Neurocomputing 72 (2009) 1859–1869

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Minimum spanning tree based one-class classifier

Piotr Juszczak [a,*], David M.J. Tax [a], Elżbieta Pękalska [b], Robert P.W. Duin [a]

[a] Information and Communication Theory Group, Delft University of Technology, Delft, The Netherlands
[b] School of Computer Science, University of Manchester, Manchester, UK

ARTICLE INFO

Article history:
Received 2 July 2007
Received in revised form
27 January 2008
Accepted 29 May 2008
Communicated by H. Yu
Available online 16 July 2008

Keywords:
One-class classification
Novelty detection
Minimum spanning tree
Recognition
Class oriented description

ABSTRACT

In the problem of one-class classification one of the classes, called the target class, has to be distinguished from all other possible objects. These are considered as non-targets. The need for solving such a task arises in many practical applications, e.g. in machine fault detection, face recognition, authorship verification, fraud recognition or person identification based on biometric data.

This paper proposes a new one-class classifier, the minimum spanning tree class descriptor (MST_CD). This classifier builds on the structure of the minimum spanning tree constructed on the target training set only. The classification of test objects relies on their distances to the closest edge of that tree, hence the proposed method is an example of a distance-based one-class classifier. Our experiments show that the MST_CD performs especially well in case of small sample size problems and in high-dimensional spaces.

1. Introduction

In the problem of one-class classification [29,17,41,27,14,19] one of the classes, called the *target class*, has to be distinguished from all other possible objects, also called *non-targets*. The need for solving such a task arises in many practical applications. Examples are any type of fault detection [46] or target detection such as face detection in images, abnormal behaviour, disease detection [40], person identification based on biometric data or authorship verification [23]. The problem of one-class classification is characterised by the presence of a target class, e.g. a collection of face images of a particular person. The goal is to determine a proximity function of a test object to the target class such that the resembling objects are accepted as targets and non-targets are rejected. It is assumed that a well-sampled training set of the target objects is available, while no (or very few) non-target examples are present. The reason for this assumption is practical since non-targets may occur only occasionally or their measurements might be very costly. Moreover, even when non-targets are available in a training stage, they may not always be trusted. They may be badly sampled, with unknown priors and ill-defined distributions. In essence, non-targets are weakly defined as they may appear as any kind of deviation or anomaly from the target examples, e.g. images of a face of non-target people or images of arbitrary (non-face) objects. Still, one-class classifiers need to be trained in such a way that the errors on both target and non-target classes are taken into account.

Many one-class classifiers have been proposed so far; see [41,19] for a survey. They often rely on strong assumptions concerning the distribution of objects, such as a normal distribution of the target class [6,2,37] or a uniform distribution of the non-target class [41]. Following the later assumption, the training of classifiers is based on a minimisation of the volume of a one-class classifier (which is the volume captured by the classifier's boundary) such that the error on the target class does not increase [42,39,4,33]. Usually such classifiers can be applied to any distribution as they do not make assumptions on the target distribution, but they may need to estimate many parameters. Examples are support vector data description (SVDD) [42] or auto-encoder neural net [17,28].

In this paper we propose a non-parametric classifier which is based on a graph representation of the target training data, aiming to capture the underlining data structure. The basic elements of the proposed one-class classifier are the edges of the graph. Graph edges can be considered as an additional set of virtual target objects. These additional objects, in turn, can help to model a target distribution in high-dimensional spaces and in small sample size problems. This enriches the representation of relations in the data. Additionally, we can look at graph edges as a set of possible transformation paths that allow one to transform one target object into another within the domain of the target class.

The layout of this paper is as follows. Section 2 presents the formal notation and describes the framework of one-class classification. In Section 3, a data descriptor based on the minimum

* Corresponding author.
E-mail address: p.juszczak@tudelft.nl (P. Juszczak).

0925-2312/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2008.05.003

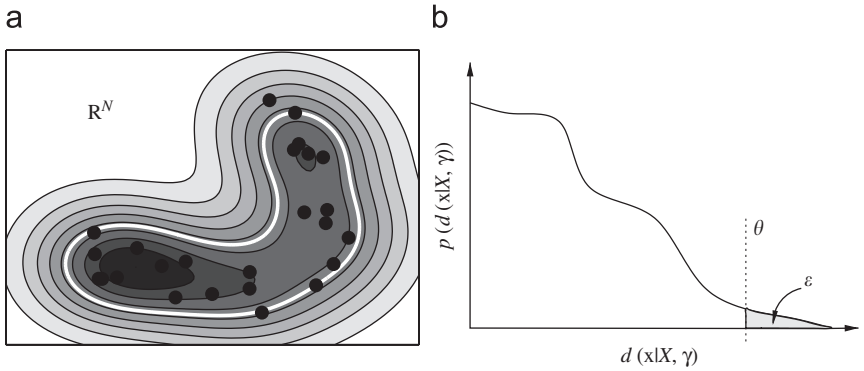


Fig. 1. (a) An example of one-class classifier. The boundary of the one-class classifier depends on the choice of the threshold θ. Different boundaries are drawn as isolines. (b) One-dimensional distribution of $d(\mathbf{x}|X,\gamma)$. The threshold θ is set to reject ε fraction of the training set X. Objects in X are distributed according to $p(d(\mathbf{x}|X,\gamma))$.

spanning tree (MST) is introduced. Section 3.2 discusses a possible complexity parameter which gives a handle to describe the data complexity and to simplify the classifier. Section 4 discusses the related work. Section 5 explores both advantages and disadvantages of the proposed classifier based on a set of experiments conducted on both artificial and real-world data. The final conclusions are presented in Section 6.

2. One-class classifiers

One-class classifiers are trained to accept target examples and reject non-targets. It is assumed that during training no or only a few non-target objects are available. In a part of the further discussion we will also assume a presence of outliers during training. Outliers may, e.g. arise from measurement errors and can be considered as mislabelled target objects in the training set.

Let $X=\{\mathbf{x}_i|\,\mathbf{x}_i\in\mathbb{R}^N, i=1,\ldots,n\}$ be a training set in an N-dimensional vector space drawn from the target distribution. Assume that a one-class classifier is sought that characterises this target class. In general, all one-class classifiers can be cast in the following form:

$$h(\mathbf{x}|X,\gamma)=\mathscr{I}(d(\mathbf{x}|X,\gamma)<\theta)=\begin{cases}1 & \text{if } \mathbf{x} \text{ is classified as a target,}\\ 0 & \text{if } \mathbf{x} \text{ is classified as a non-target,}\end{cases} \tag{1}$$

where θ is a specified threshold and $\mathscr{I}(\cdot)$ is an indicator function. The model h relies on the dissimilarity of a vector $\mathbf{x}$ to the training-target data X. In general, h can also be based on the similarity measure, for which the opposite sign ($>$) will be used above. Furthermore, γ determines the complexity of the model h. The threshold θ is optimised to reject a certain, usually user-specified fraction ε of the target class such as 0.05, for instance.

The fraction ε has to be determined from a given application. If one expects the presence of outlier objects (mislabelled target objects in the training set), setting $\varepsilon>0$ makes the descriptor h more robust. On the other hand, when there are no outliers in the training set, ε indicates either desired or allowed maximal error on the target class. For example, one may specify the maximum number of allowed false alarms in a machine condition monitoring. In such a case, θ is optimised on the target class under the assumption of a uniformly distributed non-target class. Given a fixed target acceptance rate, $(1-\varepsilon)$, the threshold θ is derived from the target training set such that the one-class classifier accepts the fraction $(1-\varepsilon)$ of the target examples; see Fig. 1(b). That is, given n training samples, θ is determined such that

$$\min \quad \theta \tag{2a}$$

$$\text{s.t.} \quad \frac{1}{n}\sum_{i=1}^{n}\mathscr{I}\{d(\mathbf{x}|X,\gamma)\geqslant\theta\}=\varepsilon, \tag{2b}$$

where $d(\mathbf{x}|X,\gamma)$ is estimated on the training set X. A similar equation, with a different sign ($\leqslant$), can be used when a similarity is estimated by a classifier.

Apart from the threshold θ, the performance of a one-class classifier is influenced by the complexity parameter γ of the classifier, e.g. a number of nodes in neural networks or prototypes in a nearest neighbour. In general, a complexity parameter γ can be determined during training if the errors on both the target and non-target classes are estimated, e.g. by using cross-validation. However, in one-class classification only target examples are available. Therefore, several criteria have been proposed to determine γ based on a single class. For instance, one can assume a uniform distribution of non-target objects and select γ which minimise both the error on the target class and the volume of the one-class classifier [43].

Many principles used in construction of two-class or multi-class classifiers can also be applied for solving one-class classification problems. Fig. 2 shows some examples of the one-class classifiers trained on a 2D toy problem.[2] A more detailed description of the methods can be found in [41,19]. The most common approach to one-class classification is probabilistic in nature [2]. Basically, the target class is modelled by a probability density function (`pdf`). Specifying a suitable threshold on such a `pdf` allows one to determine the class boundary, hence the rejection point. A test sample is judged as a member of the target class if its estimated probability is higher than the given threshold. This can be realised by a parametric method such as a mixture of Gaussians (`MoG`), Fig. 2(b) [37] or even a single Gaussian `pdf` equipped with a threshold, Fig. 2(a) [6], or by a non-parametric method such as a Parzen density estimator based on

[2] Solid lines denote classifier boundaries in Figs. 2(a)–(i), while dotted lines illustrate the principles based on which classifiers are constructed in Figs. 2(a)–(g). In subplots (a) and (b), the dotted lines indicate probability levels for a single Gaussian and a mixture of three Gaussians, respectively. In subplot (c), the dotted lines illustrate the width of a smoothing parameter. In subplot (d) the data have been projected onto two features and the dotted lines show densities estimated per feature; the densities are multiplied to create the classifier (solid line). In subplots (e) and (f), the dotted lines show the three means and the averaged nearest neighbour distance, respectively. Finally, in subplot (g), the first `PCA` direction of the data is shown. The last two plots show kernel-based one-class classifiers for which the underlying principles cannot be easily depicted in the input space.

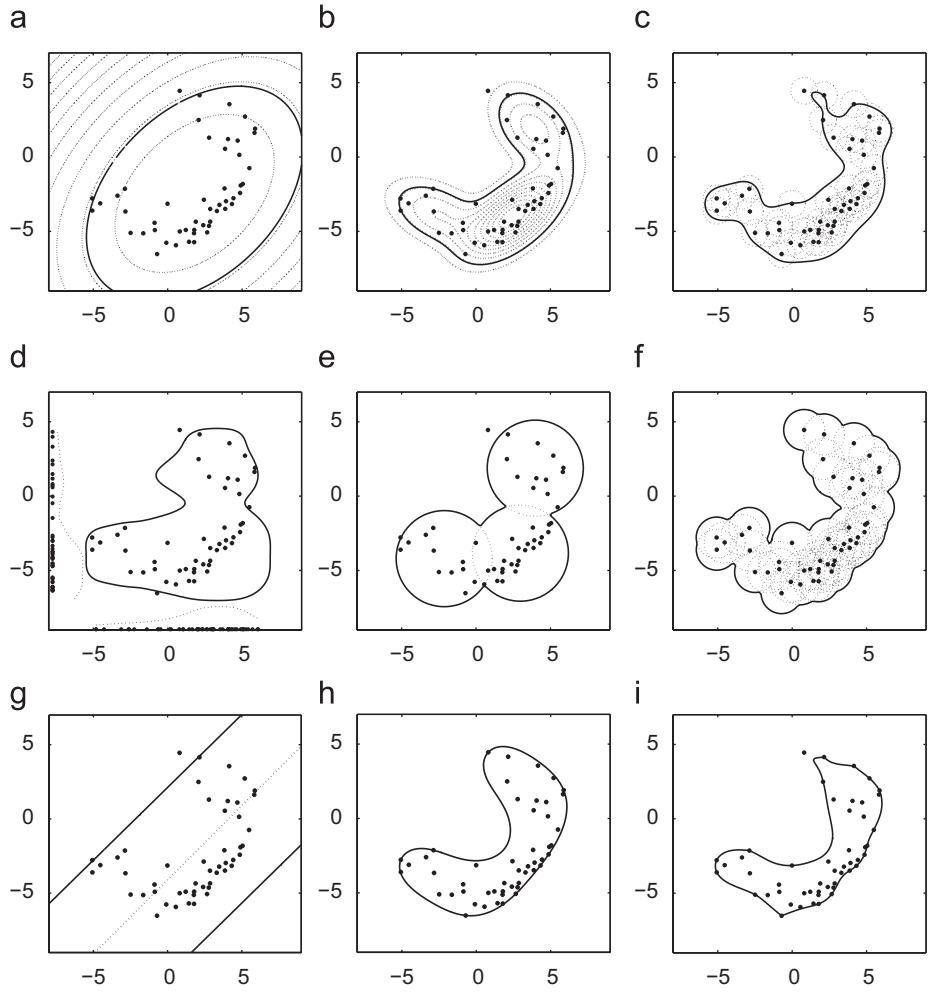


Fig. 2. Examples of one-class classifiers marked by solid lines. The principles behind the creation of some one-class classifiers are denoted by dotted lines. The threshold was set to $\varepsilon = 0.01$. Concerning the 1-nearest neighbour (1-NN), the threshold was applied to the average nearest neighbour distance. (a) Gaussian density, (b) MoG density, (c) Parzen density, (d) Naïve Parzen, (e) *k*-means, (f) 1-NN, (g) PCA, (h) SVDD and (i) LPDD.

the Gaussian kernel, Fig. 2(c) [31] or *k*-nearest neighbour estimators, Fig. 2(f) [22]. Other popular approaches include neural networks with auto-encoders [17,28] and self-organising maps [30], as well as clustering techniques such as *k*-means, Fig. 2(e) [18] or *k*-centres [16].

Alternative methods proposed to tackle the problem of one-class classification do not rely on a probabilistic approach. Instead, they aim to minimise the volume of the target domain, which may be cast out in the form of linear programming, Fig. 2(i) [4,25,33] or quadratic programming, Fig. 2(h) [42,39]. In particular, the SVDD, a one-class classifier defined in the framework of kernel methods, has been introduced in [42]. In the simplest case, this classifier finds the smallest *N*-sphere that encloses all objects from the target class; other flexible descriptions are enabled by the use of kernels, Fig. 2(h).

In this paper, we propose a distance-based one-class classifier that is based on a local neighbourhood relations but it takes into account also the global structure of the data. The description of the target class relies on a graph structure of the MST.

3. Description of a target class by an MST

Let $\{\mathbf{x}_i, \mathbf{x}_j\} \in X \subset \mathbb{R}^N$ be two examples from a target class. If these two examples describe two similar objects in reality, they should be neighbours in the representation space $\mathbb{R}^N$. We assume that not only these examples but points from their proper neighbourhoods also belong to the target class. For example, if we assume the continuity within the target class in $\mathbb{R}^N$, then there exists a continuous transformation between these two examples. This means that we can find a transformation for which all points lying along such a path will also belong to the target class. For simplicity, we can assume that the changes between neighbouring objects can be approximated by linear transformations:

$$\mathscr{T}(\mathbf{x}_i, \boldsymbol{\lambda}_{ij}) = \mathbf{x}_i + \lambda_{ij}(\mathbf{x}_j - \mathbf{x}_i), \quad 0 \leqslant \lambda_{ij} \leqslant 1. \tag{3}$$

Several transformations are, however, possible since the training set has usually more than two objects. To select a set of most probable transformations we relate the length of each transformation vector to its probability of existence. To satisfy our previous assumption about continuity in the target class we need to select only $(n-1)$ linear transformations between the target training objects.

Assume that X is a training set of n target examples $\mathbf{x}_i \in \mathbb{R}^N$, $i = 1, 2, \ldots, n$. Let $G = (X, \mathscr{E})$ be a fully connected, undirected graph defined on X and let $\mathscr{E} = \{e_{ij}\}$ be a set of all edges $e_{ij} = (\mathbf{x}_i, \mathbf{x}_j)$. To each edge e_{ij} we additionally assign a weight w_{ij} related by its length, i.e. $w_{ij} \equiv \|e_{ij}\| = \|\mathbf{x}_i - \mathbf{x}_j\|$. Note that the edges e_{ij} specify our linear transformations. Our task is to find a subgraph $\mathscr{G}$, without loops, which connects all the vertices such that the total weight, or the total length of the edges, is minimum. This gives us the most probable set of transformations. Since the weights are

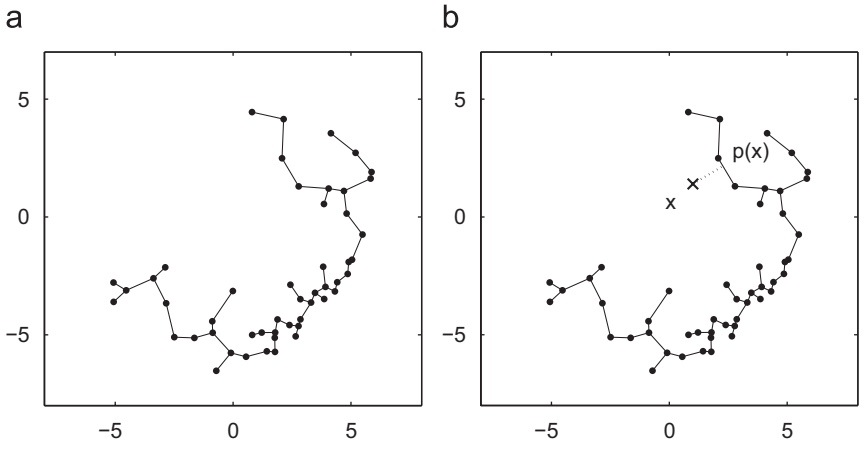


Fig. 3. (a) Minimum spanning tree computed on a 2D training target set denoted by dots. (b) The distance of a new object $\mathbf{x}$ to an MST_CD is measured as the shortest distance between $\mathbf{x}$ and its projection $\mathbf{p}(\mathbf{x})$ onto the closest edge in the MST.

symmetric, it is sufficient to consider only the edges e_{ij} for which $i>j$ (or, alternatively, the edges e_{ij} for which $i<j$). Hence, $|\mathscr{E}| = \frac{n^2-n}{2}$. This can equivalently be formulated as a problem of finding $(n-1)$ edges that form a tree with a minimum total weight:

$$\begin{aligned} \min \quad & \sum_{e\in\mathscr{G}} \|e\| \\ \text{s.t} \quad & |\mathscr{G}| = n-1, \ \mathscr{G}\subset\mathscr{E} \\ & \exists!\ \text{path}(\mathbf{x}_i,\mathbf{x}_j), \ \forall\{\mathbf{x}_i,\mathbf{x}_j\}\in X, \ i>j, \end{aligned} \tag{4}$$

where $\exists!$ denotes a unique existence and $\text{path}(\mathbf{x}_i,\mathbf{x}_j) \equiv \{\mathbf{x}_i,\mathbf{x}_{i+1},\ldots,\mathbf{x}_j\}$. Note that the transformations $\mathscr{T}$ are defined by a set of subsequent linear transformations on the path from $\mathbf{x}_i$ to $\mathbf{x}_j$.

Optimisation (4) is equivalent to computing the MST [13] for a given training set. Hence, training a descriptor on the target class can be formulated as finding the MST on a set of training objects; see also Fig. 3(a). As a result, the proposed classifier is called the minimum spanning tree class descriptor (MST_CD).

Several algorithms have already been proposed for finding MST, as it arises in many theoretical and practical problems. The most popular algorithms, and ones of the oldest, are Prim's [35] and Kruskal's [24] algorithms. Prim's algorithm starts from an arbitrary vertex, while Kruskal's algorithm starts from the shortest edge. The former procedure requires that the next edge to be added is incident with a vertex in a temporary tree, whereas the latter procedure just adds the next shortest edge that does not form a loop.

3.1. Recognition with MST_CD

The training set is finite, so we can assume that not only the edges of MST belong to the target class but also their neighbourhoods. A new object $\mathbf{x}$ is considered as a target if it lies in the vicinity of the determined MST_CD. This is judged by the shortest distance to the tree, i.e. the smallest distance to the set of the $(n-1)$ edges; see Fig. 3(b). The projection of $\mathbf{x}$ onto a line defined by the vertices $\{\mathbf{x}_i,\mathbf{x}_j\}$ is

$$\mathbf{p}_{e_{ij}}(\mathbf{x}) = \mathbf{x}_i + \frac{(\mathbf{x}_j-\mathbf{x}_i)^{\mathrm{T}}(\mathbf{x}-\mathbf{x}_i)}{\|\mathbf{x}_j-\mathbf{x}_i\|^2}(\mathbf{x}_j-\mathbf{x}_i). \tag{5}$$

If $\mathbf{p}_{e_{ij}}(\mathbf{x})$ lies on the edge $e_{ij}=(\mathbf{x}_i,\mathbf{x}_j)$, then the distance $d(\mathbf{x}|e_{ij})$ between $\mathbf{x}$ and the edge e_{ij} is computed as the Euclidean distance between $\mathbf{x}$ and its projection $\mathbf{p}_{e_{ij}}(\mathbf{x})$. Otherwise, $d(\mathbf{x}|e_{ij})$ is derived as the shortest Euclidean distance to one of the vertices $\{\mathbf{x}_i,\mathbf{x}_j\}$

$$\begin{aligned} &\textbf{if} \quad 0\leqslant\frac{(\mathbf{x}_j-\mathbf{x}_i)^{\mathrm{T}}(\mathbf{x}-\mathbf{x}_i)}{\|\mathbf{x}_j-\mathbf{x}_i\|^2}\leqslant 1 \ \textbf{then} && \text{(a)} \\ &\qquad d(\mathbf{x}|e_{ij}) = \|\mathbf{x}-\mathbf{p}_{e_{ij}}(\mathbf{x})\| && \text{(b)} \\ &\textbf{else} \\ &\qquad d(\mathbf{x}|e_{ij}) = \min\{\|\mathbf{x}-\mathbf{x}_i\|,\|\mathbf{x}-\mathbf{x}_j\|\} && \text{(c)} \\ &\textbf{end} \end{aligned} \tag{6}$$

The set of equations (6) determines the distance from a test object $\mathbf{x}$ to a single edge e_{ij}. The distance is computed in two ways, (6b) or (6c):

(1) if the projection $\mathbf{p}_{e_{ij}}(\mathbf{x})$ is between $\mathbf{x}_i$ and $\mathbf{x}_j$ then the distance to the edge is computed as the distance between $\mathbf{x}$ and its projection; Eq. (6b);
(2) otherwise the distance is computed as a nearest neighbour distance between $\mathbf{x}$ and $\mathbf{x}_i$ or $\mathbf{x}_j$; Eq. (6c).

To determine if $\mathbf{p}_{e_{ij}}(\mathbf{x})$ is between $\mathbf{x}_i$ and $\mathbf{x}_j$ we check relation (6a). If this relation is true then $\mathbf{p}_{e_{ij}}(\mathbf{x})$ is between $\mathbf{x}_i$ and $\mathbf{x}_j$.

The distance of a new object $\mathbf{x}$ to the target class defined by X is computed as the minimum distance to the set of $(n-1)$ edges of the MST_CD

$$d_{\text{MST\_CD}}(\mathbf{x}|X) = \min_{e_{ij}\in\text{MST}} d(\mathbf{x}|e_{ij}). \tag{7}$$

The decision whether $\mathbf{x}$ belongs to the target or non-target class is based on the threshold θ. This threshold is set on the distance $d_{\text{MST\_CD}}(\mathbf{x}|X)$ such that

$$h_{\text{MST\_CD}} \equiv \mathscr{I}(d_{\text{MST\_CD}}(\mathbf{x}|X)\leqslant\theta). \tag{8}$$

Note, however, that according to the above definition, the distance $d_{\text{MST\_CD}}(\mathbf{x}|X)$ equals zero for all training objects. Therefore, all target training objects will be accepted by the MST_CD for any positive θ. Hence, the threshold θ cannot correspond to an error on the training class as computed in Eq. (2). This is the same situation as for the k-NN classifier; see Fig. 2(f), where the threshold is computed as the average nearest neighbour distance instead of estimating it by (2). In our case we derive the threshold θ based on a quantile function of the distribution of edge weights $w_{ij}=\|e_{ij}\|$ in the given MST.

Denote $\tilde{\mathbf{e}} = (\|e_{(1)}\|,\|e_{(2)}\|,\ldots,\|e_{(n)}\|)$ as a sorted sequence of scalars such that $\|e_{(1)}\|\leqslant\|e_{(2)}\|\leqslant\cdots\leqslant\|e_{(n)}\|$. The quantile function,

specifying θ, is defined as

$$\theta \equiv \mathcal{Q}_\alpha(\tilde{\mathbf{e}}) = \|e_{([\alpha n])}\|, \tag{9}$$

where $[a]$ returns the nearest integer of a and $\alpha \in [0,1]$. Thus, $\mathcal{Q}_0(\tilde{\mathbf{e}})$ is the minimum of $\tilde{\mathbf{e}}$, $\mathcal{Q}_1(\tilde{\mathbf{e}})$ is the maximum and $\mathcal{Q}_{0.5}(\tilde{\mathbf{e}})$ the median. Therefore, θ is determined from a particular set of distances between the neighbouring objects in the MST. In this case, the MST_CD has a single parameter θ to be estimated. The majority of one-class classifiers have at least two parameters to be estimated: θ and the complexity parameter γ.

Fig. 4 shows a 2D toy problem (as considered before) with example MST_CD classifiers. The threshold θ equals $\mathcal{Q}_{0.5}(\tilde{\mathbf{e}})$, $\mathcal{Q}_{0.7}(\tilde{\mathbf{e}})$ or $\mathcal{Q}_{0.95}(\tilde{\mathbf{e}})$, respectively. Comparing the boundaries of the MST_CD to other one-class classification models presented in Fig. 2, we can observe that a manifold structure is more emphasised in case of the MST_CD. Also, the description of the target class is tighter when compared to the density-based classifiers or the 1-NN from Fig. 2. The proposed MST_CD method is an example of a domain-based classifier [19]. In such a case, it is required that the training examples are drawn from a domain of a class (e.g. as specified by a geometrical form in the input space), but not necessarily according to their pdf's as density-based classifiers demand. As a result, the proposed one-class classifier is less sensitive to sampling, which can be beneficial in frameworks such as active learning [20].

3.2. Sparse MST_CD

The data description shown in Fig. 4 may be judged to be more complex than strictly necessary; the original MST_CD can therefore be simplified. By removing edges and vertices from the MST_CD we obtain a sparse data description. This can be done without a significant change in the original description.

To find a set of superfluous edges we first compute the longest path in the MST, i.e. the first *principal direction* of the graph. This is the path between two vertices that yields maximum length. The second longest path, found by excluding edges from the first principal direction, represents the second principal direction, and so on. Fig. 5 shows the first and second principle directions in an MST example.

The tree representation of the data can therefore be simplified by considering only a few principle directions. All paths in the graph can be computed using either Dijkstra's or Jordan's algorithm [7]. To simplify the graph description of the target data in this paper, we use Jordan's algorithm since it is more computationally efficient. Moreover, it allows one to compute all paths in the MST simultaneously.

Given an initial set $\mathscr{E} = \{e_{ij}\}$ of $\frac{n^2-n}{2}$ edges in a fully connected undirected data graph, we have simplified the description of the target data to $(n-1)$ edges of the corresponding MST. In addition, we have introduced a complexity parameter γ which specifies the number of principle directions used to describe the target data. γ can be estimated as a fraction of the total length of edges in the MST_CD. Now, since we may have disconnected objects, the threshold θ can be set, if possible, such that ε is an error on the training set. ε equals the fraction of objects that lie outside the class descriptor.

Fig. 6 shows a sparse MST_CD with a fixed value of θ determined such that $\varepsilon = 0.1$, equal to the error on the training set, and three values of the complexity parameter. We can notice that a sparse one-class classifier, $\gamma = 1$, captures the data characteristics almost as good as the MST_CD with the maximum complexity; it equals to the complete MST. This is because the intrinsic dimensionality of the data in Fig. 6 is one. Hence, a single principle direction is sufficient to capture the characteristics of our 2D toy problem. This can easily be seen if we plot the ratio of the length of each principal direction (PD_i) to the total sum of all edge lengths ($\sum \|PD_i\|$) in the complete MST; see Fig. 7(a). In addition, we also compute this statistics for four UCI repository data sets. These plots provide some insight whether it is beneficial to simplify the graph or not.

If needed, our one-class classifier can be simplified even further by removing superfluous vertices and the associated

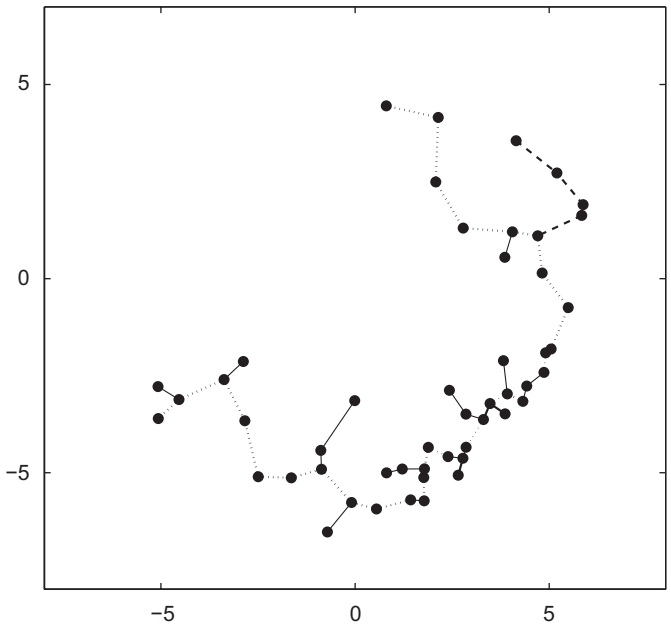


Fig. 5. First (dotted line) and second (dashed line) principal directions in an example MST. Solid lines denote the remaining edges.

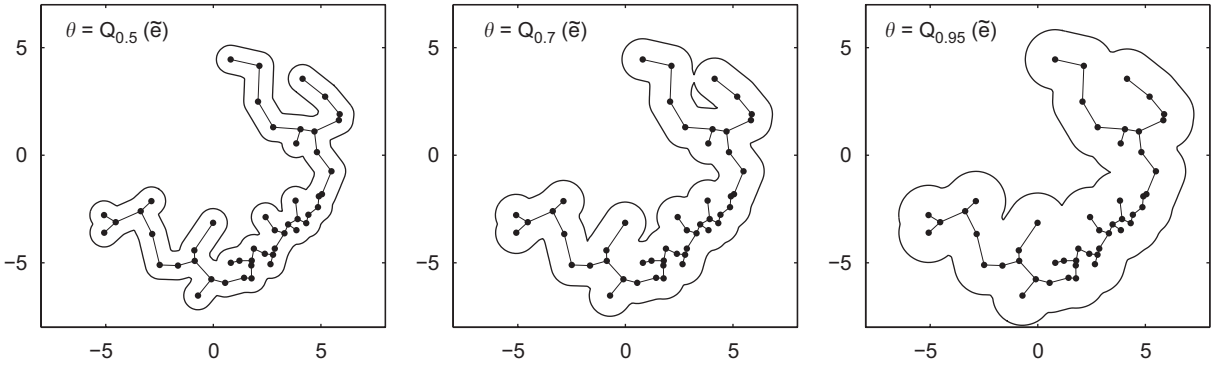


Fig. 4. Example solutions of the MST_CD for different thresholds θ.

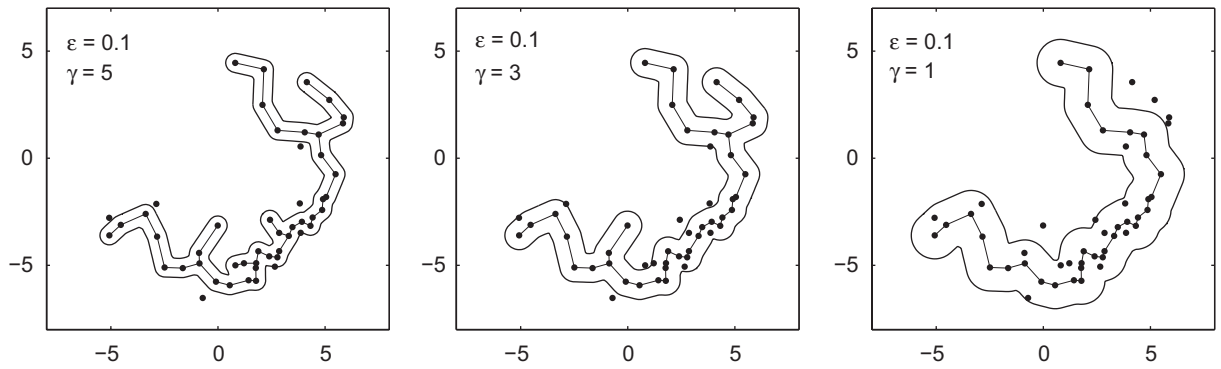


Fig. 6. Examples of a sparse MST_CD. The threshold θ was determined on the target class such that ε equals 0.1 and the complexity γ decreases from left to right as $\gamma = 5, 3$ and 1.

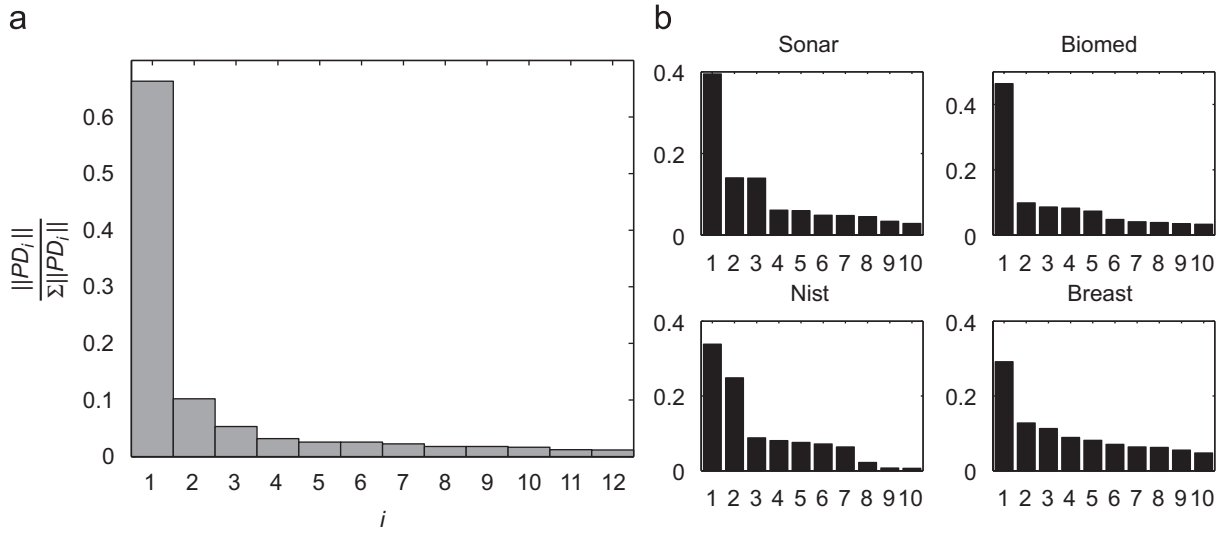


Fig. 7. The ratio of the length of each principle direction to the sum of all edge lengths in the MST: (a) for data shown in Fig. 3 and (b) for four UCI repository data sets.

edges from the MST_CD. Given an existing tree, a vertex $\mathbf{x}_k$ and the two associate edges are removed if the distance between $\mathbf{x}_k$ and its projection onto the new edge e_{ij} is larger than a specified constant. We use Eq. (5) to find the projection of the vertex on the new edge. If for the triple $\{\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k\} \in X$, and the edges $\{e_{ik}, e_{kj}\} \in$ MST, the distance between $\mathbf{x}_k$ and its projection onto e_{ij} is smaller than ε, $\|\mathbf{x}_k - \mathbf{p}_{e_{ij}}(\mathbf{x}_k)\| \leqslant \varepsilon$, then the edges e_{ik} and e_{kj} are replaced by a new edge e_{ij} and $\mathbf{x}_k$ is removed from the class descriptor. The complexity of such an MST classifier can be adjusted to the complexity of the data.

Similar to other distance-based and density-based one-class classifiers, our class descriptor may be influenced by outliers in a training set. A small set of outliers in the training set can largely influence a description boundary. If one expects that outliers are present, the MST_CD can be made more robust by removing, e.g. a fraction of the training objects from the MST with the longest edges. In the experiments presented in the paper, we assume no outliers are present in the training set.

4. Related work

The proposed classifier can be related to other known methods. In particular, a multi-class classifier, the nearest feature line method (NFLM) was introduced in [26]. In the NFLM, one describes a training set by a set of lines between all pairs of objects from a particular class. The new object is classified into one of the classes from the training set based on the distance to the nearest line from the set. It has been shown there that the NFLM performs well in face recognition problems in comparison to the nearest neighbour methods. However, the NFLM method has some disadvantages. A test object can be close to a line determined by two training objects, which may lie far from the bulk of the training set. Hence, the method is not suitable for one-class classification problems since the volume of the resulting descriptors may be infinite. In our MST_CD, we focus on the description of the target class only and we compute the distance to the closest edge of the (sparse) MST of that class. Moreover, in the NFLM, the testing stage can be computationally expensive for a large training set, since for each test object it is required to compute distances to $\frac{1}{2}\sum_{i=1}^{C} n_i(n_i - 1)$ lines, where n_i is a number of objects in the ith class and C is the total number of classes. In our method, we compute distances to at most $(n - 1)$ edges.

The idea of describing a data set by a graph is not entirely new. During a training of a self-organising map (SOM) one usually fits a 2D or 3D grid to represent an underlying distribution of the data. As a result, a set of vertices is obtained, while the corresponding graph edges are neglected. Moreover, the size and the (usually low) dimensionality of the grid have to be specified beforehand. Here, we describe the target data by using the entire graph, the

MST, including the edges. Moreover, we introduced a graph description where the structure of the graph is learnt from a given training set and not specified beforehand.

Several clustering algorithms proposed in the literature also rely on graphs. Examples include CURE, ROCK and CHAMELEON; see [21] for a survey. The CURE algorithm is very similar to the MST with the possibility of data pre-selection and scaling. The CHAMELEON procedure is a more advanced two-stage clustering method. In the first stage, small clusters are created based on the k-NN criterion, and they are agglomerated in the second stage. The decision to join clusters is based on the length of edges of graphs within and between the small clusters.

Similar to the proposed MST_CD we could also describe the data manifold by non-parametric density methods based on the Parzen or nearest neighbour estimators. However, the finite training size makes it usually difficult to accurately approximate the density of a target class by non-parametric models. Another possibility is to use subspace methods such as kernel-PCA [38]. However, the parameters of such descriptions (parameters of the kernel function as well as the intrinsic dimensionality of the data) have to be estimated from the given training set or specified by the user. Several techniques that are used for visualisation of high-dimensional data can also be applied here. For example, the techniques like isomap [45], local linear embedding [36] or multi-dimensional scaling [12]. However, they may be difficult to apply to new data without re-computing their models. In addition, these types of algorithms often require several parameters to be set beforehand. Moreover, since the volumes of the resulting descriptions (as defined by the boundaries of the one-class classifier) are infinite, this may lead to high non-target acceptance rates.

5. Experiments

To study the performance of one-class classifiers, a receiver–operator characteristics (ROC) curve is often used [3]. It is a function of the true positive ratio (target acceptance) versus the false positive ratio (non-target acceptance). Of course, examples of non-target objects are necessary to evaluate it. These are available in a validation stage only. In order to compare the performance of various classifiers, the area under the ROC curve (AUC) measure can be used [3]. It computes the AUC, which is the total performance of a one-class classifier integrated over all thresholds. The larger the AUC is, the better is the performance of a one-class classifier. The AUC value less than 0.5 indicates that a particular classifier is worse than random guessing.

Except for the Colon data in Table 1, the experiments are performed on the data sets taken from the UCI repository [15]. The Colon data set is an example of gene expression data and a very small sample size problem; the details can be found in [1]. All multi-class problems are transformed to one-class classification problems by setting a chosen class as a target set and all remaining classes as non-targets. Tables 1 and 2 compare the performance of the MST_CD and several existing one-class classifiers. The comparison is based on the AUC measure. The target class was randomly split into equal parts between the training and test sets. All one-class classifiers were trained on the target data only and tested on both target data and all available non-target data. The experiments were repeated 20 times and the results are averaged.

The name of a class that was set to a target class is indicated in the parentheses below the name of the data set. The total sizes of the target class and the non-target class, denoted as $|X|/|\bar{X}|$, and the dimensionality N of data are also provided. Concerning the Parzen and k-NN one-class classifiers, the complexity parameter γ was optimised by the leave-one-out error [10]. For the k-means, k-centres, MoG, SOM, SVDD, MPM and LPDD methods, γ was optimised using the consistency approach [44] with the threshold θ determined such that $\varepsilon = 0.1$. One-class PCA retains 0.95 variance of the training set. For the CHAMELEON descriptor, the 3-NN is used as the first stage clustering and the threshold is computed as the average length of the cluster edges. In MST_CD, the complete MST is used (hence γ is maximal). The best and no significantly worse than the best result in Tables 1 and 2 are marked bold.

Table 1
The value of AUC with standard deviations (in parentheses) for a number of one-class classifiers trained on small sample size problems from the UCI repository

Data set	*Spectf*	*Sonar*	*Nist*	*Arrhythmia*	*Concordia*	*Colon*
Target class	(0)	(mines)	(0)	(normal)	(2)	(normal)
$\lvert X\rvert/\lvert\bar{X}\rvert$	95/254	111/97	200/1800	237/183	400/3600	22/44
Dim. N	44	60	256	278	1024	1908
Classifier				AUC		
Gauss	83.3(3.3)	68.0(3.1)	90.3(1.3)	60.6(0.6)	80.3(1.7)	70.4(1.1)
MoG	77.6(3.1)	70.4(3.5)	50(0.0)	57.7(16.6)	50(1.1)	50.0(0.0)
Naïve Parzen	90.2(3.7)	53.2(3.9)	83.6(5.5)	77.4(0.7)	84.6(0.7)	70.0(1.5)
Parzen	87.9(2.7)	**80.5(3.1)**	55.1(3.2)	57.7(16.6)	50.2(2.2)	36.4(22.4)
k-Means	92.3(1.7)	69.8(3.7)	97.6(0.7)	76.6(0.6)	86.2(2.5)	66.8(3.1)
1-NN	92.6(2.9)	76.3(4.3)	86.6(5.7)	76.0(0.8)	90.1(0.8)	71.3(3.3)
k-NN	92.3(1.5)	69.6(4.8)	**98.0(0.5)**	76.0(0.8)	**90.1(0.9)**	71.3(3.3)
Auto-encoder	81.7(6.2)	59.6(6.5)	83.2(2.8)	52.2(2.1)	51.2(1.5)	50.0(0.0)
PCA	90.1(3.0)	69.6(3.3)	**98.2(0.8)**	**80.7(1.0)**	82.4(0.4)	70.7(1.6)
SOM	97.5(2.1)	**80.1(3.4)**	96.9(0.8)	77.2(0.7)	88.7(2.0)	68.2(2.6)
MST_CD	**98.1(2.6)**	**81.1(3.1)**	**98.3(0.6)**	**79.6(0.6)**	**91.1(0.1)**	**73.3(3.0)**
k-Centres	90.9(1.6)	66.8(4.1)	96.9(0.7)	76.7(1.6)	81.5(3.6)	68.4(2.9)
SVDD	**97.8(3.3)**	76.1(3.2)	0.3(0.2)	58.1(16.4)	12.1(1.1)	36.4(22.4)
MPM	**98.0(7.4)**	78.5(3.0)	0.3(0.2)	77.1(0.5)	**90.1(0.6)**	50.0(0.0)
LPDD	93.4(3.3)	63.6(2.7)	0.3(0.2)	57.7(16.6)	86.4(0.4)	41.8(20.0)
CHAMELEON	94.4(0.7)	77.8(1.0)	95.8(2.1)	76.0(0.8)	80.7(0.4)	39.1(5.1)

The mean and standard deviation were calculated from 20 hold-out repetitions.

Table 2
The value of AUC with standard deviations (in parentheses) for a number of one-class classifiers trained on low-dimensional problems from the UCI repository

Data set	*Biomed*	*Liver*	*Ecoli*	*Diabetes*	*Breast*	*Abalone*
Target class	(normal)	(healthy)	(periplasm)	(present)	(benign)	(classes 1–8)
$\lvert X\rvert/\lvert\bar{X}\rvert$	127/67	145/200	52/284	500/268	241/458	1407/2770
Dim. N	5	6	7	8	9	10
Classifier				AUC		
Gauss	90.0(0.4)	58.6(0.5)	**92.9(0.3)**	**70.5(0.3)**	82.3(0.2)	86.1(0.2)
MoG	91.2(0.9)	**60.7(0.6)**	**92.0(0.4)**	67.4(0.3)	78.5(1.3)	85.3(0.5)
Naïve Parzen	**93.1(0.2)**	61.4(0.7)	**93.0(0.8)**	67.9(0.3)	**96.5(0.4)**	85.9(0.4)
Parzen	90.0(1.1)	59.0(0.3)	92.2(0.4)	67.6(0.4)	72.3(0.5)	86.3(0.1)
k-Means	87.8(1.2)	57.8(1.0)	89.1(1.6)	65.9(0.7)	84.6(3.5)	79.2(1.1)
1-NN	89.1(0.8)	59.0(0.9)	90.2(0.9)	66.7(0.7)	69.4(0.6)	86.5(0.1)
k-NN	89.1(0.8)	59.0(0.9)	90.2(0.9)	66.7(0.7)	69.4(0.6)	86.5(0.1)
Auto-encoder	85.6(2.2)	56.4(0.9)	87.8(1.0)	59.8(1.8)	38.4(0.9)	82.6(0.3)
PCA	89.7(0.5)	54.9(0.5)	66.9(1.1)	58.7(0.2)	30.3(1.0)	80.2(0.1)
SOM	88.7(0.8)	59.6(0.7)	89.0(1.1)	69.2(0.7)	79.0(2.3)	81.4(0.3)
MST_CD	89.8(1.0)	58.0(0.9)	89.7(0.9)	66.9(0.7)	75.6(1.8)	**87.5(0.1)**
k-Centres	87.8(2.4)	53.7(4.1)	86.3(1.2)	60.6(1.6)	71.5(12.4)	76.0(0.8)
SVDD	2.2(0.3)	4.7(1.4)	89.4(0.8)	57.7(9.8)	70.0(0.6)	80.6(0.1)
MPM	79.2(5.7)	58.7(0.9)	80.2(0.5)	65.6(0.7)	69.4(0.6)	59.4(0.1)
LPDD	86.5(2.6)	56.4(2.6)	89.6(0.5)	66.8(0.7)	80.0(0.5)	69.7(0.1)
CHAMELEON	72.7(1.9)	58.0(0.9)	75.8(1.6)	65.1(1.0)	66.9(0.8)	70.6(0.4)

The mean and standard deviation were calculated from 20 hold-out repetitions.

The data sets in Table 1 represent a collection of small sample size problems, in which the number of training samples is smaller or similar to the number of dimensions (remember that only half of $|X|$ reported in Tables 1 and 2 is used for training). We can observe that the MST_CD always performs best or one of the best, however, not always significantly better than other methods. In high-dimensional spaces, density-based classifiers do not perform well. This can be observed from the AUC performances presented in Table 1. The performance of 50.0 that appears for the classifiers like Parzen, auto-encoder or MoG indicates that all test objects were classified as non-targets. This is caused by insufficient information in the training set to correctly estimate the parameters of those classifiers. This also leads to the worse than random performance of the kernel-based methods (SVDD, MPM and LPDD), especially for the *Nist* data.

In Table 1, the *Nist* and *Concordia* sets describe hand-written digit recognition problems. Objects in these data sets are represented by vectors in pixel-based vector spaces. Digits represented in such a way are usually distributed along lower-dimensional manifolds since pixels corresponding to the corners, edges of images do not contribute to the information about classes. Therefore, our one-class classifier describes the target class better by emphasising the low-dimensional manifold structure of the data. The average of two vectors representing digits in the pixel-based space, e.g. 1 s rotated clockwise and anticlockwise, does not necessarily create a representation of a new digit 1. Therefore, the data subspace of the digits is not a lower-dimensional hyperplane, but rather a nonlinear manifold. Since the MST_CD is able to model that it outperforms the PCA on these data sets.

The data sets in Table 2 represent a collection of low-dimensional problems, in which the size of the training set is larger than the data dimensionality. We can observe that most one-class classifiers, including the MST_CD, perform well on such problems. However, on small sample size problems reported in Table 1, the MST_CD performs among first five best classifiers. In addition, we can notice that the performance of a PCA based one-class classifier, which performed well in high-dimensional problems, now deteriorates. For the *Biomed*, *Ecoli*, *Liver* and *Abalon* data sets, the MST_CD classifier performs insignificantly worse than the best classifier. Comparing the sizes of the training sets with data dimensionality, these sets can be considered as moderate sample size problems.

Analysing the results from Table 1, we can conclude that the MST_CD tends to outperform other one-class classifiers in (very) small sample size problems and for high-dimensional data. The results from Table 2 also confirm that the MST_CD performs well in moderate sample size problems.

Optimisation of complexity parameter γ is an important issue when performance of a classifier is considered. Several algorithms have been proposed to optimise γ [43,44]. In Table 3 we present few interesting examples of how AUC changes for different values of γ. For *Nist* data set increasing complexity parameter also increases the performance of MST_CD. The data set is an example of a high-dimensional manifold-like structure. Therefore, MST_CD, with a high complexity, as well as other subspace-based classifiers, performs well on such data. Similar results were obtained for Concordia data set.

Table 3
AUC versus complexity parameter γ

γ / Data set	1	3	5	7	MST
			AUC		
Nist	90.1(0.3)	91.0(0.5)	93.2(1.3)	92.1(0.2)	98.3(0.6)
Sonar	80.8(3.2)	79.5(2.7)	80.9(1.8)	83.1(2.5)	81.1(3.1)
Breast	92.1(1.2)	89.3(1.7)	86.3(0.5)	85.2(1.5)	75.6(1.8)

On the other hand for *Sonar* data set a wide range of complexity parameters gives similar performance. To explain this phenomenon see Fig. 7(b). *Sonar* data set has a long first principal direction. Almost an entire data set structure is captured in the first direction. Therefore, adding additional directions changes performance slightly.

The last example is the most interesting one. For the *Breast* data set reducing complexity of MST_CD increases its performance. It is well known that classes in this data set are normally distributed. Decreasing γ in MST_CD changes the shape of a decision boundary making it more ellipsoidal; Fig. 4. Therefore, the performance for *Breast* data set improves as the complexity of the classifier decreases. Similar behaviour of MST_CD performance has been observed for *Breast* and *Biomed* data sets. We can conclude that a reduction of complexity of MST_CD may improve the performance for data sets from Table 2.

5.1. Comparison of one-class classifiers in a classifier projection space

The AUC performance is an overall measure which does not provide full information about the performance and comparison of classifiers. The same values of AUC for two classifiers may be computed on different objects in a test set. In order to get more insight into the behaviour of various one-class classifiers, we can compare classifiers by comparing their assignments for the given test data. One of the simplest measure is the disagreement coefficient. It counts the number of assignments that a pair of classifiers disagree, normalised by the cardinality of the test set. This defines a dissimilarity between two classifiers with respect to the given data.

Given a set of L one-class classifiers, an $L \times L$ disagreement matrix reflects the pairwise relations between all examined classifiers. Now a vector space representation can be found such that the computed disagreements are preserved as distances in that space as well as possible. This is achieved by a 2D linear projection, a variant of multi-dimensional scaling [8]. The resulting space is called a classifier projection space [32], since the vectors in that space represent the classifiers, and the Euclidean distances between them reflect the disagreements between the classifiers. It helps one to visually judge the similarity between classifiers, a step towards the classifier tribology.

Fig. 8 allows us to judge the similarity of the predictions made by the MST_CD to these of other one-class classifiers. The two plots shown there visualise the characterisation of one-class classifiers by their label disagreements on the test set, consisting of both target and non-target objects. Two $(L+1) \times (L+1)$ disagreement matrices are computed. They are based on the predicted assignments of $L = 16$ one-class classifiers applied to the test data from Tables 1 and 2, respectively. Since we know the true labels for the test data, they are used as the assignments of the oracle, i.e. of the perfect $(L+1)$-th class descriptor. The oracle is denoted as 'O' and added as a comparison. Remember that this presentation is *not* based on the AUC as the classifiers' disagreements are calculated for a single threshold of (0.1) on the ROC curve.

Dashed circular lines represent the same performance, hence classifiers with the same number of misclassified objects in a test set, as they reflect the distance to the oracle O. We can see from these plots that the MST_CD is the most similar to the NN, k-NN and the k-means, while it is very different from the density-based one-class classifier as well as the support vector based classifiers, SVDD, LPDD and MPM. This is, of course judged in the sense of predicted

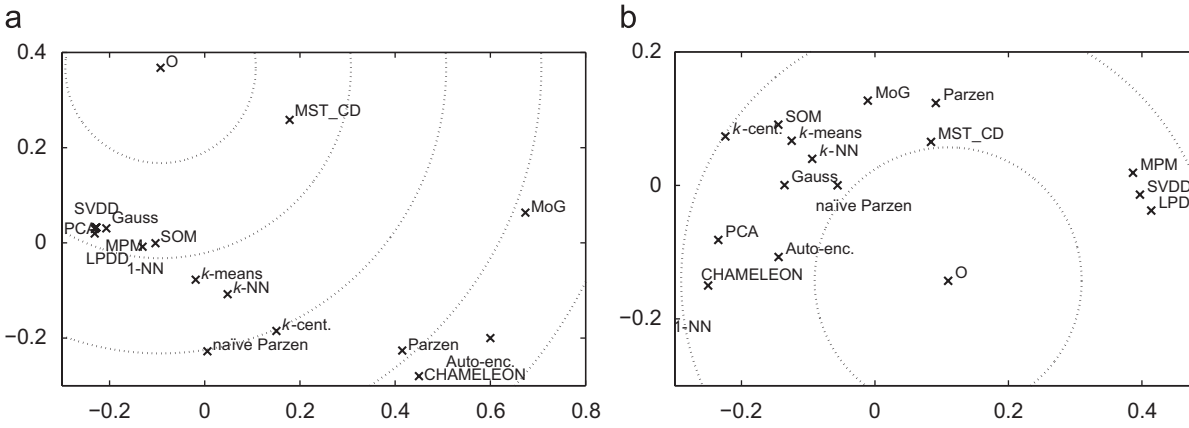


Fig. 8. Classifier projection space: the characterisation of one-class classifiers by their disagreements for the data sets from (a) Table 1 and (b) Table 2. The oracle, i.e. a classifier that predicts true labels, is denoted by O.

membership. Although the values of AUC for the MST_CD, PCA and SOM methods are quite similar in Table 1, the way the assignments are made, indicate the classifiers are really different. This is reflected in large distances between their vector representations in the classifier projection space, as depicted in Fig. 8.

By observing Fig. 8(a), we can notice that the bulk of classifiers built by the methods such as the SVDD, MPM, PCA, Gauss, 1-NN or SOM has a similar performance to that of the MST_CD. This is judged by the similar Euclidean distance of each class descriptor to the oracle O. The labels predicted by these classifiers are, however, very different than those of the MST_CD as reflected in the, large distance of the MST_CD to this bulk. Fig. 8(b) shows that the majority of classifiers performs well on these well-sampled data sets, but still the distances from MST_CD to other classifiers are large in the classifier projection space, which indicates that the MST_CD descriptor is of a different kind. In this case, the MST_CD is most similar to the Parzen class descriptor. Since the MST_CD is quite diverse from the standard one-class classifiers, it may be considered as a valuable member in the committee where one-class classifiers need to be combined.

We have noticed that the MST_CD descriptor performs well for high-dimensional data, e.g. on the digit data represented as vectors in a pixel-based space in the *Nist* and *Concordia* data sets. The other group of problems that is usually characterised by high dimension and small sample size are problems encountered in bioinformatics, similar to *Colon* data set in Table 1.

Here, we focus on the microarray gene expression data. Microarrays are a new technology to investigate the expression levels of thousands of genes simultaneously. They present new statistical problems because such problems are usually characterised by a huge dimensionality and a very small number of examples. To describe the behaviour of one-class classifiers we select the *Leukemia* data with 25 target objects in 3571-dimensional space and the *Mates* data with 46 target objects in 4919-dimensional space. These data sets were originally used in [11]. The threshold of the one-class classifiers was set such that no training objects are rejected as the data are extremely small. The related classifier projection space is shown in Fig. 9. Although the presented results suggest that the MST_CD performs well on microarray data, the reader should be aware that such claims need much more support than just two publicly available data.

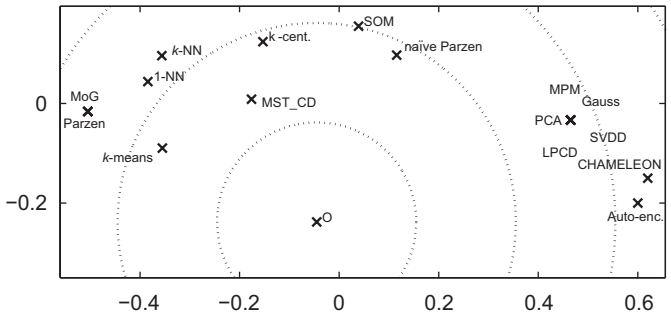


Fig. 9. Classifier projection space for one-class classifiers trained on the *Mates* and *Leukemia* data.

5.2. Computational complexity

To determine the MST for a given data set we can use either Prim's or Kruskal's algorithm. Their computational complexity is $\mathcal{O}(|\mathscr{E}|\log|X|)=\mathcal{O}(\frac{n^2-n}{2}\log n)$. To find the threshold θ from (9), at most $\mathcal{O}(n\log n)$ operations are needed as it involves only sorting of edge lengths. Hence, the computational complexity of MST_CD based on the full MST is $\mathcal{O}(\frac{n^2-n}{2}\log n)$. Recently, authors [34,5] have proposed even less computationally expensive algorithms for determining the MST, in the order of $\mathcal{O}(\frac{n^2-n}{2}\alpha(\frac{n^2-n}{2},n))$, where α is an inverse of Ackermann's function.[3] Pettie and Ramachandran [34] have also shown that the proposed algorithm has a high probability to be linear in n. Moreover, efficient parallel algorithms exist to compute the MST [9], making it a computationally attractive data descriptor for large data sets. It should also be noted that the computational complexity of training the MST_CD is lower than the computational complexity of many other machine learning algorithms, which can be in the order of $\mathcal{O}(n^3)$, for the support vector machine (SVM), for instance.

The sparse MST_CD involves the computation of principle directions. To compute all principle directions using Jordan's algorithm requires $(\mathcal{O}(n^2\log n))$ operations [7]. Therefore, the complexity is still quadratic in the number of objects.

Table 4 presents averaged running time in seconds for several one-class classifiers. The complexity of classifiers were set in the same way as in the experiments in Tables 1 and 2 and the threshold was set to 0.1. Experiments were performed on MacBook Pro, 2 GHz Intel, 2 GB RAM and all classifiers were implemented in Matlab. The time was measured for two data sets: *Diabetes* with 250 training, 518 test objects and 8 features

[3] The Ackermann function $A(i,j)$ is a function of two parameters whose values grow extremely fast. It is defined recursively as $A(0,j)=j+1$ for $j\geqslant 0$, $A(i,0)=A(i-1,1)$ for $i>0$ and $A(i,j)=A(i-1,A(i,j-1))$ for $i,j>0$. The inverse Ackermann function $\alpha(i,j)$ is a function of two parameters whose values grow extremely slowly. It is defined as $\alpha(m,n)=\min\{i\geqslant 1: A(i,\lfloor\frac{m}{n}\rfloor)>\log_2 n\}$.

Table 4
Running time of classifiers averaged over 20 trials for two data sets, *Diabetes* and *Concoria*

Gauss	MoG	N. Parzen	Parzen	*k*-Means	1-NN	*k*-NN	A. Encoder	PCA	SOM	MST_CD	*k*-Centres	SVDD	MPM	LPDD	CHAMELEON
Diabetes															
0.5	0.7	2.4	0.2	0.1	0.1	0.1	120	0.3	195.9	1.3	4.1	0.8	0.5	7.1	50.4
0.3	0.1	1.2	0.2	0.1	0.2	0.2	0.5	0.1	0.1	0.5	0.2	0.2	0.2	0.3	0.4
Concordia															
2.4	6504	121.4	0.6	0.6	0.2	0.2	7401	2.8	636.8	1.3	1.0	0.5	0.3	2.4	121
1.3	17107	354.5	3.4	0.4	2.1	2.3	4092	8.8	1.7	2.9	1.0	1.3	1.2	1.4	10.1

The first row shows training and the second testing time.

and *Concordia* data set with 200 training, 3800 test objects and 1024 features. We can see that running times for MST_CD almost do not depend on dimensionality of data sets. Similar to other distance-based classifiers, like 1-NN *k*-centres, the computational complexity of MST_CD is mostly driven by the number of objects. The same holds for kernel-based classifiers like LPDD and SVDD. On the other hand density-based classifiers, e.g. a single Gauss, MoG or neural network based auto-encoder depend on both the number of objects and the number of dimensions. The computation of all principal paths for *Diabetes* took about 20 s and for Concordia about 12 s. Comparison of the optimisation of σ using the consistency criterion [44] for support vector-based SVDD took 210 and 185 s, respectively.

An implementation of the MST_CD has been included into DD_TOOLS, a data description toolbox for one class classification problems. It is freely available for academic purposes and can be downloaded from www.prtools.org.

6. Conclusions

This paper proposes a new one-class classifier based on the minimum spanning tree (MST). The complexity of the classifier equals the complexity of the MST and the threshold is determined as a fraction on a distribution of the edge lengths in the MST. The basic elements of the classifier are the edges of the tree which can be considered as additional virtual elements that capture more characteristics of the training objects. As an extension, we also propose a way to reduce the complexity of the classifier either by selecting only the first few principal directions or by removing superfluous edges.

The presented classifier performs well in high-dimensional spaces and in small sample size problems in comparison to other existing one-class classifiers. Since the MST_CD is constructed on different principles than other one-class classifiers, it becomes a valuable member of a committee in the task of combing classifiers. Possible extensions of the MST_CD are the one-class classifiers based on different graph structures such as the minimum stainer tree, for instance.

Acknowledgement

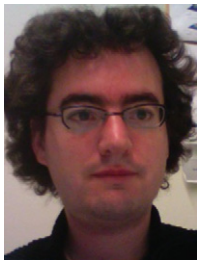
This work was partly supported by the Dutch Organisation for Scientific Research (NWO).

References

[1] U. Alon, N. Barkai, D.A. Notterman, K. Gish, S. Ybarra, D. Mack, A.J. Levine, Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays, Proc. Natl. Acad. Sci. 96 (1999) 6745–6750.
[2] C.M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, Walton Street, Oxford OX2 6DP, 1995.
[3] A.P. Bradley, The use of the area under the ROC curve in the evaluation of machine learning algorithms, Pattern Recognition 30 (7) (1997) 1145–1159.
[4] C. Campbell, K.P. Bennett, A linear programming approach to novelty detection, in: Neural Information Processing Systems, 2000, pp. 395–401.
[5] B. Chazelle, A minimum spanning tree algorithm with inverse-Ackermann type complexity, J. ACM 47 (6) (2000) 1028–1047.
[6] C.K. Chow, On optimum recognition error and reject tradeoff, IEEE Trans. Inf. Theory IT-16 (1) (1970) 41–46.
[7] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, MIT Press, Cambridge, MA, 1990.
[8] T.F. Cox, M.A.A. Cox, Multidimensional Scaling, Chapman & Hall, 1994.
[9] F. Dehne, S. Gotz, Practical parallel algorithms for minimum spanning trees, in: The 17th IEEE Symposium on Reliable Distributed, 1998, p. 366.
[10] R.P.W. Duin, On the choice of the smoothing parameters for Parzen estimators of probability density functions, IEEE Trans. Comput. C-25 (11) (1976) 1175–1179.
[11] T.R. Golub, D.K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J.P. Mesirov, H. Coller, M.L. Loh, J.R. Downing, M.R. Caligiuri, C.D. Bloomfield, E.S. Lander, Molecular classification of cancer: class discovery and class prediction by gene expression monitoring, Science 285 (1999) 531–537.
[12] J.C. Gower, Metric and Euclidean properties of dissimilarity coefficients, J. Classification 3 (1986) 5–48.
[13] R.L. Graham, P. Hell, On the history of the minimum spanning tree problem, Ann. Hist. Comput. 7 (1) (1985) 43–57.
[14] S. Harmeling, G. Dornhege, D.M.J. Tax, F. Meinecke, K.-R. Mueller, From outliers to prototypes: ordering data, Neurocomputing 69 (13–15) (2006) 1608–1618.
[15] S. Hettich, C.L. Blake, C.J. Merz, UCI repository of machine learning databases, 1998 ⟨http://www.ics.uci.edu/~mlearn/MLRepository.html⟩.
[16] D. Hochbaum, D. Shmoys, A best possible heuristic for the k-center problem, Math. Oper. Res. 10 (2) (1985) 180–184.
[17] N. Japkowicz, Concept-learning in the absence of counter-examples: an autoassociation-based approach to classification, Ph.D. Thesis, New Brunswick Rutgers, The State University of New Jersey, 1999.
[18] M.F. Jiang, S.S. Tseng, C.M. Su, Two-phase clustering process for outliers detection, Pattern Recognition Lett. 22 (6–7) (2001) 691–700.
[19] P. Juszczak, Learning to recognise. A study on one-class classification and active learning, Ph.D. Thesis, Delft University of Technology, 2006, ISBN: 978-90-9020684-4.
[20] P. Juszczak, R.P.W. Duin, Uncertainty sampling for one-class classifiers, in: N. Chawla, N. Japkowicz, A. Kolcz (Eds.), ICML-2003 Workshop: Learning with Imbalanced Data Sets II, 2003, pp. 81–88.
[21] G. Karypis, E. Han, V. Kumar, Chameleon: a hierarchical clustering using dynamic modeling, IEEE Comput. 32 (8) (1999) 68–75.
[22] E.M. Knorr, R.T. Ng, V. Tucakov, Distance-based outliers: algorithms and applications, VLDB J. Very Large Data Bases 8 (3–4) (2000) 237–253.
[23] M. Koppel, J. Schler, Authorship verification as a one-class classification problem, in: International Conference on Machine Learning, 2004, pp. 489–495.
[24] J.B. Kruskal, On the shortest spanning subtree of a graph and the travelling salesman problem, Am. Math. Soc. 7 (1) (1956) 48–50.
[25] G.R.G. Lanckriet, L. El Ghaoui, M.I. Jordan, Robust novelty detection with single-class MPM, in: S. Becker, S. Thrun, T. Obermayer (Eds.), Advances in Neural Information Processing Systems, vol. 15, MIT Press, Cambridge, MA, 2003, pp. 905–912.
[26] S.Z. Li, J. Lu, Face recognition using the nearest feature line method, Neural Networks 10 (2) (1999) 439–443.
[27] L. Manevitz, M. Yousef, One-class svms for document classification, J. Mach. Learn. Res. 2 (2002) 139–154.
[28] L. Manevitz, M. Yousef, One-class document classification via neural networks, Neurocomputing 70 (7–9) (2007) 1466–1481.
[29] M.R. Moya, M.W. Koch, L.D. Hostetler, One-class classifier networks for target recognition applications, in: Proceedings World Congress on Neural Networks, Portland, OR, 1993, International Neural Network Society, pp. 797–801.
[30] L. Parra, G. Deco, S. Miesbach, Statistical independence and novelty detection with information preserving nonlinear maps, Neural Comput. 8 (1996) 260–269.
[31] E. Parzen, On the estimation of a probability density function and mode, Ann. Math. Stat. 33 (1962) 1065–1076.

[32] E. Pękalska, M. Skurichina, R.P.W. Duin, A discussion on the classifier projection space for classifier combining, in: J. Kittler, F. Roli (Eds.), Multiple Classifier Systems, Lecture Notes in Computer Science, vol. 2364, Springer, Berlin, 2002, pp. 137–148.
[33] E. Pękalska, D.M.J. Tax, R.P.W. Duin, One-class LP classifier for dissimilarity representations, in: Neural Information Processing Systems, 2003, pp. 761–768.
[34] S. Pettie, V. Ramachandran, An optimal minimum spanning tree algorithm, J. ACM 49 (1) (2002) 16–34.
[35] R.C. Prim, Shortest connection networks and some generalisations, Bell Syst. Tech. J. (1957) 1389–1410.
[36] S. Roweis, L. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326.
[37] S.R. Sain, H.L. Gray, W.A. Woodward, M.D. Fisk, Outlier detection from a mixture distribution when training data are unlabeled, Bull. Seismol. Soc. Am. 89 (1999) 294–304.
[38] B. Schölkopf, S. Mika, A.J. Smola, G. Rätsch, K.R. Müller, Kernel PCA pattern reconstruction via approximate pre-images, in: L. Niklasson, M. Bodén, T. Ziemke (Eds.), Proceedings of the 8th International Conference on Artificial Neural Networks, Springer, Berlin, 1998, pp. 147–152.
[39] B. Schölkopf, R.C. Williamson, A.J. Smola, J. Shawe-Taylor, J.C. Platt, Support vector method for novelty detection, in: T.K. Solla, S.A. Leen, K.R. Müller (Eds.), Neural Information Processing Systems, 2000, pp. 582–588.
[40] L. Tarassenko, P. Hayton, M. Brady, Novelty detection for the identification of masses in mammograms, in: International Conference on Artificial Neural Networks, 1995, pp. 442–447.
[41] D.M.J. Tax, One-class classification, Ph.D. Thesis, Delft University of Technology, 2001.
[42] D.M.J. Tax, R.P.W. Duin, Support vector domain description, Pattern Recognition Lett. 20 (11–13) (1999) 1191–1199.
[43] D.M.J. Tax, R.P.W. Duin, Uniform object generation for optimizing one-class classifiers, J. Mach. Learn. Res. (2001) 155–173.
[44] D.M.J. Tax, K.R. Müller, A consistency-based model selection for one-class classification, in: International Conference on Pattern Recognition, IEEE Computer Society, Los Alamitos, CA, 2004, pp. 363–366.
[45] J. Tenenbaum, V. de Silva, J. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (5500) (2000) 2319–2323.
[46] A. Ypma, R.P.W. Duin, Support objects for domain approximation, in: International Conference on Artificial Neural Networks, Springer, Berlin, 1998, pp. 719–724.

Piotr Juszczak studied physics at the Wrocław University of Technology, Poland and computer science at the National University of Ireland, Galway. He received Master degree in 2001 with the thesis "Automatic recognition of arrhythmia from ECG signal". He received his PhD in 2006 from the Information and Communication Theory Group at Delft University of Technology, under the supervision of R.P.W. Duin. The title of his PhD thesis is "Learning to recognise. Study on one-class classification and active learning". Since 2006 he is a research associate at the Institute for Mathematical Science at Imperial College London. His main research interest involves theoretical and practical aspects of machine learning especially one-class classification, domain-based classification and enhancement of classification models by unlabelled data.



David M.J. Tax studied physics at the University of Nijmegen, The Netherlands in 1996, and received Master degree with the thesis "Learning of structure by Many-take-all Neural Networks". After that he had his PhD at the Delft University of Technology in the Pattern Recognition group, under the supervision of R.P.W. Duin. In 2001 he promoted with the thesis 'One-class classification'. After working for two years as a Marie Curie Fellow in the Intelligent Data Analysis group in Berlin, at present he is post doc in the Information and Communication Theory group at the Delft university of Technology. His main research interest is in the learning and development of outlier detection algorithms and systems, using techniques from machine learning and pattern recognition. In particular the problems concerning the representation of data, simple and elegant classifiers and the evaluation of methods have focus.

Elżbieta Pękalska is a purpose-driven, creative and inspiring researcher, teacher, facilitator and mentor. She studied computer science at the University of Wrocław, Poland. In the years 1998 - 2004 she worked on both applied and fundamental projects in pattern recognition and machine learning at the Delft University of Technology, The Netherlnds. In 2005 she obtained a cum-laude PhD degree, under the supervision R.P.W. Duin, for her seminal work on dissimilarity-based learning methods, aka generalised kernel approaches. Currently, she is an Engineering and Physical Sciences Research Council postdoctoral fellow at the University of Manchester, UK. She is passionate about the learning process and learning strategies. This not only includes intelligent learning from data and sensors, but also humans on their development paths. Some of her key questions refer to the issue of representation, learning and combining paradigms and the use of proximity and kernel in the learning from examples.

Robert P.W. Duin studied applied physics at Delft University of Technology in the Netherlands. In 1978 he received the Ph.D. degree for a thesis on the accuracy of statistical pattern recognizers. In his research he included various aspects of the automatic interpretation of measurements, learning systems and classifiers. Between 1980 and 1990 he studied and developed hardware architectures and software configurations for interactive image analysis. After this period his interest was redirected to neural networks and pattern recognition.

At this moment he is an associate professor of the Faculty of Electrical Engineering, Mathematics and Computer Science of Delft University of Technology. His present research is in the design, evaluation and application of algorithms that learn from examples. This includes neural network classifiers, support vector machines and classifier combining strategies. Recently he started to investigate alternative object representations for classification and became thereby interested in dissimilarity based pattern recognition and in the possibilities to learn domain descriptions. Additionally he is interested in the relation between pattern recognition and consciousness.