sizes more assumptions have to be made about the data for the method to find a good solution. For very small sample sizes it becomes very hard to find a representative training set. In practice it means that an atypical sample is being used. For atypical data the training dataset is not identically distributed as the 'true' distribution, only the area covered by both is about equal. Here, the boundary methods perform better. For larger sample sizes the SVDD works best, while for very low sample sizes the nearest neighbor method is to be preferred.

In summary, no final winner for the one-class classification problem can be given, it depends too much on the task at hand. It depends then not only on the scaling of the features, preprocessing, the sample size in comparison with the dimensionality, but also on

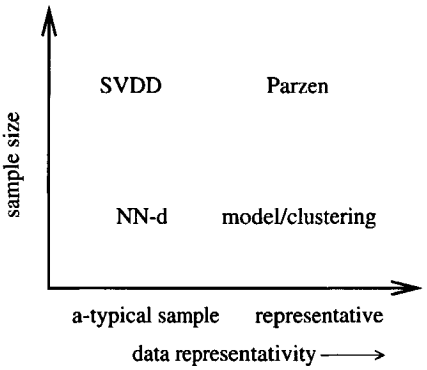


Fig. 6.1: Different one-class classification methods for different sample sizes and for both representative training sets and atypical training samples.

time and computation constraints, and what target rejection rate is acceptable. Therefore, all such constraints should be taken into account, before choosing the best suitable method.

6.3 What can be done in the future?

The problems encountered in one-class classification are most often general problems in pattern recognition. Unfortunately, some of these problems become more prominent in one-class classification. For instance, how can we be sure that the training sample is a good approximation to the 'true' distribution? To measure this, how can we estimate the difference between a training distribution and a testing distribution? This is especially important when an already optimized classifier is applied to a similar classification problem. When we are sure to have a good representative dataset available, the density estimators might perform well enough and the computation of, for example, a support vector data description can be avoided.

Because no example outliers are available, other measures have to be used to estimate the error of the second kind (see section 1.4). In the support vector data description, the volume of the description is used to minimize the chance of accepting outliers. Are there other criteria which can be used? Are there efficient methods for creating outlier objects?

On the other hand, when the volume of the description is used, the scaling of the features becomes important. Different scaling results in different solutions for the data description. Is there a sensible way to obtain a good distance definition for the problem at hand? Can data be rescaled such that the description becomes more simple (preferably a hypersphere solution, which best suits the support vector data description)? Can redundant features be identified (if necessary using example outlier objects)?

Further, focused more on the support vector data description, can this method be "patched" to show some extrapolation ability? In the current solution the boundary of

the description passes right through the support vectors. This means that around the individual support vectors about half of the objects will be rejected. It might be reasonable to assume that the objects in the dataset are not purely given by vectors, but by small (Gaussian) distributions. Can this be incorporated to obtain another data description?

Finally, there is one very substantial problem which we did not discuss at all, but also becomes very important for one-class classification: How can a dynamic data distribution be followed? In the one-class classification methods we discussed in this thesis, all training was done beforehand using a training set (perhaps even a atypical dataset). In a dynamical situation, new objects are measured in time. Starting with none (or a few) example objects, the dataset might evolve through time. Would it be possible to automatically detect from new, incoming objects when the target distribution starts to deviate from the original distribution and adapt the boundary accordingly? Is there a natural extension of the SVDD which can cope with this type of data?

To make the available classification methods more useful and applicable, an attempt should be made to answer these questions. Currently, still a large gap exists between the data obtained in real-life environments (containing noise, missing measurements, outliers and measured from atypical and non-stationary data distributions) and the nice theoretical statistical classifiers (which assume well-defined, well-sampled and stationary data distributions). One-class classifiers is just one of the tools which can help to bridge this gap.

APPENDIX

A. SUPPORT VECTOR DATA DESCRIPTION

In section 2.1 on page 21 the Quadratic Optimization problem is derived. In the derivation of the constraints and the interpretation of the constraints on the free parameters α_i, some extra explanation might be useful.

We start again by defining the error to minimize:

$$\mathcal{E}(R, \mathbf{a}) = R^2 + C \sum_i \xi_i \tag{A.1}$$

where we constrain the solution such that (almost) all objects lie within the hypersphere:

$$\|\mathbf{x}_i - \mathbf{a}\|^2 \leq R^2 + \xi_i, \qquad \xi_i \geq 0, \qquad \forall i \tag{A.2}$$

Constraints (A.2) can be incorporated into formula (A.1) by introducing Lagrange multipliers and constructing the Lagrangian [Strang, 1988]:

$$\begin{aligned} L(R, \mathbf{a}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\gamma}) =& R^2 + C \sum_i \xi_i \\ & - \sum_i \alpha_i \{R^2 + \xi_i - (\mathbf{x}_i \cdot \mathbf{x}_i - 2\mathbf{a} \cdot \mathbf{x}_i + \mathbf{a} \cdot \mathbf{a})\} - \sum_i \gamma_i \xi_i \end{aligned} \tag{A.3}$$

with the Lagrange multipliers $\alpha_i \geq 0$ and $\gamma_i \geq 0$. Note that for each object $\mathbf{x}_i$ a corresponding α_i and γ_i is defined. L has to be minimized with respect to $R, \mathbf{a}, \xi_i$ and maximized with respect to $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$.

Setting partial derivatives to zero gives the constraints:

$$\frac{\partial L}{\partial R} = 0 : \qquad \sum_i \alpha_i = 1 \tag{A.4}$$

$$\frac{\partial L}{\partial \mathbf{a}} = 0 : \qquad \mathbf{a} = \frac{\sum_i \alpha_i \mathbf{x}_i}{\sum_i \alpha_i} = \sum_i \alpha_i \mathbf{x}_i \tag{A.5}$$

$$\frac{\partial L}{\partial \xi_i} = 0 : \qquad \gamma_i = C - \alpha_i, \quad \forall_i \tag{A.6}$$

From the last equation we get that $\alpha_i = C - \gamma_i$. Instead of the constraints that $\gamma_i \geq 0$ and $\gamma_i = C - \alpha_i$, a new constraint on α_i can be introduced as:

$$0 \leq \alpha_i \leq C, \qquad \forall_i \tag{A.7}$$

As long as this constraint is satisfied, we can compute the Lagrange multipliers γ_i by $\gamma_i = C - \alpha_i$ and automatically $\gamma_i \geq 0$ holds.

Rewriting error (A.1) and assuming all constraints are fulfilled:

$$\begin{aligned} L(R, \mathbf{a}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\gamma}) &= R^2 - \sum_i \alpha_i R^2 + C \sum_i \xi_i - \sum_i \alpha_i \xi_i - \sum_i \gamma_i \xi_i \\ &\quad + \sum_i \alpha_i \mathbf{x}_i \cdot \mathbf{x}_i - 2 \sum_i \alpha_i \mathbf{a} \cdot \mathbf{x}_i + \sum_i \alpha_i \mathbf{a} \cdot \mathbf{a} \qquad \text{(A.8)} \\ &= 0 + 0 + \sum_i \alpha_i \mathbf{x}_i \cdot \mathbf{x}_i - 2 \sum_i \alpha_i \sum_j \alpha_j \mathbf{x}_j \cdot \mathbf{x}_i + 1 \cdot \sum_{i,j} \alpha_i \alpha_j \mathbf{x}_i \cdot \mathbf{x}_j \\ &= \sum_i \alpha_i \mathbf{x}_i \cdot \mathbf{x}_i - \sum_{i,j} \alpha_i \alpha_j \mathbf{x}_i \cdot \mathbf{x}_j \qquad \text{(A.9)} \end{aligned}$$

This error function (A.9) with constraints (A.7) presents a well-known quadratic form. Its minimization is an example of a Quadratic Programming problem and standard algorithms exist to solve this.

Error (A.9) is optimized with respect to $\boldsymbol{\alpha}$. Given the optimal values for $\boldsymbol{\alpha}$, the center $\mathbf{a}$ of the hypersphere and the errors ξ_i can be calculated using formulae (A.5) and (A.6). The radius R is defined as the distance from the center $\mathbf{a}$ to the support vectors on the boundary of the hypersphere.

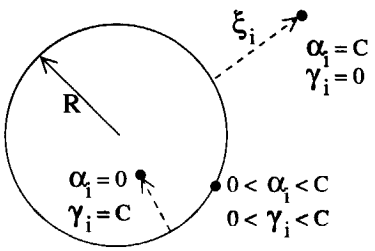


Fig. A.1: The values of the two Lagrange multipliers α_i and γ_i for objects $\mathbf{x}_i$ inside the hypersphere, on the boundary and outside the hypersphere.

By the Lagrange formulation of the problem, further interpretation of the values of $\boldsymbol{\alpha}$ can be given. A characteristic of Lagrange multipliers is, that they only play a role when the corresponding constraint should be enforced or is violated. Only in these cases they become positive (unequal to 0). For instance, when an object $\mathbf{x}_i$ is *within* the hypersphere, the constraint $\|\mathbf{x}_i - \mathbf{a}\|^2 \leq R^2$ is satisfied, and the corresponding Lagrange multiplier becomes zero: $\alpha_i = 0$. For an object $\mathbf{x}_i$ on the boundary of the sphere, $\|\mathbf{x}_i - \mathbf{a}\|^2 = R^2$, the Lagrange multiplier becomes positive: $\alpha_i > 0$. When the Lagrange multiplier α_i hits the upper bound C, the hypersphere description is not adjusted further to include the corresponding object $\mathbf{x}_i$ in the description. The Lagrange multiplier α_i will stay at C and the object $\mathbf{x}_i$ will be outside the hypersphere. Objects for which the Lagrange multiplier is larger than zero, $\alpha_i > 0$, will be called the support vectors.

Table A.1: The values of the Lagrange multipliers α_i and γ_i for different placements of the object with respect to the hypersphere.

Object placement	α_i	γ_i
Object $\mathbf{x}_i$ is *within* the hypersphere	0	C
Object $\mathbf{x}_i$ is *on* the hypersphere boundary (SV^{bnd})	between 0 and C	between 0 and C
Object $\mathbf{x}_i$ is *outside* the hypersphere (SV^{out})	C	0

In figure A.1 the three situations for an object $\mathbf{x}_i$ (inside the sphere, on the boundary and outside the sphere) are shown with their corresponding values for α_i. Furthermore, the values of the Lagrange multiplier γ_i are shown. This Lagrange multiplier forces the error ξ_i to become zero within the sphere, such that it will not be counted in the error (A.1). The Lagrange multiplier γ_i becomes therefore active ($\gamma_i > 0$) when the data vector $\mathbf{x}_i$ is within the sphere.

Because the two Lagrange multipliers α_i and γ_i are coupled (by equations (A.6)), one multiplier automatically determines the value of the other. For object $\mathbf{x}_i$ within the hypersphere, $\alpha_i = 0$ and therefore $\gamma_i = C$. This should counterbalance exactly the term $C\xi_i$ from the second term in (A.8). Note that ξ_i is the distance from the object to the boundary and objects within the sphere have negative ξ_i. By adding $\gamma_i\xi_i = C\xi_i$ in effect no error is counted. For an object on the boundary, both constraints have to be enforced and both $\alpha_i > 0$ and $\gamma_i > 0$. Finally, for objects outside the sphere, $\alpha_i = C$. The constraint that $\xi_i > 0$ is automatically satisfied and therefore $\gamma_i = 0$. All these situations are listed in table A.1.

For the computation of the radius R of the hypersphere, the distance is calculated from the center of the sphere $\mathbf{a}$ to a support vector on the boundary of the sphere. This means that the corresponding Lagrange multiplier of the support vector should fulfill $0 < \alpha_i < C$, to avoid using objects outside the hypersphere ($\alpha_i = C$).

B. NEAREST NEIGHBOR DATA DESCRIPTION

In the nearest neighbor method, NN-d, a test object $\mathbf{z}$ is accepted when its local density is larger or equal to the local density of its nearest neighbor in the training set $\mathrm{NN}^{tr}(\mathbf{z}) = \mathrm{NN}_1^{tr}(\mathbf{z})$ (see section 3.4.2 on page 69). For the local density estimation, just the first nearest neighbor is used. This results in the following acceptance function:

$$f_{\mathrm{NN}^{tr}}(\mathbf{z}) = I\left(\frac{\|\mathbf{z} - \mathrm{NN}^{tr}(\mathbf{z})\|}{\|\mathrm{NN}^{tr}(\mathbf{z}) - \mathrm{NN}^{tr}(\mathrm{NN}^{tr}(\mathbf{z}))\|} \leq 1\right) \tag{B.1}$$

This means that the distance from object $\mathbf{z}$ to its nearest neighbor in the training set $\mathrm{NN}^{tr}(\mathbf{z})$ is compared to the distance from this nearest neighbor $\mathrm{NN}^{tr}(\mathbf{z})$ to *its* nearest neighbor (see figure B.1).

This NN-d has several predefined choices, choices for the magic parameters. First of all, more neighbors can be considered. One can use the distance to the k-th nearest neighbor, one can use the average of the k distances to the first k neighbors or one can change the threshold of 1.0 to either higher or lower values. Increasing the number of neighbors will decrease the local sensitivity of the method, but it will make the method less noise-sensitive. Because we often consider low sample sizes, $k = 1$ is chosen to retain the local sensitivity. In the few coming experiments in this appendix, the threshold of 1.0 is used, but in the experiments in chapter 4 the threshold is changed to obtain ROC curves.

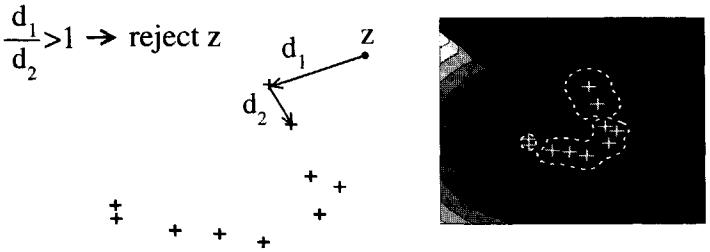


Fig. B.1: The NN-d compares the distance from a test object $\mathbf{z}$ to its nearest neighbor in the training set $\mathrm{NN}^{tr}(\mathbf{z})$ with the distance from this object to its nearest neighbor $\mathrm{NN}^{tr}(\mathrm{NN}^{tr}(\mathbf{z}))$.

The extrapolation ability and the sensitivity to outliers of this function can be shown by applying it to some simple distributions. For three data distributions, which differ in the sharpness of the distribution boundary, the performance of the NN-d is compared to the Gaussian method. The distributions considered are a uniform distribution (with data uniformly between -1 and $+1$), Gaussian distribution (zero mean and unit variance) and t-distribution (with the number of degrees of freedom equal to the data dimensionality). For uniformly distributed data the NN-d is expected to work well due to the sharp boundaries of the data. The t-distribution, on the other hand, is an example of a distribution containing a lot of remote objects which will deteriorate the performance of the NN-d. For normally distributed data the Gaussian method should work very well, the model fits the data perfectly. The performance of the NN-d can then be compared with the optimal model.

B.1 1-dimensional data

For low dimensional data and low sample sizes we can easily derive the differences between the NN-d and the Gaussian model. A NN-d and Gaussian model is trained for several samples of the uniform distribution between -1 and 1. The size or width of the description is obtained by calculating the difference in the upper and lower boundary.

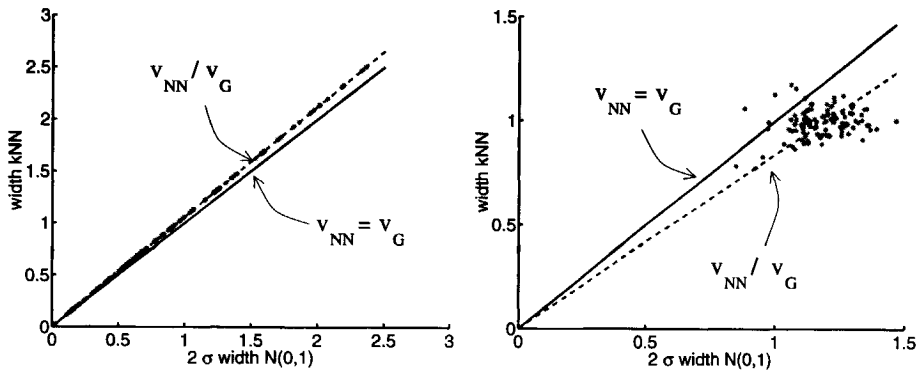


Fig. B.2: Covered surface of NN-d versus Gaussian model for a 1-dimensional uniform distribution (between -1 and 1), from left to right: 2 and 25 training objects. The results are averaged over 100 runs.

For the Gaussian density the threshold is set at 2σ, for the NN-d the fixed threshold of 1 is applied in the acceptance function (3.26). In figure B.2 the width of the NN-d is plotted versus the Gaussian description for 100 1-dimensional samples, each sample containing 2 training objects (such that the boundaries of the distribution should be estimated on the basis of just 2 objects). Trained on 2 objects, the width of the Gaussian description is consequently a bit tighter than the NN-d method. For a sample size of just 2 the width of the NN-d and Gaussian distributions can be computed exactly: $v_{NN} = \frac{3}{2}d$ and $v_G = \sqrt{2}d$

where d is the distance between the two points. v_{NN}/v_G corresponds to the slope in figure B.2.

The NN-d gives tighter descriptions for a sample size of 25 (right subplot of figure B.2). This can be concluded from the figure by the fact that almost all ratios v_{NN}/v_G are smaller than 1. For larger sample sizes, the NN-d finds on average a width of 1.0, while the Gaussian method, in general, overestimates the width of the description. The dashed line is the minimum least squares estimate over 100 experiments. It gives an indication of the relative covered surfaces by the two methods. A slope smaller than 1.0 indicates that on average the NN-d method covers less space than the Gaussian model. It also means that a part of the uniform distribution is not captured. So, for this uniform distribution with sharp boundaries the NN-d gives tighter descriptions than the Gaussian model while it rejects a smaller part of the target distribution.

In figure B.3 the volumes of the NN-d and the Gaussian model are compared for increasing sample sizes and for different data distributions. We considered the uniform distribution (between -1 and 1), the Gaussian distribution ($\mu = 0, \sigma = 1$) and the student-t distribution (with d degrees of freedom). In the left plot of figure B.3 the ratio between the covered space by the NN-d and by the Gaussian model is shown. In the right plot the ratio of the total covered probabilities are shown. For the uniform distribution, the

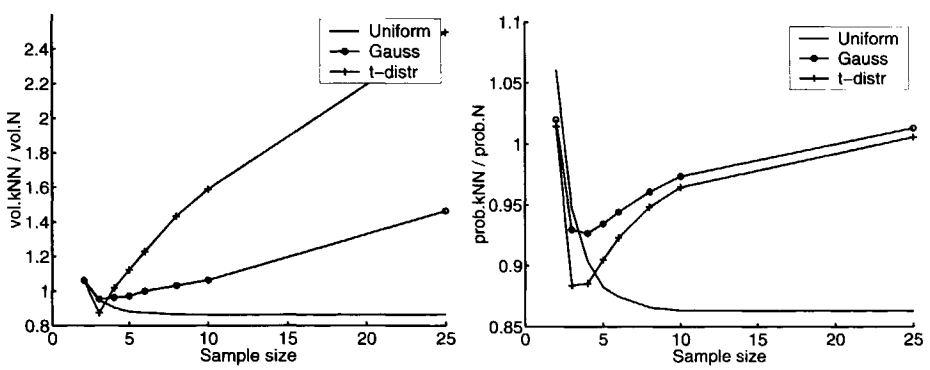


Fig. B.3: The ratio v_{NN}/v_G for three 1-dimensional datasets (uniform, Gaussian and t-distribution) and different sample sizes.

NN-d covers far less volume than the Gaussian method, at the expense of a 10% decrease in covered target distribution. For sample sizes smaller than 5 the NN-d is also more efficient in describing the Gaussian distributed data and the t-distributed data. For the Gaussian distribution and the t-distribution and sample sizes larger than 5 the Gaussian method becomes more efficient. The captured volume is less than for the NN-d while it still captures more of the distribution.

B.2 2-dimensional data

The same characteristics can be observed in higher dimensional feature spaces. In figure B.4 the NN-d and the Gaussian model are compared for all 2-dimensional data sets. In 2

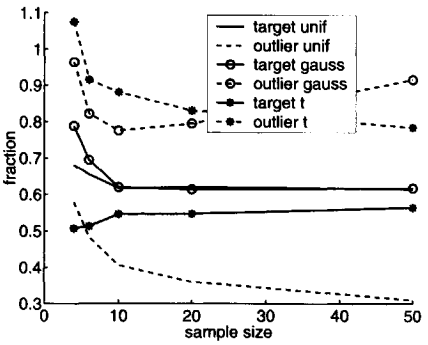


Fig. B.4: Ratio between NN-d and Gaussian acceptance rates for target and outlier data from uniform, 2-dimensional normal and 2-dimensional t-distributions as function of the sample size.

dimensions the captured volume can not be calculated directly, but it is estimated using a grid of outlier objects. In the figure an indication of the relative performances is shown:

$$f_{NN/G} = \frac{v_{NN}}{v_G} = \frac{\#\text{ accepted by the NN-d}}{\#\text{ accepted by the Gaussian model}} \tag{B.2}$$

This ratio can be estimated for the covered feature space area or the fraction of target objects accepted. A ratio lower than 1.0 for the target set indicates that the NN-d accepts less target objects than the Gaussian. On the other hand, a ratio lower than 1.0 for the outlier set means a better outlier rejection. The figure shows that for all data distributions the NN-d only captures about 60% of the target set which is captured by the Gaussian model. On the other hand, the outlier acceptance of the NN-d is smaller in comparison to the Gaussian model. So, the same characteristics as in the 1-dimensional data appear, the NN-d is especially useful for the uniform distribution (where it only covers one third of the surface covered by the normal distribution), but far less for the normal distribution or the t-distribution.

B.3 Data in a subspace

When data is distributed in a subspace, the target density distribution is truly bounded. For data outside (some band around) the subspace the probability drops to zero. To simulate data in a subspace, a Gaussian distribution is taken, which is embedded in n dimensions. The target class has a standard deviation of 1.0 in the first two features,

and 0.1 in the remaining features, thus creating a 2-dimensional subspace. To detect how tightly the subspace is captured by the outlier methods, the outlier class contains two subclasses with the same standard deviations, but with their means at ± 1 for the low variance features (the means are equal in the first two features). The Matlab code is given below.

```
% Matlab code for creating artificial 'pancake'
% data in a subspace.

tar_mean = zeros(1,d);
tar_cov = 0.1*ones(1,d);
tar_cov(1:2) = 1.0;

out_mean = ones(1,d);
out_mean(1:2) = 0.0;
out_cov = tar_cov;

x_tar = gauss(tar_mean,diag(tar_cov));
x_out = [gauss(out_mean,diag(out_cov));
         gauss(-out_mean,diag(out_cov))];
```

A scatter plot of the second and third feature is shown in the left plot of figure B.5. Note that when the largest components are considered, all important structure in the data is removed. Again the ratio $f_{NN/G}$ is measured for the target and outlier data. In the

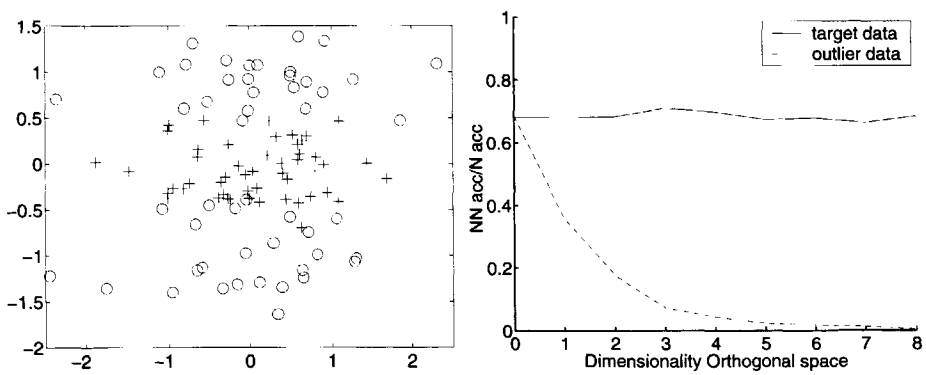


Fig. B.5: Performance of the NN-d and the Gaussian density for the **pancake** data for different dimensionalities. Target objects are marked using '+', outlier objects are marked 'o'.

testing of the Gaussian distribution with both target and outlier objects, it turns out (not visible in this plot) that the Gaussian model accepts about 85% of the target set, but

also 85% of the outliers. The NN-d, on the other hand, accepts about 60% of the target data, but it rejects almost all outliers, especially in high dimensional spaces. This can be observed in the right plot in figure B.5. The ratio of target data accepted by the Gaussian density to the NN-d is about 70% for all dimensionalities (this is the ratio 60%/85%). The ratio of outlier data which is rejected decreases rapidly for increasing dimensionality, such that for 7 dimensional data the nearest neighbor method significantly outperforms the Gaussian model.

Finally, it appears that by increasing the threshold of NN-d from 1.0 to 2.0 the target acceptance increases to 80%, but the outlier acceptance to 30%.

B.4 Empty areas

The properties of the NN-d in empty areas can only be considered when the data distribution has a limited volume, i.e. for regions far from the target center, the density should become 0: $p(\mathbf{z}) = 0$. This is due to the fact that the method (in this form) tries to find the exact boundary of the target distribution. Objects $\mathbf{z}$ are accepted when:

$$f_{\mathrm{NN}^{tr}}(\mathbf{z}) = I\left(V(\mathbf{z} - \mathrm{NN}^{tr}(\mathbf{z})) \leq V(\mathrm{NN}^{tr}(\mathbf{z}) - \mathrm{NN}^{tr}(\mathrm{NN}^{tr}(\mathbf{z})))\right) \tag{B.3}$$

Now for increasing sample size:

$$\lim_{N\to\infty} V(\mathrm{NN}^{tr}(\mathbf{z}) - \mathrm{NN}^{tr}(\mathrm{NN}^{tr}(\mathbf{z}))) = 0 \tag{B.4}$$

thus object $\mathbf{z}$ is only accepted when also

$$\lim_{N\to\infty} V(\mathbf{z} - \mathrm{NN}^{tr}(\mathbf{z})) = 0 \tag{B.5}$$

or for a spherical volume V:

$$\lim_{N\to\infty} \|\mathbf{x} - \mathrm{NN}^{tr}(\mathbf{x})\|^2 = 0 \tag{B.6}$$

Thus for increasing sample sizes an object $\mathbf{z}$ is only accepted when the distance to the nearest neighbor in the target distribution vanishes and the object is *in* the target distribution ($p(\mathbf{z}) > 0$). When a normal distribution is considered, everywhere in the feature space the probability density is larger than 0.

On the other hand, the NN-d also reject parts of the feature space which are within the target distribution. When we consider a uniform target distribution and we leave one object $\mathbf{x}$ out of the training, it could only be accepted by the description when it becomes the nearest neighbor of its nearest neighbor in the training set $\mathrm{NN}^{tr}(\mathbf{x})$. Thus, only the fraction of objects which are their mutual nearest neighbors will be consistently accepted. This rate is independent of the sample size, but does depend on the dimensionality of the data. In table 3.1 (page 71) the fraction of the target data (drawn from a uniform distribution), is accepted by the NN-d, is shown for some dimensionalities and number of neighbors.

C. HANDWRITTEN DIGITS DATASET

The images of handwritten digits are extracted from nine original maps from a Dutch public utility. The maps represent the position of a conduit system with respect to certain landmarks and were hand drawn by a large group of drawing engineers over a period of more than 25 years. The dataset is composed of separate digits which give the dimensions of different parts in the map. The digits were automatically extracted from a binary image of the map, deskewed and normalized to fit exactly into a 30 by 48 pixel region. Finally, the digits were labeled manually. This data set is also used and explained in [van Breukelen et al., 1997]. In figure C.1 a small part (600×500 pixels) of a large scanned map is shown. A few example digits are shown in figure 4.21 on page 113.

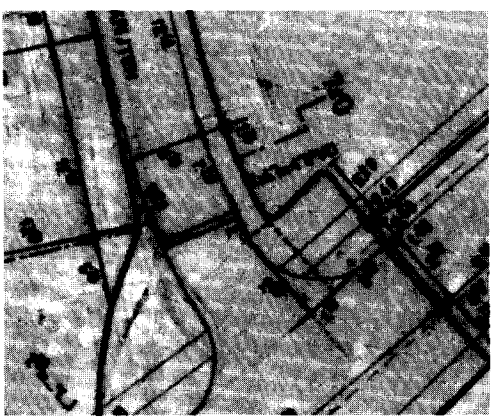


Fig. C.1: Small region (600×500 pixels) from a large utility map.

The data consists of 10 classes, each having 200 examples (thus equal class probabilities are assumed). From the set of 2000 digits 4 types of feature sets are extracted: Zernike moments, Fourier descriptors, profile images and image pixel vectors. Zernike moments are the projection of the image function onto a set of orthogonal basis functions. There are 13 orders of Zernike moments with 49 moments associated in total. For the feature extraction only the last 11 orders were used resulting in a subset of 47 Zernike moments. Also some morphological features were added (the number of holes in the image and the number of end points and branch points of the skeleton). This dataset is an adapted version of the UCI dataset, multiple features dataset, which can be found at [Duin, 1999].

The Fourier feature set contains 76 Fourier coefficients, computed by transforming the original pixel image. The Fourier set is rotation invariant, just like the Zernike feature set. In the profile feature set 12 faces and cross-sections of all digits are made. From these 12 vectors the correlations to 18 reference digits are computed. This results in a feature vector of 216 correlation features per digit.

The fourth feature set is based on a simple transform of the binary image to a pixel vector. To reduce the number of features and the possible loss of information, the normalized image was divided into tiles of 2 by 3 pixels resulting in a total of 240 tiles. Each tile represents an element of the feature vector and the value corresponding to each tile was calculated simply by counting the number of object pixels within the tile.

The total number of 2000 digits is divided into a training set and a testing set of 1000 objects each; 100 digits per class. In the application of one-class classification this means that for training, the training sets consists of just 100 target objects. For testing again 100 target and 900 outlier objects are available.

The feature sizes of the original datasets are relatively high (53 for the Zernike dataset, 76 for Fourier, 216 features for profile and 240 for the pixel dataset). Considering that just 100 target objects are available for training, the number of features is reduced by projecting the data on the first few principal components. By retaining 90% of the variance in the data, the dimensionality is could be significantly reduced (13 for the Zernike dataset, 47 for Fourier, 19 features for profile and 51 for the pixel dataset). These reduced datasets are used in all experiments.

C.1 Individual classifiers

Table C.1: Results of all one-class classifiers on the Zernike dataset (integrated ROC ×100). The best results are marked in bold.

class	Gau	MoG	Par	NNd	SVDD	LVQ	k-m	k-c	aut	PCA	SOM
0	2.80	1.15	0.45	11.67	4.63	**0.16**	1.98	0.25	2.46	1.47	0.51
1	2.00	1.17	0.72	3.26	6.02	5.81	2.67	1.81	2.73	**0.63**	0.72
2	2.21	1.82	**1.05**	4.56	6.82	1.93	2.98	2.02	1.62	2.69	2.69
3	**3.60**	5.30	3.83	10.92	11.28	5.03	5.30	7.41	7.46	4.80	4.61
4	1.07	3.56	0.78	4.46	7.94	4.13	2.89	3.38	4.50	**0.75**	2.53
5	2.01	3.26	**1.62**	7.24	10.48	4.89	3.03	6.20	6.39	6.52	2.88
6	4.20	5.79	**4.19**	13.23	8.97	4.90	5.06	4.53	4.63	4.25	4.48
7	0.55	1.10	**0.35**	4.86	10.13	0.78	0.65	1.67	0.80	1.27	1.36
8	1.80	7.22	0.87	3.69	5.34	**0.47**	1.30	0.61	1.57	0.96	2.73
9	4.28	4.97	4.07	12.84	8.05	4.40	**3.92**	5.26	6.51	4.33	4.22
average	2.45	3.54	**1.79**	7.67	7.97	3.25	2.98	3.31	3.87	2.77	2.67

The methods presented in chapter 3 are applied to this handwritten digits data. The methods which are considered here are: Gaussian model (`Gau`), Mixture of Gaussians (`MoG`), Parzen density estimate (`Par`), nearest neighbor method (`NNd`), support vector data description (`SVDD`), (in some cases support vector data description using negative examples: `SVDn`), Learning Vector Quantization (`LVQ`), k-means clustering (`k-m`), k-center method (`k-c`), auto-encoder network (`aut`), principal component analyzer (`PCA`) and the self-organizing map (`SOM`). All classifiers are trained for a range of threshold values and the integrated ROC error is computed. The error is integrated from 50% to 95% target acceptance.

Table C.2: Results of all one-class classifiers on the Fourier dataset (integrated ROC $\times 100$). The best results are marked in bold.

class	Gau	MoG	Par	NNd	SVDD	LVQ	k-m	k-c	aut	PCA	SOM
0	4.00	0.93	**0.04**	1.60	16.22	0.21	0.79	1.02	0.81	3.60	1.01
1	28.47	7.27	**5.30**	13.85	22.87	7.45	7.00	9.63	7.94	26.88	8.79
2	13.61	2.09	**0.96**	5.63	27.71	2.21	1.94	3.19	2.57	21.94	2.54
3	33.64	11.32	9.15	9.82	23.44	12.57	11.18	14.91	**7.80**	42.34	10.78
4	37.95	18.14	**12.01**	16.18	26.47	22.84	21.32	17.26	15.70	29.87	14.76
5	43.05	37.89	37.31	38.46	**26.53**	38.15	38.11	39.15	37.88	43.45	37.64
6	33.67	19.56	15.72	16.75	32.77	20.24	20.27	24.48	**15.20**	18.76	20.41
7	13.72	3.81	**1.48**	3.80	22.72	5.00	3.14	4.07	4.26	7.32	5.32
8	31.22	45.00	**18.28**	21.40	30.10	18.34	19.30	20.53	25.04	39.61	23.98
9	31.20	20.12	**14.82**	22.20	30.58	17.32	18.87	17.94	14.84	22.42	16.75
average	27.06	16.62	**11.51**	14.97	25.94	14.43	14.19	15.22	13.20	25.62	14.20

In tables C.1 to C.4 the errors on the 4 feature sets, Zernike, Fourier, profile and pixel are presented for all methods and all target classes. The error is computed using formula (3.7) (and is multiplied by 100 to avoid printing all preceding zeros). In each row in the tables another class is used as target class. For each target class the best results are indicated in bold. The last row shows the average performance of each method averaged over all 10 target classes.

Looking at the results, it appears that no single one-class classifier outperforms all other methods, but some trends are visible. On the Zernike data the Gaussian model, the k-means and k-centers perform well. This indicates that the Zernike data is nicely clustered. Some of the target classes are easier to describe, like class '0' and '8' (both by the k-center method). Others show larger overlap with other data, for instance, in classes '3', '6' and '9'. The problems with the last two classes is not surprising for the Zernike dataset, because the Zernike features are rotation invariant and it is therefore very hard to distinguish the '6' and the '9'. When one method has to be used for all classes, the Gaussian model best fits this data.

Table C.3: Results of all one-class classifiers on the profile dataset (integrated ROC ×100). The best results are marked in bold.

class	Gau	MoG	Par	NNd	SVDD	LVQ	k-m	k-c	aut	PCA	SOM
0	0.65	0.35	**0.08**	2.18	9.58	1.04	2.33	2.14	5.08	4.48	0.35
1	3.42	2.50	**0.13**	5.34	9.38	2.60	1.26	1.82	1.33	0.52	1.01
2	0.85	0.20	**0.07**	4.51	7.67	0.45	0.48	1.71	0.86	1.08	0.16
3	2.24	1.33	0.59	9.71	10.14	0.58	0.76	**0.52**	4.97	5.30	1.08
4	0.13	0.16	**0.13**	4.04	11.06	0.47	0.68	0.84	5.28	0.23	1.13
5	2.42	1.86	**0.78**	7.65	12.79	1.31	0.82	2.35	9.83	8.72	2.02
6	9.44	0.15	0.16	4.76	7.13	0.15	0.25	0.26	0.89	**0.09**	0.47
7	18.06	1.44	**0.13**	0.92	10.10	0.89	1.02	0.48	1.53	0.95	2.31
8	2.17	0.86	1.04	4.70	16.22	**0.75**	1.33	2.35	6.51	5.89	1.41
9	1.57	0.67	**0.21**	3.08	8.45	0.32	0.38	0.80	2.95	1.73	1.05
average	4.09	0.95	**0.33**	4.69	10.25	0.86	0.93	1.33	3.92	2.90	1.10

Table C.4: Results of all one-class classifiers on the pixel dataset (integrated ROC ×100). The best results are marked in bold.

class	Gau	MoG	Par	NNd	SVDD	LVQ	k-m	k-c	aut	PCA	SOM
0	21.60	9.99	**0.09**	2.36	17.22	0.62	1.45	0.86	8.60	26.80	0.39
1	34.60	3.29	**0.04**	1.94	20.20	0.45	0.26	0.94	7.00	18.20	0.38
2	29.80	5.41	**0.01**	2.22	17.00	0.10	0.13	2.12	6.81	14.63	3.42
3	45.00	10.29	**0.14**	5.81	10.38	0.64	0.46	0.87	6.26	26.80	1.63
4	24.00	9.85	**0.01**	3.40	27.80	0.69	0.11	0.90	4.01	12.01	0.43
5	34.80	3.41	**0.47**	7.79	21.51	2.08	1.28	1.97	**3.63**	32.41	2.28
6	31.00	2.43	**0.16**	4.14	17.02	0.68	0.62	1.52	3.86	17.60	1.38
7	45.00	1.01	**0.01**	0.88	18.00	0.05	0.02	0.47	1.20	5.20	0.02
8	39.04	4.82	**0.62**	4.10	24.97	2.41	2.89	4.80	6.78	41.81	3.38
9	40.60	0.30	**0.16**	3.51	19.45	0.62	0.40	1.64	0.94	21.21	0.75
average	34.54	5.08	**0.17**	3.61	19.36	0.83	0.76	1.61	4.91	21.67	1.40

On the Fourier, profile and pixel data sets another picture arises. Here, the Parzen density often works well, with some exceptions for some classes. Still, on average, the Parzen density estimation is the best for all feature sets.

The average performance of the methods is good for Zernike, profile and pixel datasets, but in the Fourier dataset some classes are very hard to describe, for instance, classes '3'-'6', '8' and '9'.

The pixel dataset contains 51 features (this is after the PCA mapping which retains

Table C.5: Results of all one-class classifiers on the pixel dataset (integrated ROC ×100). The integration boundary is changed: the maximal target acceptance rate is set to 85%.

class	Gau	MoG	Par	NNd	SVDD	LVQ	k-m	k-c	aut	PCA	SOM
0	11.60	7.02	0.02	1.35	7.61	0.11	0.06	0.08	1.40	16.80	**0.01**
1	24.60	0.03	0.01	0.91	10.20	0.05	0.05	0.04	1.00	10.20	**0.01**
2	19.80	0.40	**0.00**	0.83	7.00	0.01	0.01	0.05	0.60	5.23	0.00
3	35.00	7.02	**0.02**	3.53	1.93	0.14	0.06	0.13	0.42	16.80	0.04
4	14.00	7.01	**0.00**	1.86	17.80	0.04	0.02	0.02	0.60	4.20	0.00
5	24.80	0.19	0.10	3.91	11.51	0.53	0.21	0.34	0.08	22.41	**0.05**
6	21.00	0.04	0.01	2.15	7.22	0.11	0.08	0.19	**0.01**	7.60	0.01
7	35.00	0.00	0.00	0.31	8.00	0.00	0.00	0.03	**0.00**	0.20	0.00
8	29.04	0.61	**0.19**	2.16	15.17	0.93	0.65	0.82	1.59	31.81	0.40
9	30.60	0.07	0.05	1.75	9.45	0.15	0.09	0.26	0.05	11.40	**0.04**
average	24.54	2.24	0.04	1.88	9.59	0.21	0.12	0.20	0.57	12.67	0.06

about 90% of the variance in the data) and the target class contains just 100 training objects. It is, therefore, to be expected that the performance for high target acceptance rates might be poor. In these cases all methods have to extrapolate to reject just a small fraction (5%) of the target objects. In table C.5 other integration bounds for the ROC curves are used in the pixel database. Instead of a maximal target acceptance rate of 95%, this rate is decreased to 85%. Comparing the results of tables C.5 and C.4 we see that the errors obtained by all methods decrease for smaller target acceptance rates. Furthermore, we see that the extrapolation ability is best in the Parzen density estimation. When the extrapolation ability is stressed less, other methods can model the data better. For lower target acceptance rates both the auto-encoder and the SOM perform well. Especially in this high dimensional data it is to be expected that data are in a subspace. The fact that the PCA performs far worse, indicates that this subspace is not linear.

C.2 Combining classifiers

Now that we have trained the individual classifiers on all the digit classes, we can investigate the performance of the five different combining rules, defined in formulae (5.20) to (5.24) on page 125. First we will combine different one-class classifiers which are trained on the same data set. All methods are trained and optimized to have a target acceptance rate of 90%. The outputs of all methods are combined and on the resulting output the target rejection rates are varied, such that a ROC curve and the error on that ROC curve can be calculated. These ROC errors are shown in tables C.6 to C.9. Numbers in bold indicate that this error is smaller than the best performance reached by an individual classifier (and thus combining classifiers is useful).

Table C.6: Integrated ROC errors (×100), combining the all one-class classifiers trained on the Zernike dataset. The 5 combining rules are defined in formulae (5.20) to (5.24) on page 125.

combining method	target class nr. 0	1	2	3	4	5	6	7	8	9
mv	**0.01**	**0.21**	**0.55**	6.69	0.89	**1.51**	**3.69**	0.49	**0.01**	4.16
mwv	**0.01**	**0.20**	**0.55**	6.71	0.87	**1.49**	**3.67**	0.45	**0.01**	4.12
pwv	**0.01**	**0.21**	**0.54**	6.58	0.89	**1.50**	**3.67**	0.47	**0.01**	4.15
mp	**0.01**	**0.16**	**0.59**	5.66	**0.65**	1.72	**3.77**	**0.22**	**0.00**	4.22
pp	**0.00**	**0.09**	**0.40**	4.45	**0.48**	**1.01**	**3.83**	**0.16**	**0.00**	4.13

Table C.7: Integrated ROC errors (×100), combining all the one-class classifiers trained on the Fourier dataset. The 5 combining rules are defined in formulae (5.20) to (5.24) on page 125.

combining method	target class nr. 0	1	2	3	4	5	6	7	8	9
mv	0.30	8.82	4.43	19.80	28.95	28.37	23.09	4.45	**0.72**	26.76
pwv	0.30	8.72	4.28	19.25	28.33	27.80	22.94	4.23	**0.72**	26.34
pwv	0.30	8.76	4.27	19.73	28.67	28.12	23.07	3.76	**0.70**	26.68
mp	**0.00**	6.85	2.33	15.86	23.12	**22.05**	20.89	2.49	45.00	24.85
pp	**0.00**	5.83	1.28	9.94	17.34	**13.88**	**13.47**	**1.30**	45.00	17.02

In tables C.6, C.7 and C.8 we see that for some classes combining several methods is actually improving the classification performance. While the performance is increased in classes '1', '2' and '7' in the Zernike dataset, for the other classes no improvement is obtained. Probably these classes have boundaries which are hard to fit using just one model and a combination of several models have to be used, to get best performance. For other classes one of the methods already fit quite well (for instance, for class '0' in the Zernike dataset, the Parzen, LVQ or k-means already give a very good description of class '0').

When we just concentrate on the combining rule, we see that the combining rule based on the product of the estimated probabilities (5.24), in general, outperforms all other rules. This is somewhat surprising because this rule use a product combination which is noise sensitive. In the combination of standard 2-class classifiers it appears that in many situations the more robust average combination rule is to be preferred. Here extreme posterior probability estimates are averaged out. In one-class classification on the other hand, only the target class is modeled and a low uniform distribution is assumed for the

Table C.8: Integrated ROC errors (×100), combining all the one-class classifiers trained on the profile dataset. The 5 combining rules are defined in formulae (5.20) to (5.24) on page 125.

combining method	target class nr. 0	1	2	3	4	5	6	7	8	9
mv	**0.07**	1.42	0.10	**0.40**	0.28	1.86	0.47	**0.02**	1.20	0.39
mwv	**0.06**	1.59	0.09	**0.40**	0.27	1.88	0.33	**0.02**	1.15	0.37
pwv	**0.07**	1.37	0.10	**0.39**	0.28	1.70	0.44	**0.01**	1.14	0.37
mp	**0.05**	0.37	0.09	**0.27**	0.17	1.00	0.42	**0.00**	0.87	**0.19**
pp	**0.07**	**0.09**	**0.06**	**0.25**	**0.12**	**0.65**	0.12	**0.03**	0.85	0.21

Table C.9: Integrated ROC errors (×100), combining all the one-class classifiers trained on the pixel dataset. The 5 combining rules are defined in formulae (5.20) to (5.24) on page 125.

combining method	target class nr. 0	1	2	3	4	5	6	7	8	9
mv1	1.08	5.61	0.94	6.29	2.13	5.80	3.57	0.10	8.69	2.47
mwv	0.96	5.14	0.88	5.97	1.76	5.37	3.23	0.11	8.37	2.11
pwv	1.05	4.94	0.83	6.15	1.79	5.67	3.13	0.07	8.57	2.31
mp	0.35	3.11	0.48	4.70	0.63	1.83	1.69	0.03	4.39	1.08
pp	**0.01**	0.62	0.15	1.99	0.11	**0.28**	0.17	**0.01**	1.97	0.31

outlier class. This makes this classification problem asymmetric and extreme target class estimates are not cancelled by extreme outlier estimates. When a few extreme target class probability estimates are present, the mean combination will cover a broad domain in feature space, while the product rule has restricted range. In particular, in high dimensional spaces this extra area will cover a large volume and potentially a large number of outliers.

C.3 Combining featuresets

Finally, the combination rules are applied to the results of the same type of one-class classifier trained on different feature sets. Again the classifiers are trained to accept 90% of the target class, and the errors are again the integrated ROC curves.

For a reference, the best individual performances (over all different feature sets) for each of the classes is:

When finally the classifiers trained on seperate feature sets are combined, we see the following performances for the different combination rules:

Table C.10: Best individual performances of all methods over all feature sets. Performances are given in integrated ROC errors (×100).

	one-class classification method										
classnr	Gau	MoG	Par	NNd	SVDD	SVDn	LVQ	k-m	k-c	PCA	SOM
0	0.65	0.35	0.04	1.60	4.63	5.56	0.16	0.79	0.25	1.47	0.35
1	2.00	1.17	0.04	1.94	6.02	6.27	0.45	0.26	0.94	0.52	0.38
2	0.85	0.20	0.01	2.22	6.82	11.64	0.10	0.13	1.71	1.08	0.16
3	2.24	1.33	0.14	5.81	10.14	7.27	0.58	0.46	0.52	4.80	1.08
4	0.13	0.16	0.01	3.40	7.94	7.35	0.47	0.11	0.84	0.23	0.43
5	2.01	1.86	0.47	7.24	10.48	9.55	1.31	0.82	1.97	6.52	2.02
6	4.20	0.15	0.16	4.14	7.13	11.67	0.15	0.25	0.26	0.09	0.47
7	0.55	1.01	0.01	0.88	10.10	9.11	0.05	0.02	0.47	0.95	0.02
8	1.80	0.86	0.62	3.69	5.34	10.62	0.47	1.30	0.61	0.96	1.41
9	1.57	0.30	0.16	3.08	8.05	8.14	0.32	0.38	0.80	1.73	0.75
average	1.60	0.74	0.17	3.40	7.66	0.40	0.45	0.84	1.98	1.83	0.71

comb.	one-class classification method										
method	Gau	MoG	Par	NNd	SVDD	SVDn	LVQ	k-m	k-c	PCA	SOM
						class 0					
mv	**0.00**	**0.01**	**0.00**	**0.00**	**0.00**	**0.00**	**0.01**	**0.00**	**0.00**	**0.00**	**0.00**
mwv	**0.00**	**0.01**	**0.00**	**0.00**	**0.00**	**0.00**	**0.01**	**0.00**	**0.00**	**0.00**	**0.00**
pwv	**0.00**	**0.01**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
mp	**0.01**	**0.03**	**0.00**	**0.02**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
pp	**0.00**	**0.00**	**0.00**	**0.01**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
						class 1					
mv	**0.04**	**0.28**	0.05	3.40	**5.79**	**4.13**	0.63	0.35	**0.04**	**0.25**	**0.33**
mwv	**0.04**	**0.28**	0.05	3.32	**5.78**	**3.98**	0.64	0.34	**0.04**	**0.25**	**0.33**
pwv	**0.04**	**0.38**	**0.04**	3.39	**5.60**	**3.46**	**0.38**	**0.20**	**0.05**	**0.25**	0.57
mp	**0.09**	**0.21**	0.15	**0.50**	**2.17**	**0.28**	**0.01**	**0.01**	**0.04**	**0.06**	**0.00**
pp	**0.00**	**0.01**	**0.00**	**0.57**	**2.17**	**0.28**	**0.01**	**0.01**	**0.00**	**0.00**	**0.00**
						class 2					
mv	**0.02**	**0.04**	**0.00**	**0.82**	**0.31**	**0.02**	**0.06**	**0.12**	**0.14**	**0.11**	**0.00**
mwv	**0.02**	**0.04**	**0.00**	**0.96**	**0.32**	**0.02**	**0.09**	**0.12**	**0.14**	**0.11**	**0.00**
pwv	**0.02**	**0.03**	**0.00**	**0.82**	**0.32**	**0.02**	**0.06**	**0.09**	**0.14**	**0.11**	**0.00**
mp	**0.27**	0.45	0.43	**0.03**	**0.04**	**0.00**	**0.00**	**0.00**	**0.19**	**0.01**	**0.00**
pp	**0.00**	**0.00**	**0.00**	**0.04**	**0.03**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**

comb. method	one-class classification method										
	Gau	MoG	Par	NNd	SVDD	SVDn	LVQ	k-m	k-c	PCA	SOM
class 3											
mv	**0.28**	1.46	0.41	**3.84**	**4.70**	**6.59**	1.71	3.82	0.57	**0.59**	**0.67**
mwv	**0.28**	1.41	0.25	**3.09**	**4.52**	**6.05**	1.47	3.31	0.57	**0.59**	**0.67**
pwv	**0.27**	1.73	0.41	**3.79**	**6.36**	**6.42**	1.88	3.82	0.77	**0.62**	**0.72**
mp	**1.31**	4.16	7.90	**0.93**	**3.44**	**1.77**	**0.45**	**0.35**	0.52	**0.25**	**0.10**
pp	**0.04**	1.77	0.15	**0.98**	**3.50**	**1.85**	**0.50**	0.51	**0.11**	**0.04**	**0.17**
class 4											
mv	**0.04**	1.05	0.11	**3.35**	**7.92**	**3.01**	2.81	1.28	**0.56**	**0.08**	1.75
mwv	**0.04**	1.02	0.10	**2.18**	**7.87**	**2.27**	2.33	1.10	**0.54**	**0.08**	1.75
pwv	**0.06**	1.54	0.30	**3.35**	9.47	**3.29**	3.12	1.62	0.87	**0.08**	3.24
mp	0.18	1.67	0.82	**0.45**	**2.27**	**0.71**	**0.23**	0.12	**0.39**	**0.00**	**0.00**
pp	**0.01**	0.66	**0.00**	**0.43**	**2.27**	**0.68**	**0.22**	**0.10**	**0.00**	**0.00**	**0.00**
class 5											
mv	**0.17**	3.23	1.06	10.92	**7.72**	**5.99**	3.59	2.63	**0.41**	**0.86**	**0.92**
mwv	**0.17**	2.85	0.84	11.15	**7.42**	**4.31**	2.77	2.12	**0.41**	**0.86**	**0.92**
pwv	**0.16**	2.85	1.06	10.92	**8.80**	**5.98**	3.68	2.62	**0.59**	**0.88**	**1.16**
mp	**0.65**	**1.78**	1.31	**3.53**	**2.22**	**0.79**	**0.35**	**0.33**	**0.28**	**0.19**	**0.15**
pp	**0.02**	**1.03**	**0.35**	**6.13**	**2.24**	**0.80**	**0.36**	**0.30**	**0.01**	**0.01**	**0.18**
class 6											
mv	**0.87**	1.69	**0.11**	**1.54**	**4.02**	**2.33**	2.39	0.72	0.69	1.50	1.36
mwv	**0.87**	1.65	**0.09**	**1.36**	**3.89**	**2.12**	2.33	0.66	0.69	1.50	1.36
pwv	**1.27**	2.04	**0.11**	**1.54**	**4.73**	**2.27**	2.47	0.72	1.19	2.10	1.82
mp	4.24	5.65	6.20	**2.41**	**1.09**	**1.15**	1.25	0.96	1.29	1.57	0.70
pp	**0.00**	0.56	**0.00**	**2.73**	**0.99**	**0.90**	1.09	0.74	**0.02**	**0.00**	0.18
class 7											
mv	**0.00**	**0.04**	**0.00**	**0.17**	**0.21**	**0.05**	**0.03**	0.05	**0.01**	**0.06**	**0.09**
mwv	**0.00**	**0.04**	**0.00**	**0.12**	**0.21**	**0.05**	**0.03**	**0.02**	**0.01**	**0.06**	**0.09**
pwv	**0.00**	**0.10**	**0.01**	**0.17**	**0.36**	**0.05**	**0.03**	0.05	**0.01**	**0.06**	**0.20**
mp	**0.07**	**0.35**	0.11	**0.01**	**0.00**	**0.00**	**0.00**	**0.00**	**0.04**	**0.00**	**0.00**
pp	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
class 8											
mv	**0.01**	**0.40**	**0.11**	**1.15**	**1.05**	**0.40**	**0.14**	**0.56**	**0.15**	**0.12**	**0.00**
mwv	**0.01**	**0.40**	**0.18**	**1.43**	**1.13**	**0.62**	**0.17**	**0.64**	**0.17**	**0.12**	**0.00**
pwv	**0.01**	**0.40**	**0.11**	**1.15**	**0.87**	**0.40**	**0.14**	**0.39**	**0.13**	**0.12**	**0.00**
mp	**0.00**	85.00	**0.01**	**0.09**	**0.07**	**0.02**	**0.00**	**0.02**	**0.02**	**0.01**	**0.00**

comb.	one-class classification method										
method	Gau	MoG	Par	NNd	SVDD	SVDn	LVQ	k-m	k-c	PCA	SOM
pp	**0.01**	85.00	**0.00**	**0.10**	**0.08**	**0.03**	**0.01**	**0.02**	**0.00**	**0.00**	**0.00**
					class 9						
mv	**0.97**	1.86	0.40	3.37	**5.06**	**2.66**	1.67	2.95	**0.56**	1.50	**1.58**
mwv	**0.97**	1.86	0.34	**2.31**	**5.00**	**2.31**	1.69	2.76	**0.56**	1.50	**1.58**
pwv	1.65	2.77	0.40	3.37	**7.49**	**2.66**	1.97	3.18	**0.56**	2.03	2.77
mp	4.30	5.78	6.30	3.40	**1.92**	**1.64**	1.03	1.35	1.18	1.44	**0.78**
pp	**0.00**	1.16	**0.01**	3.49	**1.83**	**1.49**	0.69	1.16	**0.00**	**0.00**	**0.33**

Table C.11: Integrated ROC errors (×100) for the five combining methods over the four featuresets.

The results in table C.11 show that combining performance does not always increase over the individual best performances. The numbers in bold indicate when the performance is better than the best individual classifier given in table C.10. Some classes are clearly easier to distinguish from the rest, especially classes 0, 2, 7 and 8. Furthermore, the combination of the normal density is now often superior to the Parzen density estimator. Overall the best performance will be obtained when normal density models are trained on individual feature sets and the results are combined using the product combination rule.

BIBLIOGRAPHY

[Adams and Hand, 2000] Adams, N. and Hand, D. (2000). Improving the practice of classifier performance assessment. *Neural Computation*, 12(2):305–311.

[Antani et al., 1998] Antani, S., Kasturi, R., and Jain, R. (1998). Pattern recognition methods in image and video databases: past, present and future. In *Advances in Pattern Recognition, Proceedings of SPR'98 and SSPR'98*, pages 31–53, Berlin. IAPR, Springer-Verlag.

[Baldi and Hornik, 1989] Baldi, P. and Hornik, K. (1989). Neural networks and principal component analysis: learning from examples without local minima. *Neural networks*, 2:53–58.

[Barnett and Lewis, 1978] Barnett, V. and Lewis, T. (1978). *Outliers in statistical data*. Wiley series in probability and mathematical statistics. John Wiley & Sons Ltd., 2nd edition.

[Battiti and Colla, 1994] Battiti, R. and Colla, A. (1994). Democracy in neural nets: Voting schemes for classification. *Neural Networks*, 7(4):691 707.

[Benediktsson and Swain, 1992] Benediktsson, J. and Swain, P. (1992). Consensus theoretic classification methods. *IEEE Transactions on Systems, Man and Cybernetics*, 22(4):688–704.

[Bishop, 1994a] Bishop, C. (1994a). Mixture density networks. Technical Report NCRG 4288, Neural computation research group, Aston University, Birmingham.

[Bishop, 1994b] Bishop, C. (1994b). Novelty detection and neural network validation. *IEE Proceedings on Vision, Image and Signal Processing. Special Issue on Applications of Neural Networks*, 141(4):217–222.

[Bishop, 1995] Bishop, C. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press, Walton Street, Oxford OX2 6DP.

[Blake et al., 1998] Blake, C., Keogh, E., and Merz, C. (1998). UCI repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html, University of California, Irvine, Dept. of Information and Computer Sciences.

[Bourlard and Kamp, 1988] Bourlard, H. and Kamp, Y. (1988). Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59:291–294.

[Bradley, 1997] Bradley, A. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159.

[Breunig et al., 2000] Breunig, M., Kriegel, H.-P., Ng, R., and Sander, J. (2000). LOF: indentifying density-based local outliers. In *Proceedings of the ACM SIGMOD 2000 international conference on management of data.*

[Burges, 1998] Burges, C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2).

[Carpenter et al., 1991] Carpenter, G., Grossberg, S., and Rosen, D. (1991). ART 2-A: an adaptive resonance algorithm for rapid category learning and recognition. *Neural Networks*, 4(4):493–504.

[Devijver and Kittler, 1982] Devijver, P. and Kittler, J. (1982). *Pattern Recognition, A statistical approach.* Prentice-Hall International, London.

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

[Duin, 1976] Duin, R. (1976). On the choice of the smoothing parameters for Parzen estimators of probability density functions. *IEEE Transactions on Computers*, C-25(11):1175–1179.

[Duin, 1999] Duin, R. (1999). UCI dataset, multiple features database. Available from ftp://ftp.ics.uci.edu/pub/machine-learning-databases/mfeat/.

[Friedman, 1997] Friedman, J. (1997). On bias, variance, 0/1 loss, and the curse-of-dimensionality. *Data Mining and Knowledge discovery*, 1(1):55–77.

[Fukanaga, 1990] Fukanaga, K. (1990). *Statistical pattern recognition.* Academic press.

[Geman et al., 1992] Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58.

[Grimmett and Stirzaker, 1982] Grimmett, G. and Stirzaker, D. (1982). *Probability and random processes.* Clarendon press.

[Hashem, 1994] Hashem, S. (1994). Optimal linear combinations of neural networks. *Neural Networks*, .

[Hassibi and Stork, 1993] Hassibi, B. and Stork, D. (1993). Second order derivatives for network pruning: optimal brain surgeon. In Hanson, S., Cowan, J., and Giles, C., editors, *Advances in Neural Information Processing Systems*, volume 5, pages 164–172. Morgan-Kaufmann.

[Haykin, 1999] Haykin, S. (1999). *Neural Networks, a comprehensive foundation.* Prentice-Hall.

[Hertz et al., 1991] Hertz, J., Krogh, A., and Palmer, R. (1991). *Introduction to the theory of neural computation.* Addison Wesley Publishing Company.

[Heskes, 1998] Heskes, T. (1998). Bias/variance decomposition for likelihood-based estimators. *Neural Computation*, 10:1425–1433.

[Jacobs, 1995] Jacobs, R. (1995). Method for combining experts' probability assessments. *Neural Computation*, 7(5). 867-888.

[Japkowicz, 1999] Japkowicz, N. (1999). *Concept-Learning in the absence of counter-examples: an autoassociation-based approach to classification.* PhD thesis, New Brunswick Rutgers, The State University of New Jersey.

[Japkowicz et al., 1995] Japkowicz, N., Myers, C., and Gluck, M. (1995). A novelty detection approach to classification. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 518–523.

[Kittler et al., 1996] Kittler, J., Hatef, M., and Duin, R. (1996). Combining classifiers. *Proc. of ICPR'96*, :897–901.

[Kittler et al., 1997] Kittler, J., Hojjatoleslami, A., and Windeatt, T. (1997). Weighting factors in multiple expert fusion. In Clark A.F., editor, *Proceedings of the 8th British Machine Vision Conference 1997*, pages 41–50. University of Essex Printing Service.

[Knorr et al., 2000] Knorr, E., Ng, R., and Tucakov, V. (2000). Distance-based outliers: algorithms and applications. *VLDB journal*, 8(3):237–253.

[Koch et al., 1995] Koch, M., Moya, M., Hostetler, L., and Fogler, R. (1995). Cueing, feature discovery and one-class learning for synthetic aperture radar automatic target recognition. *Neural Networks*, 8(7/8):1081–1102.

[Kohonen, 1995] Kohonen, T. (1995). *Self-organizing maps.* Springer-Verlag, Heidelberg, Germany.

[Kolmogorov and Tikhomirov, 1961] Kolmogorov, A. and Tikhomirov, V. (1961). ϵ-entropy and ϵ-capacity of sets in function spaces. *Trans. of the American Mathematical Society*, 17:277–364.

[Kraaijveld and Duin, 1991] Kraaijveld, M. and Duin, R. (1991). A criterion for the smoothing parameter for parzen-estimators of probability density functions. Technical report, Delft University of Technology.

[Le Cun et al., 1989] Le Cun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., and Jackel, L. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551.

[LeCun et al., 1990] LeCun, Y., Denker, J., Solla, S., Howard, R., and Jackel, L. (1990). Optimal brain damage. In *Advances in neural information processing systems*, volume II. Morgan Kauffman.

[MacKay, 1992] MacKay, D. (1992). Bayesian methods for adaptive models. Master's thesis, California Institute of Technology, Pasadena, California.

[Messer, 1999] Messer, K. (1999). *Automatic image database retrieval system using adaptive colour and texture descriptors.* PhD thesis, University of Surrey, Guildford.

[Metz, 1978] Metz, C. (1978). Basic principles of ROC analysis. *Seminars in Nuclear Medicine*, VIII(4).

[Mitchell, 1993] Mitchell, J. S. (1993). *An introduction to machinery analysis and monitoring - 2nd ed.* PennWell Publ. Comp.

[Moya and Hush, 1996] Moya, M. and Hush, D. (1996). Network contraints and multi-objective optimization for one-class classification. *Neural Networks*, 9(3):463–474.

[Moya et al., 1993] Moya, M., Koch, M., and Hostetler, L. (1993). One-class classifier networks for target recognition applications. In *Proceedings world congress on neural networks*, pages 797–801, Portland, OR. International Neural Network Society, INNS.

[Parra et al., 1996] Parra, L., Deco, G., and Miesbach, S. (1996). Statistical independence and novelty detection with information preserving nonlinear maps. *Neural Computation*, 8:260–269.

[Parzen, 1962] Parzen, E. (1962). On estimation of a probability density function and mode. *Annals of Mathenatical Statistics*, 33:1065–1076.

[Proakis and Manolakis, 1992] Proakis, J. and Manolakis, D. (1992). *Digital signal processing - principles, algorithms and applications, 2nd ed.* MacMillan Publ., New York.

[Pudil et al., 1994] Pudil, P., Novovicova, J., and Kittler, J. (1994). Floating search methods in feature selection. *Pattern Recognition Letters*, 15(11):1119–1125.

[Randall, 1987] Randall, R. B. (1987). *Frequency analysis.* Brüel & Kjaer, Denmark.

[Raudys and Jain, 1991] Raudys, S. and Jain, A. (1991). Small sample size effects in statistical pattern recognition: recommendations for practioners. *IEEE Transactions on pattern analysis and machine intelligence*, 13(3):252–264.

[Richard and Lippmann, 1991] Richard, M. and Lippmann, R. (1991). Neural network classifiers estimate Bayesian a posteriori probabilities. *Neural Computation*, 3:461–483.

[Ripley, 1996] Ripley, B. (1996). *Pattern Recognition and Neural Networks.* Cambridge University Press, Cambridge.

[Rissanen, 1978] Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, 14:465–471.

[Ritter and Gallegos, 1997] Ritter, G. and Gallegos, M. (1997). Outliers in statistical pattern recognition and an application to automatic chromosome classification. *Pattern Recognition Letters*, 18:525–539.

[Roberts and Penny, 1996] Roberts, S. and Penny, W. (1996). Novelty, confidence and errors in connectionist systems. Technical report, Imperial College, London. TR-96-1.

[Roberts et al., 1994] Roberts, S., Tarassenko, L., Pardey, J., and Siegwart, D. (1994). A validation index for artificial neural networks. In *Proceedings of Int. Conference on Neural Networks and Expert Systems in Medicine and Healthcare*, pages 23–30.

[Rogova, 1994] Rogova, G. (1994). Combining the results of several neural network classifiers. *Neural Networks*, 7(5):777–781.

[Roweis and Ghahramani, 1997] Roweis, S. and Ghahramani, Z. (1997). A unifying review of linear gaussian models. Technical report, Computation and neural systems, California Institute of Technology.

[Ruck et al., 1990] Ruck, D., Rogers, S., Kabrisky, M., Oxley, M.E., and Suter, B.W. (1990). The multilayer perceptron as an approximation to a Bayes optimal discrimination function. *IEEE Transactions on Neural Networks*, 1(4):296–298.

[Rumelhart and McClelland, 1986] Rumelhart, D. and McClelland, J. (1986). *Parallel Distributed Processing: Explorations in the microstructure of cognition*, volume I and II. MIT press.

[Schölkopf, 1997] Schölkopf, B. (1997). *Support Vector Learning*. PhD thesis, Technischen Universität Berlin.

[Schölkopf et al., 1999] Schölkopf, B., Williamson, R., Smola, A., and Shawe-Taylor, J. (1999). SV estimation of a distribution's support. In *NIPS'99*.

[Sharkey and Sharkey, 1995] Sharkey, A. and Sharkey, N. (1995). How to improve the reliability of artificial neural networks. Technical Report CS-95-11, Department of Computer Science, University of Sheffield.

[Smola et al., 1998] Smola, A., Schölkopf, B., and Müller, K. (1998). The connection between regularization operators and support vector kernels. *Neural Networks*, 11:637–649.

[Smolensky et al., 1996] Smolensky, P., Mozer, M., and Rumelhart, D. (1996). *Mathematical perspectives on neural networks*. Lawrence Erlbaum Associates.

[Strang, 1988] Strang, G. (1988). *Linear algebra and its applications.* Harcourt Brace Jovanovich College Publishers.

[Tanigushi and Tresp, 1997] Tanigushi, M. and Tresp, V. (1997). Averaging regularized estimators. *Neural Computation*, 9:1163–1178.

[Tarassenko et al., 1995] Tarassenko, L., Hayton, P., and Brady, M. (1995). Novelty detection for the identification of masses in mammograms. In *Proc. of the Fourth International IEE Conference on Artificial Neural Networks*, volume 409, pages 442–447.

[Tax and Duin, 1998] Tax, D. and Duin, R. (1998). Outlier detection using classifier instability. In Amin, A., Dori, D., Pudil, P., and Freeman, H., editors, *Advances in Pattern Recognition, Lecture notes in Computer Science*, volume 1451, pages 593–601, Berlin. Proc. Joint IAPR Int. Workshops SSPR'98 and SPR'98 Sydney, Australia, Springer.

[Tax and Duin, 1999] Tax, D. and Duin, R. (1999). Data domain description using support vectors. In Verleysen, M., editor, *Proceedings of the European Symposium on Artificial Neural Networks 1999*, pages 251–256. D.Facto, Brussel.

[Tax and Duin, 2000] Tax, D. and Duin, R. (2000). Data descriptions in subspaces. In *Proceedings of the International Conference on Pattern Recognition 2000*, volume 2, pages 672–675.

[Tax et al., 1997] Tax, D., Duin, R., and van Breukelen, M. (1997). Comparison between product and mean classifier combination rules. In Pudil P., Novovicova J, and Grim J, editors, *1st international workshop on statistical techniques in pattern recognition*, pages 165–170. Institute of Information Theory and Automation.

[Tax et al., 2000] Tax, D., van Breukelen, M., Duin, R., and Kittler, J. (2000). Combining multiple classifiers by averaging or multiplying? *Pattern Recognition*, 33(9):1475–1485.

[Tax et al., 1999] Tax, D., Ypma, A., and Duin, R. (1999). Pump failure detection using support vector data description. In Hand, D., Kok, J., and Berthold, M., editors, *Advances in Intelligent Data Analysis*, volume 3, pages 415–425.

[Tipping and Bishop, 1999] Tipping, M. and Bishop, C. (1999). Mixtures of probabilistic principal component analyzers. *Neural Computation*, 11(2):443–482.

[Tumer and Ghosh, 1995] Tumer, K. and Ghosh, J. (1995). Order statistics combiners for neural classifiers. In *Proceedings of the World Congress on Neural Networks.*, pages I:31–34. Washington D.C. INNS Press.

[Tumer and Ghosh, 1996] Tumer, K. and Ghosh, J. (1996). Analysis of decision boundaries in linearly combined neural classifiers. *Pattern Recognition*, 29(2):341–348.

[Ullman, 1978] Ullman, N. (1978). *Elementary statistics, an applied approach.* Wiley and Sons.

[van Breukelen et al., 1997] van Breukelen, M., Duin, R., and Tax, D. (1997). Combining classifiers for the recognition of handwritten digits. In Pudil P., Novovicova J. and Grim J. editors, *1st international workshop on statistical techniques in pattern recognition*, pages 13- 18. Institute of Information Theory and Automation.

[Vapnik, 1995] Vapnik, V. (1995). *The Nature of Statistical Learning Theory.* Springer-Verlag New York, Inc.

[Vapnik, 1998] Vapnik, V. (1998). *Statistical Learning Theory.* Wiley.

[Vesanto et al., 2000] Vesanto, J., Himberg, J., Alhoniemi, E., and Parhankangas, J. (2000). Som toolbox for matlab 5. Report A57, Helsinki University of Technology, Neural Networks Research Centre, Espoo, Finland.

[Wallace and Dowe, 1999] Wallace, C. and Dowe, D. (1999). Minimum message length and kolmogorov complexity. *Computer Journal*, 42(4):270–283.

[Weisstein, 1998] Weisstein, E. (1998). *The CRC concise encyclopedia of mathematics.* CRC Press, Boca Raton, FL. http://mathworld.wolfram.com/.

[Wolpert, 1992] Wolpert, D. (1992). Stacked generalization. *Neural Networks*, 5:241–259.

[Wolpert, 1994] Wolpert, D. (1994). *The Mathematics of Generalization.* Goehring D., Santa Fe Institute.

[Xu et al., 1992] Xu, L., Krzyżak, A., and Suen, C. (1992). Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man and Cybernetics*, 22(3):418–435.

[Ypma and Duin, 1998] Ypma, A. and Duin, R. (1998). Support objects for domain approximation. In *ICANN'98*, Skovde (Sweden).

[Ypma and Pajunen, 1999] Ypma, A. and Pajunen, P. (1999). Rotating machine vibration analysis with second-order independent component analysis. In *Proceedings of the First International Workshop on Independent Component Analysis and Signal Separation, ICA'99*, pages 37–42.

[Ypma et al., 1999] Ypma, A., Tax, D., and Duin, R. (1999). Robust machine fault detection with independent component analysis and support vector data description. In Hu, Y.-H., Larsen, J., Wilson, E., and Douglas, S., editors, *Neural Networks for signal processing IX*, pages 67–76.

[Zadeh, 1965] Zadeh, L. (1965). Fuzzy sets. *Information and control*, 8:338–353.

SUMMARY ONE-CLASS CLASSIFICATION

This thesis treats the problem of one-class classification. It starts with an introduction of the problem of conventional, multi-class classification. Next, it explains the problem of one-class classification, where it is the goal to distinguish between objects from one class and all other possible objects. It is assumed that only examples of one of the classes, the target class, are available. The fact that no examples not belonging to the target class (outliers) are available, complicates the training of a one-class classifier. It is not enough to minimize the number of errors the classifier makes on the target set, but it is also required to minimize in some way the chance that it makes an error on the outlier data. One way to minimize the chance of error on the outlier data, is to minimize the volume what the one-class classifier covers in the feature space.

In the second chapter a new type of one-class classifier is presented, the support vector data description. It models the boundary of the target data by a hypersphere with minimal volume around the data. The boundary is described by a few training objects, the support vectors. Analogous to the support vector classifier, the support vector data description has the ability to replace normal inner products by kernel functions to obtain more flexible data descriptions. Furthermore, the fact that only the boundary of the data is modelled, makes the support vector classifier less dependent on the quality of the sampling of the target class. The SVDD can cope with situations in the exact density of the target class is unknown, but where just the area in the feature space can be estimated.

In chapters three and four several other one-class classifiers are investigated. Three types of one-class classifiers are considered, the density estimators, the boundary methods and the reconstruction methods. On several artificial datasets, each with their own distribution characteristics, the performance of these methods have been evaluated. Finally the one-class classifiers are applied to some real world problems, to investigate which data characteristics can be identified.

Inspired by the gains in performance which have be obtained when normal classifiers are combined, we looked at the possibilities of combining one-class classifiers in the last chapter. Due to the different characteristics of one-class classifiers with respect to conventional two-class classifiers, other results for combining classifiers is expected. One concern is that some one-class classifiers output not a posterior probability for the target class, but output a distance to a model. This distance should be mapped to a posterior probability before the outputs of several one-class classifiers can be combined. The results of experiments show, that combining one-class classifiers often improve results, especially when classifiers

trained on different feature sets are combined.

David M.J. Tax

SAMENVATTING ONE-CLASS CLASSIFICATION

Dit proefschrift behandelt het probleem van "één-klasse klassificatie" (one-class classification). Het begint met een introductie van de conventionele, twee-klasse klassificatie. Vervolgens legt het het probleem van one-class classification uit, waar het de bedoeling is om objecten uit één klasse te onderscheiden van alle andere objecten. Er wordt aangenomen dat alleen voorbeelden van één van de klassen aanwezig zijn, de "doel-klasse" (target class). Het feit dat er geen voorbeelden beschikbaar zijn die niet tot de target class behoren (de uitbijters of "outliers"), compliceert de training van een één-klasse klassificator. Het is niet voldoende dat het aantal fouten dat de klassificator maakt op de target klasse wordt geminimaliseerd, maar het is ook vereist dat op een of andere manier de kans wordt geminimaliseerd dat het een fout maakt op de outliers. Een manier om de kans op een fout op de outlier klasse te minimaliseren, is om het volume dat de één-klasse klassificator inneemt in de kenmerkruimte, te minimaliseren.

In het tweede hoofdstuk hebben we een one-class klassificator, de "support vector data description" (de support vector data beschrijving), gepresenteerd. Het modelleert de rand van de target data met een hyperbol met minimum volume rond de data. Analoog aan de "support vector classifier" (support vector klassificator) heeft de support vector data description de mogelijkheid om normale inproducten te vervangen door kernel functies waardoor flexibelere data beschrijvingen verkregen kan worden. Het feit dat alleen de rand van de data wordt gemodelleerd, maakt dat de support vector data description minder afhankelijk is van de kwaliteit van de trekkingen uit de target klasse. De support vector data description kan omgaan met situaties waarin de exacte dichtheid van de target klasse onbekend is, maar waarin alleen het gebied in de kenmerkruimte kan worden geschat.

In hoofdstukken drie en vier worden verschillende andere klassificatoren onderzocht. Drie types van één-klasse klassificatoren worden bekeken, de dichtheidsschatters, de grens methodes en de reconstructie methodes. Op verschillende kunstmatige datasets, elk met hun eigen karakteristieken, worden de prestaties van de klassificatoren geëvalueerd. Tenslotte zijn de één-klasse klassificatoren toegepast op echte problemen, om te zien welke data karakteristieken kunnen worden geïdentificeerd.

Geïnspireerd door de prestatie-winst die gehaald wordt wanneer verschillende klassificatoren worden gecombineerd, hebben we in het laatste hoofdstuk gekeken naar de mogelijkheden om één-klasse klassificatoren te combineren. Door het verschil in karakter tussen één-klasse klassificatoren en conventionele twee-klass klassificatoren, kunnen andere resultaten verwacht worden. Een zorg is dat sommige klassificatoren geen a posterior kans-

dichtheid opleveren, maar een afstand tot een model. Deze afstand zal getransformeerd moeten worden naar een kansdichtheid voordat de uitvoer van de verschillende klassificatoren gecombineerd kunnen worden. De resultaten van de experimenten laten zien dat het combineren van één-klasse klassificatoren de prestaties vaak verbeteren, in het bijzonder wanneer klassificatoren die op verschillende kenmerk-sets zijn getraind.

David M.J. Tax

CURRICULUM VITAE

David Tax was born in Ede on June 17, 1973. He obtained his VWO diploma at the Marnix College in Ede. In 1991 he went to Nijmegen to study Physics at the University of Nijmegen. He obtained his Masters degree in 1996. The subject of his Masters thesis was "Learning of Structure by Many-Take-All Neural Networks.". The reseach was conducted at the Department of Medical Physics and Biophysics under the supervision of H.J. Kappen.

In 1996 he started his Ph.D. research at the Pattern Recognition Group. This research is titled 'Confidence levels in Neural Networks'. The activities at the Pattern Recognition Group took place under supervision of dr. ir. R.P.W. Duin.

ACKNOWLEDGEMENTS

In the life of a (graduate) student the main focus is on learning and teaching science. Although this might sound pretty dry, this is not just thinking hard, programming and writing. It also includes presentations, discussions, and drinking beers with your colleagues. The large number of topics which is covered in our group makes it possible to regularly discuss the relative merits of the double (or triple) Hough transform, the confocal microscope and the dynamics (or statics) in robo-soccer. Although it is funny to makes jokes about it, it also forces you to think about what you are doing and how to present yourself. It is not only very rewarding to work with equal minded people, spending your time with Mike, Judith, Cris, Bernd, Michiel, Peter, Frank, Gerold and Marjolein after (and during) working hours is always fun. This is not possible when there isn't large support from the rest of the group. Thanks Ted, Bob, Lucas, Albert, Frans, Piet, Pieter, Wim and my predecessors Aarnoud, Stephanie and Eddy! Our group keeps alive the history of 'gezelligheid'.

I especially want to thank the foreign people in our group. They showed me what problems foreigners encounter when they try to adapt to Dutch customs and habits. They made me aware of what I take for granted and what courage and perseverance you need to survive in a completely strange country. I humbly apologize for our bread, but I will defend Dutch coffee with my life.[1]

The most important person I would like to thank is my supervisor, Bob Duin. Having Bob as supervisor gives you the freedom to pursue your individual interest. Furthermore, he gives you the feeling that you can do something useful, he points out, very accurately, how to avoid obvious (and not so obvious) scientific mistakes and he is *always* available when you have something on your mind (may it good or bad). He saved me many years in my promotion, and, most importantly, kept me enthousiastic for science. Thanks Bob!

The most valuable times during my promotion were the discussions with Bob, my room mates Dick, Alex, Ela and, almost room mates, Pavel and Marina. All topics could be discussed, from the most basic to the most esoteric. During the survival of conferences (from Heijen to Barcelona), dagjes-uit, courses and cinema's, I got intimate views of these people and this makes discussions far more interesting. The possibility to discuss not only the present research topics with them but also more general problems of life (how to communicate with bricks?) gives me the reassurance that we are not mere colleagues. This is why I'm doing all this and I tremendiously enjoyed it. Thanks guys!

Finally, my parents and my sister, where would I be without them? (Not here) Thanks for everything!

[1] At this time I should not forget the *tea*: Ela, dziękuję bardzo za herbatę i twój uśmiech!

INDEX