Available online at www.sciencedirect.com
SCIENCE DIRECT®
Neurocomputing 69 (2006) 1608–1618
NEUROCOMPUTING
www.elsevier.com/locate/neucom

From outliers to prototypes: Ordering data

Stefan Harmeling[a,b,*], Guido Dornhege[a], David Tax[c], Frank Meinecke[a], Klaus-Robert Müller[a,b]

[a]*Fraunhofer FIRST.IDA, Kekuléstrasse 7, 12489 Berlin, Germany*
[b]*Department of Computer Science, University of Potsdam, August-Bebel-Strasse 89, 14482 Potsdam, Germany*
[c]*Delft University of Technology, Information and Communication Theory Group, P.O. Box 5031, 2600 GA, Delft, The Netherlands*

Received 1 July 2004; received in revised form 21 May 2005; accepted 24 May 2005
Available online 1 December 2005
Communicated by S. Hochreiter

Abstract

We propose simple and fast methods based on nearest neighbors that order objects from high-dimensional data sets from typical points to untypical points. On the one hand, we show that these easy-to-compute orderings allow us to detect outliers (i.e. very untypical points) with a performance comparable to or better than other often much more sophisticated methods. On the other hand, we show how to use these orderings to detect prototypes (very typical points) which facilitate exploratory data analysis algorithms such as noisy nonlinear dimensionality reduction and clustering. Comprehensive experiments demonstrate the validity of our approach.

Keywords: Outlier detection; Novelty detection; Ordering; Noisy dimensionality reduction; Clustering; Nearest neighbors

1. Introduction

Exploratory data analysis tries to find simple representations of high-dimensional data that best reflect the underlying structures. There are few robust methods that can be used for this purpose in high dimensions. In fact, most real-world data sets are spoiled by outliers arising from many different processes, e.g. measurement errors or miss-labeled samples. So it is necessary to remove outliers from the data to avoid erroneous results.

In the statistics literature, a large emphasis is put on the problem of outlier detection in *univariate* data [2,18]. In univariate data the objects are trivially ordered, which eliminates the problem of finding a one-dimensional measure for characterizing the typicality of an object. The main challenge is then to decide where to set the threshold to distinguish between genuine and outlier objects.

For the problem of outlier detection in multivariate data, more complicated models have to be applied in order to impose an ordering on the data. A well known method from the statistics community is the minimum covariance determinant (MCD) estimator [29]: find h observations out of n, such that its covariance matrix has the smallest determinant. The objects are then ordered according to their Mahalanobis distance to the data mean. Although this method is very robust, it is not very flexible because it only fits a Gaussian distribution to the data. More flexible density models include the Parzen density model [3] or the mixture of Gaussians [31]. The Parzen density can be approximated by defining the support of a data set by fitting balls of fixed size around the training set [12,1]. Unfortunately, density estimation in high-dimensional spaces is difficult, and in order to reliably estimate the free parameters, the models have to be restricted significantly in complexity.

From the pattern recognition/machine learning field more heuristic methods originate, for instance, neural

*Corresponding author. Fraunhofer FIRST.IDA, Kekuléstrasse. 7, 12489 Berlin, Germany.
E-mail addresses: harmeli@first.fhg.de (S. Harmeling), dornhege@first.fhg.de (G. Dornhege), davidt@first.fhg.de (D. Tax), meinecke@first.fhg.de (F. Meinecke), klaus@first.fhg.de (K.-R. Müller).

0925-2312/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2005.05.015

network models [23,19] or models which are inspired by the support vector classifiers [32,7,34]. They avoid performing the often very difficult density estimation, and directly fit a decision boundary around the data, but are often not simple to implement and optimize. Also the outputs of traditional two-class classifiers can be used for outlier detection [38], thus focusing on the outliers from the perspective of the classification problem.

Ref. [20] studies distance-based outliers which are defined with respect to two parameters p and D: a data point x is a distance-based outlier, abbr. $DB(p, D)$-outlier, if at least fraction p of the other points lies greater than distance D from x. The $DB(p, D)$-outlier are global outliers, because D and p are chosen for all data points. If the data consists of several clusters with different variances, it can be difficult to choose a single D which is appropriate. Thus, methods have been developed which focus on local properties of the data, e.g. local outlier factors (LOF, see [6]). LOF introduces an outlier index which is based on a sophisticated theory of "local reachability" and nearest neighbors, which gives rise to a somewhat convoluted index. The outlier indices proposed in this paper are defined in terms of nearest neighbors as well, but are designed to be as simple and straightforward as possible.

Most of the existing methods implicitly imply certain definitions of what an outlier actually is, which are often not explicitly stated. In this paper, we call data points outliers if their true probability density is very low. Obviously, the difficulty of this particular notion is that the true probability density is unknown and it is a challenge to obtain a reasonable estimate—especially in high dimensions. However, the indices proposed in this paper, some of which have been previously used for outlier detection in [28,14], coarsely approximate the probability density. Thus these indices are in principle applicable to all settings that assume outliers to be data points in sparse regions.

Notice that most methods mentioned above provide an ordering of objects in a data set, according to their typicality. Very untypical objects are candidates to be labeled as outliers. On the other hand, it is of similar importance to detect the most common or prototypical samples in a data set. The latter is often useful to gain a better *understanding* of the data. This idea will be elaborated in this paper as well.

Summing up, this paper proposes simple indices (see Section 2) based on nearest neighbors that allow an ordering of the data from outliers to prototypes. Once this representation is established we can use it for (1) prototype detection (see Section 3.1), (2) outlier removal, or accordingly novelty detection (see Section 3.2), and (3) robustification of unsupervised algorithms (see Section 3.3). Experiments on toy and real data sets and handwritten digits underline the practicability of our algorithm, in particular for high-dimensional data sets.

2. Indices for ordering

Consider a set of n data points from the d-dimensional Euclidean space,

$$\{x_1, \ldots, x_n\} \subset \mathfrak{R}^d,$$

with the Euclidean norm, $\|x\| = \sqrt{x^\top x}$, and the Euclidean metric. Other metrics (e.g. other Riemannian metrics or Mahalanobis distance) can be effortlessly incorporated in our framework. For a data point $x \in \mathfrak{R}^d$, let

$$z_1(x), \ldots, z_k(x) \in \{x_1, \ldots, x_n\} \subset \mathfrak{R}^d,$$

be its k nearest neighbors among the given data points $x_1, \ldots, x_n$ (with respect to the chosen metric). In terms of these neighbors, we define three indices for each point $x \in \mathfrak{R}^d$. We will later use them for the ordering process. As usual, the choice of k influences the perception of the data: if k is chosen too small the focus is too local, if k is too large it is too global.

2.1. Kappa

The k-nearest neighbor density estimator assesses the density at a particular point by calculating the volume of the smallest ball centered at that point which contains its k nearest neighbors and relating it to the quotient k/n. It can be proven that this density estimator is L_2-consistent (see [22]). Unfortunately, the estimate is not always very accurate if the number of data points is small or the dimensionality is high. However, outlier detection does not require the actual density. In order to decide whether a data point is an outlier or not, an approximate estimate is a sufficient indicator. Our first index thus represents the essence of the k nearest neighbor density estimator: $\kappa(x)$ is the radius of the smallest ball centered at x containing its k nearest neighbors, i.e. the distance between x and its kth nearest neighbor,

$$\kappa(x) = \|x - z_k(x)\|.$$

Obviously, in dense regions κ is small and in sparse regions κ is large, making it a good candidate for an outlier index, as the rationale is that outliers lie in sparse regions.

2.2. Gamma

The index κ, however, seems to be somewhat wasteful: it considers the distance to the kth nearest neighbor, but it ignores the distances to the closer neighbors. This suggests a refined index that takes the distances to all k nearest neighbors into account: $\gamma(x)$ is x's average distance to its k nearest neighbors,

$$\gamma(x) = \frac{1}{k} \sum_{j=1}^{k} \|x - z_j(x)\|.$$

This index enables us to distinguish the two situations depicted on the left panel of Fig. 1: the value of κ is the

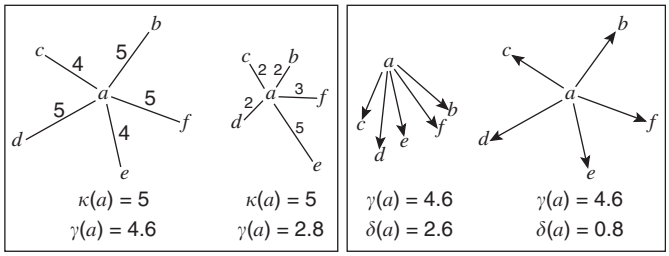

Fig. 1. The left panel shows that γ can distinguish better between sparse and dense regions than κ. The right panel shows that δ takes also the directions to the neighbors into account, whereas γ does not. Both panels assume $k = 5$.

same in both situations, because the 5th nearest neighbor of a has both times the same distance to a, although the neighborhood on the right is denser. By exploiting all distances, γ can distinguish both situations.

2.3. Delta

Looking at the right panel of Fig. 1, we see two situations that are indistinguishable for γ (and for κ as well), because the distances from a to its neighbors are the same in both settings. The directions of a's neighbors reveal the difference: on the left, a's neighbors point approximately into the same direction. On the right, a's neighbors are spread out in all directions. This information is captured by the following definition. $\delta(x)$ is the length of the mean of the vectors pointing from x to its k nearest neighbors,

$$\delta(x) = \left\| \frac{1}{k} \sum_{j=1}^{k} (x - z_j(x)) \right\|.$$

δ is large if the neighbors are located in the same direction. This happens especially for outliers since they have often all (or most) of their neighbors in one closest cluster. δ enables us to distinguish points in regularly filled sparse regions (all neighbors in different directions) from points which are outside clusters (all neighbors in the same direction).

2.4. Comparison

We note that δ is bounded by γ which itself is bounded by κ,

$$\kappa(x) \geqslant \gamma(x) \geqslant \delta(x) \geqslant 0,$$

which is easily seen by observing that $2\max(\|a\|, \|b\|) \geqslant \|a\| + \|b\| \geqslant \|a + b\| \geqslant 0$ holds. This means that if $\delta(x)$ is large (implying that x is probably an outlier) also $\gamma(x)$ is large. On the contrary, if $\delta(x)$ is small, then $\gamma(x)$ does not need to be small, since it might have ignored some relevant information. Therefore, in contrast to δ, γ might misjudge a point from a sparser region to be an outlier. The analog discussion holds for $\gamma(x)$ and $\kappa(x)$.

2.5. Computational complexity

All three indices calculate the distance matrix, which requires $O(n^2 d)$ operations for n data points of dimension d. Finding the kth nearest neighbor of one point can be done in linear time (selection in expected linear time, see [9]). For κ we have to do this for each point, i.e. $O(n^2)$. So, the total time complexity for κ is $O(n^2 d + n^2)$. γ and δ require all k nearest neighbors, which can be found (using k times selection in expected linear time) in $O(kn)$ for each point, i.e. in total $O(n^2 d + n^2 k)$ for all points. For large k, we can also sort all neighbors of a point (in $O(n \log n)$), i.e. in total $O(n^2 d + n^2 \log n)$ for all points. Summarizing, κ requires $O(n^2 d + n^2)$ and γ and δ need $O(n^2 d + n^2 \max(k, \log n))$.

The previous considerations discuss the costs for calculating κ, γ or δ for a fixed k. In some applications (e.g. in Section 3.2) we can obtain the optimal k by cross-validation. What are the computational costs for this parameter-selection process? Usually, an r-fold cross-validation requires the complete method to be repeated r times for different splits. Fortunately, the distance matrix between all data points has to be calculated only once. After splitting the data set into training and validation set, the required distances can be read out instantly of the large distance matrix. Then for a given split having the columns of the distance matrix sorted once, all different ks can be tested with low additional costs (which is a common trick for k-nearest neighbor algorithms).

Note, that because the running time of κ, γ and δ is quadratic in the number of data points, large data sets might be difficult to process. A simple trick is to calculate the distance matrix only for blocks of a certain size along the diagonal and to calculate the indices per block. However, this would waste some of the information contained in the data. Instead we suggest to employ sophisticated data structures such as *kd*-trees, metric-trees, ball-trees (see [15,37,27]) or approximate methods to find the nearest neighbors (see [21] and references therein) which can reduce the running times significantly.

3. Applications

The definitions of κ, γ and δ have been motivated mainly by outlier detection. In Section 3.2 we show that these indicators are indeed powerful tools to detect outliers, especially if the data is high-dimensional. After that, we introduce a new approach based on these indices to robustify unsupervised algorithms such as clustering (Section 3.3.1) and nonlinear dimensionality reduction (Section 3.3.2). The idea is to discard some large proportion of the data in order to reveal the underlying structure. But first, to get an intuitive feeling for the indices, we apply them to a toy data set.

3.1. Finding prototypical data points

To understand some given set of data points, often, the most prototypical points are instructive, which we describe in this section. For data that lies on a nonlinear manifold, these points are generally quite different from the mean which in those cases does not always have a reasonable interpretation.

To illustrate the basic idea, we sort 200 points that are sampled from a circular distribution. The four panels in Fig. 2 show the results. The points are plotted as circles with varying size according to the value of the respective indices: distance to the mean and κ, γ, δ (for $k = 5$, other values for k lead to similar results). The distance to the mean ignores the underlying structure: the size of the circles depends only how close a point is to the center, which is the mean. In contrast, the proposed indices κ, γ and δ order the points from the dense regions to the sparse regions, i.e. the circles in the concentrated areas along the ring are small, in the other areas large. Therefore, the points with the smallest index are representative for the underlying structure. These prototypical points will be used in Section 3.3 to enhance algorithms that search for structure.

To explain the behaviors of the indices for two clusters, we created a data set with 100 points sampled from two Gaussian distributions with different mean and variance. The left-most panel of Fig. 3 shows that sorting the points according to the distance to the mean ignores the structure in each cluster and the fact that there are two clusters. The indices κ, γ, δ sort the data points from the inside of the clusters to the outside. However, note that points from a denser cluster are considered more prototypical than the points from the inside of a sparse cluster. Thus the most prototypical points according to κ, γ and δ of a data set might belong all to the same cluster. This fact is no problem for the examples in this paper (see e.g. Section 3.3). However, some applications such as robust independent component analysis require that the prototypes belong to different clusters. This can be achieved by a simple heuristic (see [17,24,25] for details).

Note that random sampling also provides representative points. However, in contrast to κ, γ and δ such points lie only with high probability in the densest regions. Furthermore, such a sampling does not deliver a sorting on all data points.

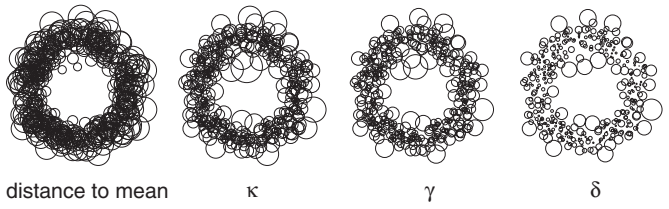

Fig. 2. 400 data points sampled from a circular distribution shown as little circles. The size of the circle indicates the values of one of four indices: distance to the mean (from left to right) and κ, γ, δ (with $k = 5$). The distances to the mean used in the left-most panel ignore the underlying structure, while the other three indices reflect this structure.

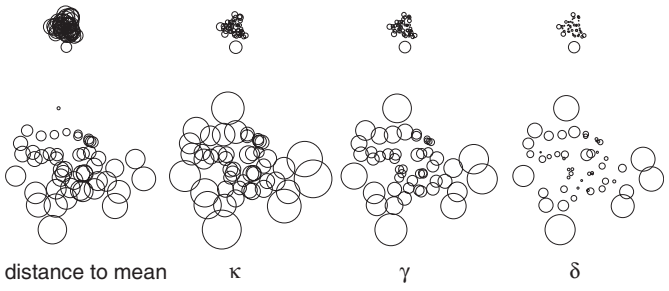

Fig. 3. 100 data points sampled from two Gaussians with different variances. The size of each circle indicates the values of one of four indices: distance to the mean and κ, γ, δ (with $k = 5$). The circle-sizes in the left-most panel ignore the structure in each cluster and the fact that there are two clusters. The other indices grow from the inside to the outside for each cluster. Since the upper cluster is more concentrated, its circles are smaller.

Table 1
The dimensionalities and sample sizes (targets) of the considered data sets

	Data set	Dims	Targets
1	Iris-virginicia	4	50
2	Breast cancer Wisconsin	9	458
3	Sonar (90% PCA)	10	111
4	Hepatitis database	18	105
5	Imports85 data set	25	71
6	Delft pump data (separate)	32	913
7	Sonar (no PCA)	60	111
8	Delft pump data (appended)	160	189
9	NIST, class 0	256	200
10	Concordia, class 0	1024	400

3.2. Detecting outliers on real data sets

In this section we compare κ, γ and δ with six other standard outlier detection methods. First, we will introduce the data sets, then we discuss the other methods and the training and evaluation procedure. Finally, we report the results.

3.2.1. Data sets

In these experiments we consider 10 real world data sets, most of them from the UCI-repository [4]. The data sets differ in dimensionality (ranging from 4 up to 1024) and sample size (from 50 to more than 900), see Table 1.

The Sonar data set appears twice: (1) as the original data (60-dimensional) and (2) with reduced dimensionality (using PCA keeping 90% of the variance).

The two pump data sets (Nos. 6 and 8) contain vibration measurements of a water pump [40,39]. The pump can be operating in normal working conditions or in one of three faulty operation modes (loose foundation, bearing failure or imbalance). Five instantaneous vibration measurements were performed on the pump. On these

time-series an auto-regressive model was fitted and the obtained coefficients were treated as feature vectors. In the first data set, these five measurements were treated independently and were added to the data set as independent samples. In the second data set these five measurements were concatenated to one large feature vector.

The NIST and Concordia data sets are high-dimensional data sets containing images of handwritten digits. The NIST data set contains 16×16 images, The data set used in the experiments was taken from the Special Database 3 distributed on CD-ROM by the U.S. NIST, the National Institute for Standards and Technology. Currently, this database is discontinued; it is now distributed together with Database 7 as Database 19 on the NIST website.[1] The preprocessing used is described in [11]. The other handwritten digits database, the Concordia data set [8], contains 32×32 images.

3.2.2. Outlier detection methods

We compare the three indices κ, γ and δ with six other outlier detection methods:

Gauss density estimator (*Gauss*): For this simple unimodal model, the mean and covariance matrix has to be estimated. When the inversion of the covariance matrix becomes problematic, the covariance matrix is regularized by adding a constant to the diagonal. This constant is the hyper-parameter of the method. A large distance to the mean distinguishes outliers from the rest of the data.

Minimum-covariance-determinant Gaussian (*MCD*): This is almost the same model as the Gaussian density method above; however, for the estimation of the covariance matrix 75% of the data is used (see [29]). This fraction of the data is selected such that it gives the smallest determinant for the covariance matrix. No hyper-parameter is optimized in this method. We used a Matlab implementation, called FAST-MCD.[2]

Parzen density estimator (*Parzen*): This is a standard kernel-based density estimator, e.g. for a Gaussian kernel it is a mixture of Gaussian distributions that are centered on each single training point. The hyper-parameter of this method is the width of the kernel. The width is optimized by maximizing the likelihood in a leave-one-out fashion on the training set (like in [13,16]).

Mixture of Gaussians (*MoG*): k Gaussian clusters with full covariance matrices are estimated using the EM algorithm [36]. The number of clusters k is the hyper-parameter.

Support vector data description (*SVDD*): SVDD fits a hyper-sphere around the target class (see [34]; alternatively, the hyper-sphere separates the origin from the data with maximum margin as described in [32]). These one-class classifiers can be made more flexible by introducing support-vector-kernel functions. In these experiments, a Gauss-kernel is chosen (also called radial-basis-function-kernel, or short RBF-kernel). The width of this kernel is the hyper-parameter.

k-means clustering (*k-means*): A k-means clustering is applied such that the means characterize the target distribution. For each object, the distance to the nearest mean measures the outlierness (see [33]). k is the hyper-parameter.

3.2.3. Error estimation

Most of the data sets are multi-class data sets. In order to test outlier detection methods on this data, we consider the points of one of the given classes as the targets (i.e. the non-outliers) and consider the points from the other classes as outliers. The target data is split into training and testing data in order to estimate the performance by a 5-fold cross-validation. The one-class methods are fitted on the training data (which contains only targets). Then the objects from the left-out data combined with the outlier data are one-by-one evaluated.

On the test data the receiver-operating-characteristic curve, abbr. ROC-curve (see [26]), is computed: for all possible threshold values, the error on the target data ε_t and on the outlier data ε_o is calculated. The resulting curves are compared by estimating the area under the ROC curves. 'Area under curve' is abbreviated as AUC (see [5]; see also Fig. 4). This is defined as

$$\mathrm{AUC} = 1 - \int_0^1 \varepsilon_t(\varepsilon_o)\,\mathrm{d}\varepsilon_o. \tag{1}$$

The larger this area, the better the distinction between the target and outlier data. Perfect separation is obtained for $\mathrm{AUC} = 1.0$. A classifier which randomly assigns the labels 'target' and 'outlier' will obtain $\mathrm{AUC} = 0.5$.

Most of the methods contain a hyper-parameter. The Parzen density and the SVDD contain the hyper-parameter σ (the width of the kernels), while the mixture of Gaussians, k-means and the κ, γ and δ-indices contain the

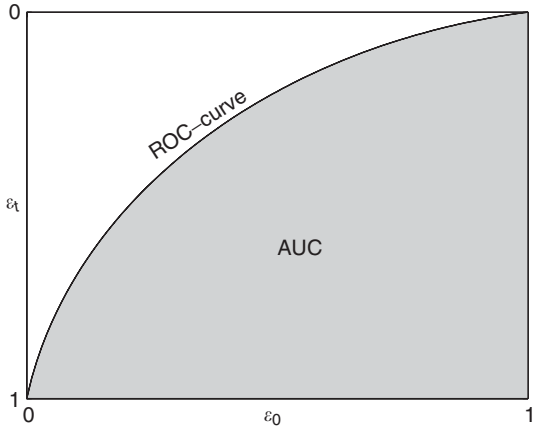

Fig. 4. The ROC-curve shows the error on the target data, ε_t, against the error on the outlier data, ε_0. The area under the curve (abbr. AUC) measures the quality of the separation. $\mathrm{AUC} = 1$ means perfect separation, $\mathrm{AUC} = 0.5$ means random separation.

[1] See http://www.nist.gov/srd/nistsd19.htm

[2] Available at http://www.agoras.ua.ac.be/Robustn.htm

Table 2
AUC performances (×100, with standard deviations in the brackets) of the considered outlier detection methods for 10 real world data sets

	1	2	3	4	5
Gauss	97.4 (±1.9)	98.9 (±0.8)	63.3 (±6.8)	80.2 (±13.9)	75.9 (±4.7)
MCD-Gauss	97.4 (±2.2)	99.2 (±0.5)	63.4 (±7.0)	81.9 (±14.8)	75.1 (±6.5)
Parzen	95.5 (±2.5)	99.3 (±0.4)	83.3 (±2.7)	57.7 (±14.8)	85.4 (±4.8)
MoG	96.5 (±2.5)	99.2 (±0.3)	73.7 (±4.4)	81.0 (±10.1)	88.6 (±2.1)
SVDD	97.7 (±1.6)	98.8 (±1.0)	86.1 (±4.8)	49.9 (±5.0)	83.4 (±5.8)
k-means	96.7 (±2.3)	99.5 (±0.3)	81.2 (±2.8)	43.9 (±12.6)	79.1 (±5.9)
κ	95.3 (±2.1)	99.6 (±0.2)	80.4 (±3.4)	55.0 (±8.2)	81.0 (±3.3)
γ	96.0 (±2.6)	99.7 (±0.2)	81.3 (±5.3)	61.8 (±7.8)	84.3 (±6.0)
δ	96.4 (±2.5)	99.7 (±0.1)	79.9 (±2.1)	57.2 (±5.4)	85.0 (±3.8)
	6	7	8	9	10
Gauss	80.1 (±0.0)	74.5 (±2.4)	99.8 (±0.0)	95.3 (±2.9)	96.9 (±0.0)
MCD-Gauss	75.1 (±0.0)	—	—	—	—
Parzen	95.2 (±0.3)	83.3 (±3.6)	99.2 (±0.2)	—	—
MoG	71.6 (±0.0)	78.8 (±2.6)	99.2 (±0.1)	—	—
SVDD	90.7 (±0.0)	59.6 (±11.7)	99.8 (±0.1)	98.1 (±0.5)	96.5 (±0.3)
k-means	92.3 (±0.1)	82.0 (±5.0)	99.4 (±0.3)	98.5 (±0.8)	96.8 (±0.4)
κ	95.5 (±0.0)	83.3 (±4.1)	99.3 (±0.1)	98.8 (±0.7)	96.3 (±0.9)
γ	97.3 (±0.0)	87.0 (±4.9)	99.3 (±0.1)	98.6 (±0.7)	96.3 (±1.7)
δ	96.4 (±0.0)	84.7 (±4.4)	99.2 (±0.1)	98.5 (±0.9)	96.1 (±0.9)

hyper-parameter k. When an outlier detection method contains a hyper-parameter, this hyper-parameter will be selected by applying nested 5-fold cross-validation on the training set. This means that one 5-fold cross-validation (to estimate the hyper-parameters) is performed within another 5-fold cross-validation (to estimate the AUC).

Note that for κ, γ and δ, the model selection, i.e. the optimization over the hyper-parameter k, is computationally inexpensive. The distance matrix of the complete training set has to be computed only once and sorted along the columns (this computation is the most expensive part). After that, the AUC for different k can be easily calculated. For other methods, the model selection is computationally much more expensive. In particular, the optimization of the SVDD and MoG is very time consuming, since the former involves solving quadratic programming problems for a large number of free parameters, and the latter requires the convergence of the EM-algorithm.

3.2.4. Results

Table 2 shows the AUC performances for all outlier detection methods on all data sets (one column per data set). We see that even for low dimensions (data sets 1, 2 and 3), the results of κ, γ and δ are very competitive. Notice that all classifiers perform almost equally and that the performance differences are not significant for the first two data sets.

In the data sets with reasonably large dimensionalities (around 30 dimensions, data sets 4 and 5), the target and the outlier class heavily overlap. Because the target class appears to be approximately Gaussian, the Gaussian density estimation works acceptably. Particularly, data set 4 is under-sampled, and the variance in the performance is high. In the five cross-validation runs for finding a good k for κ, the best ks were 3, 8, 12, 25 and 47, which indicates that the learned models of the different cross-validations are highly unstable. The simpler Gaussian, MCD Gaussian and mixture of Gaussians (the latter simply mimicking the single Gaussian) are therefore better on this data set. Nevertheless, an instability in the estimates of the hyper-parameter can be used as an indicator to discard a method for the given data set. However, noting that the methods based on covariance matrices perform well, it is worth trying to run all other methods again with the corresponding Mahalanobis distance, i.e. compensating different scaling of certain directions.[3] Table 3 shows that this intuition is right and all methods have comparable performance on data set 4 using the Mahalanobis distance.

For higher dimensions (larger than 60, especially data sets 7, 8, 9 and 10) our methods perform very well. This is in contrast to the density estimators, which can completely fail. It is very hard to estimate a density reliably in these high-dimensional spaces. The Parzen density and the mixture of Gaussian methods gave a zero density estimate for almost all objects, hereby classifying all objects as outliers. No ROC-curve and AUC performance could be computed for these methods (for that reason the table entry contains a long dash —). For the one-class classifier that can deal with a large number of dimensions, an almost optimal performance is obtained in those cases. The single Gaussian density gave acceptable results after a careful regularization of the covariance matrix. This might indicate that these data sets contain not much nonlinear structure, thus being not particularly hard. Furthermore, the fact that those high-dimensional data sets are well described by a

[3]Thanks to the reviewer for giving this suggestion.

regularized Gaussian fit might imply that those data sets are approximately Gaussian distributed. In high dimensions this means that most points are orthogonal to and have similar distances between each other. However, despite of this lack of structure, which κ, γ and δ try to exploit (see Section 2 and Fig. 1), those indices perform also very well in that situation.

Table 4 shows the training and testing times of the classifiers for the first 5 data sets (all algorithms run on a computer with an AMD Athlon processor with 1533 MHz). Each method is trained and tested 10 times on the data sets, without using model selection. The reported times are the average of those 10 runs.

- The training and testing times of the Gaussian are short (at the expense of being an inflexible model).
- The training times for the MCD-Gaussian, SVDD and mixture of Gaussians can be very long if the optimization is stuck on some plateau. Evaluations are fast.

Table 3
Changing the metric on data set 4 to Mahalanobis distance improves the performance for most methods significantly

	4	4 (Mahalanobis)
Gauss	80.2 (±13.9)	81.7 (±11.6)
MCDGauss	81.9 (±14.8)	73.5 (±15.2)
Parzen	57.7 (±14.8)	77.6 (±12.1)
MoG	81.0 (±10.1)	72.0 (±10.4)
SVDD	49.9 (±5.0)	78.2 (±13.8)
kmeans	43.9 (±12.6)	77.1 (±12.8)
κ	55.0 (±8.2)	77.3 (±12.7)
γ	61.8 (±7.8)	78.9 (±11.6)
δ	57.2 (±5.4)	79.4 (±9.8)

- The training and testing times of the κ, γ and δ indices are mainly determined by the training set size (see data set 2 and 6), as was to be expected from the discussion in Section 2.5. Note that there are data structures which can speed up methods based on nearest neighbors such as κ, γ and δ (for a discussion on this see Section 2.5).

3.3. Revealing hidden structure in noisy data

Often, the statistical structure of a data set is expressed by the points in the denser regions. Common approaches typically make strong assumptions on the data (e.g. parametric statistics presupposes certain distributions). The indices κ, γ and δ open up a new approach: by ignoring the data in sparser regions (e.g. points with large γ) and focusing on the denser regions (e.g. points with small γ), the true underlying structure can be found using simple algorithms. This concept omits not only the classical outliers, which might arise from distributions other than the true data distribution, but also data points from the true distribution which are located in sparse regions.

In the following, we illustrate this idea for clustering (Section 3.3.1) and for noisy nonlinear dimensionality reduction (Section 3.3.2).

3.3.1. Clustering

We robustify a clustering algorithm, that calculates the minimum spanning tree (abbr. MST, see [9]) of the given data points and then cuts the longest edge to obtain two clusters. Obviously, this simple method fails if the data is very noisy or if there are outliers in the data, because those are very likely the ones that will be cut off. Simply by ignoring the untypical data points (in other words: by

Table 4
Average training and testing times (in ms) of the considered outlier detection methods for 5 real world data sets (averaged over 10 repetitions)

	1	2	3	4	5
Training times (in ms)					
Gauss	13 (±2)	15 (±0)	13 (±0)	14 (±0)	14 (±0)
MCD-Gauss	3074 (±31)	24 (±1)	6284 (±28)	23 (±1)	23 (±1)
Parzen	26 (±1)	144 (±2)	30 (±1)	30 (±1)	27 (±1)
MoG	189 (±5)	293 (±3)	209 (±3)	237 (±2)	245 (±2)
SVDD	373 (±3)	14100 (±80)	349 (±10)	68 (±1)	115 (±2)
k-means	19 (±0)	77 (±15)	33 (±4)	27 (±6)	28 (±6)
κ	10 (±0)	119 (±2)	15 (±1)	16 (±4)	12 (±0)
γ	11 (±0)	118 (±1)	15 (±1)	14 (±0)	12 (±0)
δ	10 (±0)	120 (±2)	17 (±5)	14 (±0)	12 (±0)
Testing times (in ms)					
Gauss	6 (±1)	8 (±1)	7 (±1)	7 (±0)	8 (±0)
MCD-Gauss	6 (±1)	7 (±1)	7 (±0)	7 (±0)	8 (±0)
Parzen	22 (±1)	200 (±3)	30 (±1)	26 (±1)	26 (±1)
MoG	12 (±0)	17 (±3)	12 (±0)	13 (±1)	14 (±0)
SVDD	7 (±0)	47 (±1)	15 (±4)	12 (±1)	10 (±0)
k-means	7 (±1)	11 (±1)	7 (±0)	7 (±0)	7 (±1)
κ	8 (±0)	174 (±2)	17 (±1)	12 (±0)	11 (±0)
γ	9 (±1)	177 (±5)	17 (±0)	12 (±1)	11 (±1)
δ	9 (±1)	176 (±1)	20 (±5)	13 (±1)	12 (±1)

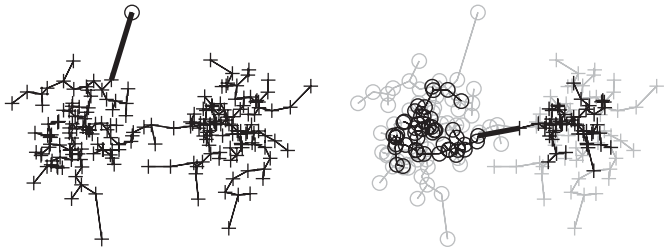

Fig. 5. Clustering results for MST-based clustering using all data (left panel) and the 50% proportion with the smallest γ (right panel, the omitted points are gray). The underlying graphs are the MSTs. The thick edges show the cuts. '+' and 'o' indicate to which cluster a given point is assigned. Right panel: after determining the cut (and therewith the two clusters) on the selected points (shown in black), the omitted points are assigned in a greedy manner. The underlying graph is the MST.

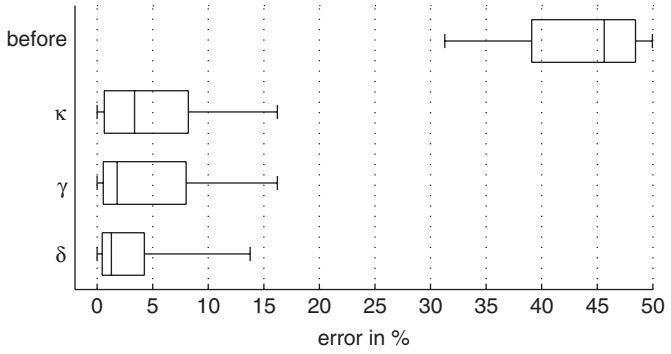

Fig. 6. Box plots (showing minimum, 25%, median, 75%, maximum) of the classification errors for the simple clustering method explained in the text for all 45 two-digit subsets of the USPS handwritten digits data set. Thinning out the data sets using κ, γ and δ reduces the original errors (shown in the first row) considerably.

focusing on the prototypes) this plain and fast clustering algorithm becomes useful.

Fig. 5 illustrates the basic idea for a toy data set (consisting of 200 data points sampled from a bimodal Gaussian distribution): applying the MST-based clustering to *all* data points cuts off only a single outlier as one of the clusters (left panel), which is not the wanted result; by ignoring 50% of the data that have a large value of γ (calculated for $k = 5$), the MST-based clustering algorithms finds a meaningful cut. The omitted points are added in a greedy manner: add the not-yet-assigned point to the closest clusters (consisting of the so-far-assigned points) until all points are assigned. The right panel shows that the correct clustering has been found ('+' and 'o' represent the assigned labels). Note that the choice of the proportion of points to keep and the choice of k is uncritical.

A more difficult test bed for clustering are the 45 subsets, each containing exactly two classes of the USPS handwritten digits (in total about 7000 data points). Because for this data the correct labels are known, we can evaluate the clustering results. To analyze the MST-based clustering that uses *all* data (as explained above), we apply it to all 45 subsets: as was conceivable, we observe that in all but two cases of the 45 experiments, the longest edge in the MST connects only one point (probably an outlier) to the rest of the points (in the other two cases it connects two points). Cutting this edge leads to one big cluster that contains all but one point (in two cases: all but two points). Assuming that the larger cluster belongs to the larger of the two classes (each subset collects data points from exactly two digits), the errors (summarized in the upper box plot in Fig. 6) reflect only the prior probabilities of the particular pairs of digits. We see that the simple algorithm based on MSTs fails if we use *all* points.

Following our robustification concept, we choose some fraction ν of the points from the denser regions; for this we employ the indices κ, γ and δ for a certain k. Then we calculate the MST on the reduced set of points and obtain two clusters by cutting its longest edge. The omitted points are added to the found clusters as explained above for the toy example.

The hyper-parameters k and $\nu \in \{0.02, 0.04, 0.06, \ldots, 1\}$ are chosen by a simple heuristic: choose k and ν such that the size of the smaller cluster is maximized. This heuristic favors solutions in which both clusters are estimated in a reliable and robust way (since they contain many points). The range of ν for which we get good results is quite large, so it is not crucial to choose it precisely. Again, the model selection procedure is inexpensive (see Section 2.5).

In order to calculate errors, we assign the known labels to the clusters. There are two possibilities, each leading to a possibly different error. The three lower box plots in Fig. 6 are based on the smaller of the errors for κ, γ and δ. In all cases, the performance has been improved considerably compared to $\nu = 1$ (i.e. keeping all points, shown in the upper box plot in Fig. 6), especially for index δ. The values of k and ν have been chosen using the heuristic explained in the previous paragraph. It turned out that often a very small fraction (as small as $\nu = 0.02$) is enough to capture the structure of the underlying clusters. Note that, having peeked at the errors for all possible values for k and ν, we observe that even smaller errors are possible for certain k and ν. This suggests that it is promising to try other model selection strategies for k and ν.

3.3.2. Noisy nonlinear dimensionality reduction

Recently, several methods have been proposed that find nonlinear embeddings of high-dimensional data (e.g. [30,35]). Unfortunately, these methods are sensitive to noise. A strategy to robustify these methods for practical applications (where noise is often present) is to remove untypical data points (that originate e.g. from noise) before applying the unsupervised algorithm. Fig. 7 shows an illustrative example: 1000 data points are drawn from a one-dimensional manifold (in the shape of a spiral) that is embedded in the two-dimensional space. The one-dimensional manifold defines a natural ordering of the points

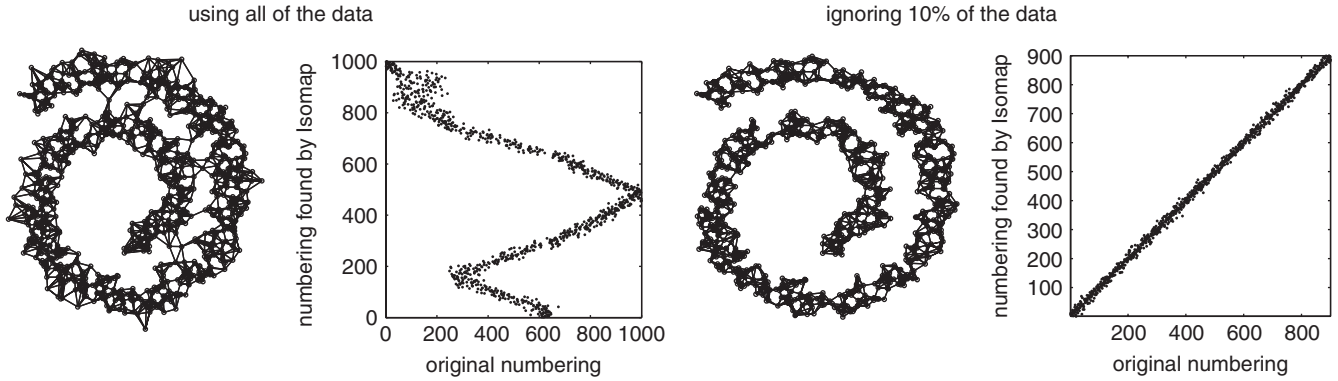

Fig. 7. The left panel shows the k nearest neighbor graph (exemplarily for $k = 6$) on the noisy spiral data set and its one-dimensional embedding found by Isomap (shown in the scatter plot against the initial ordering). The right panel shows that after eliminating the shortcuts (by removing the points with large γ) Isomap's one-dimensional embedding matches the initial ordering.

(from the inside to the outside of the spiral). Recovering the natural ordering of the given data set is a nonlinear dimensionality reduction problem that can be solved by the Isomap algorithm [35] or by locally linear embedding, abbr. LLE [30]. Here, we use Isomap, but the same considerations apply to LLE as well. Isomap employs multidimensional scaling, abbr. MDS (see [10]), to a distance matrix which measures geodesics along the k nearest neighbor graph. The left panel in Fig. 7 shows a typical problem that arises when the data is noisy: the k nearest neighbor graph contains some shortcuts between different windings of the spiral (i.e. edges that connect the inner part of the spiral in a radial direction to the outer part). The panel shows exemplarily the k nearest neighbor graph for $k = 6$. Also for all other k the graph either contains shortcuts or it is not connected (which is another requirement for Isomap). Consequently, the calculated distance matrix does not measure along the geodesics and Isomap fails to find the correct ordering of the data points (as can be seen on the scatter plot in the left panel of Fig. 7). This can be also seen from the MDS eigenvalues: the number of large MDS eigenvalues corresponds to the embedding dimensionality. For the data in the left panel, for all choices of k the resulting dimensionality is two, because the shortcuts lead to an inherently two-dimensional structure.

In order to avoid shortcuts, we propose to remove a certain amount of untypical points: after ignoring 10% of the untypical points (i.e. keeping $\nu = 90\%$ of the points that have a large γ), all the shortcuts are eliminated (as can be seen in the right panel). This allows Isomap to estimate a meaningful distance matrix along the geodesics and therewith to recover the original order of the remaining 900 data points (see the scatter-plot in the right panel). The choice of the hyper-parameters, i.e. finding the right percentage of points ν to keep and a reasonable k for Isomap and γ is uncritical in our example. These parameters are adjusted by looking at the MDS eigenvalues (see [10]), again at low computational costs (see Section 2.5). In our case they have been selected such that there is only one large MDS eigenvalue. Note, that this is only possible after removing untypical points. Thus we see that our simple and fast preprocessing step can make algorithms work that would otherwise be inapplicable.

4. Conclusion

The paper proposes three simple indices that can efficiently order high-dimensional samples from typical to untypical. Various applications, e.g. outlier and prototype detection, robustifying nonlinear dimensionality reduction and simple clustering algorithms, underline the usefulness of our approach. In particular, in outlier detection we find excellent results that compare favorably with more sophisticated algorithms. Note that also the speed of our method is very satisfying when compared with other methods.

Already κ, the distance to the kth nearest neighbor, is a useful tool for outlier detection. However, often γ, the average distance to the k nearest neighbors, which can be seen as a refinement of κ, is preferable, e.g. as seen for some of the data sets in Section 3.2. The third index δ, the length of the average direction to the k nearest neighbors, performs often similar to γ. However, for the clustering example in Section 3.3.1 it clearly shows the best performance. The reason is that δ is focussing on a slightly different property of the close neighborhood than κ and γ as discussed in Section 2.4.

The simplicity of κ, γ and δ makes them versatile tools for exploratory data analysis, especially for high-dimensional data sets. We are now free to use the ordering information as we wish, e.g. for database cleaning, for retraining in order to improve generalization properties of classifiers or simply to gain insights into the usual and unusual properties of the data at hand.

Acknowledgements

The authors are grateful for valuable discussion with Paul von Bünau, Andreas Ziehe, Christin Schäfer, Benjamin Blankertz, Sebastian Mika, Steven Lemm, Pavel Laskov and Motoaki Kawanabe. This research was partly supported by the EU-PASCAL network of excellence (IST-2002-506778) and the Deutsche Forschungsgemeinschaft (DFG SFB 618-B4 and MU 987/1-1) and through a European Community Marie Curie Fellowship. The authors are solely responsible for information communicated and the European Commission is not responsible for any views or results expressed. Finally, the authors would like to thank the anonymous reviewers for their valuable suggestions.

References

[1] A. Baillo, A. Cuevas, A. Justel, Set estimation and nonparametric detection, Can. J. Stat. 28 (3) (2000).
[2] V. Barnett, T. Lewis, Outliers in statistical data, Wiley Series in Probability and Mathematical Statistics, second ed., Wiley, New York, 1978.
[3] C.M. Bishop, Novelty detection and neural network validation, IEE Proc. Vision, Image Signal Process, Appl. Neural Networks 141 (4) (1994) 217–222 (special issue).
[4] C.L. Blake, C.J. Merz, UCI repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html, a huge collection of artificial and real-world data sets, 1998.
[5] 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.
[6] M.M. Breunig, H.-P. Kriegel, R.T. Ng, J. Sander, Lof: identifying density-based local outliers, in: ACM SIGMOD International Conference on Management of Data, Dallas, Texas, 2000.
[7] C. Campbell, K.P. Bennett, A linear programming approach to novelty detection, in: T.K. Leen, T.G. Dietterich, V. Tresp (Eds.), Advances in Neural Information Processing Systems, vol. 13, MIT Press, Cambridge, MA, 2001, pp. 395–401.
[8] S.-B. Cho, Neural-network classifiers for recognizing totally unconstrained handwritten numerals, IEEE Trans. Neural Networks 8 (1) (1997).
[9] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, MIT Press, Cambridge, MA, 1989.
[10] T.F. Cox, M.A.A. Cox, Multidimensional Scaling, Chapman & Hall, London, 2001.
[11] D. de Ridder, Shared weights neural networks in image analysis, Master's Thesis, Technische Universiteit Delft, 1996.
[12] L. Devroye, G.L. Wise, Detection of abnormal behavior via nonparametric estimation of the support, SIAM J. Appl. Math. 38 (1980) 480–488.
[13] 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.
[14] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, S. Stolfo, A geometric framework for unsupervised anomaly detection: detecting intrusions in unlabeled data, Applications of Data Mining in Computer Security, Kluwer, Docdrecht, 2002.
[15] J.H. Friedman, J.L. Bentley, R.A. Finkel, An algorithm for finding best matches in logarithmic expected time, ACM Trans. Math. Software 3 (3) (1977) 209–226.
[16] W. Härdle, Applied Nonparametric Regression, Cambridge University Press, Cambridge, 1990.
[17] S. Harmeling, Independent component analysis and beyond, Ph.D. Thesis, Universität Potsdam, Germany, 2004.
[18] P.J. Huber, Robust Statistics, Wiley, New York, 1981.
[19] 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.
[20] E.M. Knorr, R.T. Ng, V. Tucakov, Distance-based outliers: algorithms and applications, Int. J. Very Large Data Bases 8 (3–4) (2000) 237–253.
[21] T. Liu, A. Moore, A. Gray, K. Yang, An investigation of practical approximate nearest neighbor algorithms, in: L.K. Saul, Y. Weiss, L. Bottou (Eds.), Advances in Neural Information Processing Systems, vol. 17, MIT Press, Cambridge, MA, 2005.
[22] D. Loftsgaarden, C. Quesenberry, A nonparametric estimate of a multivariate density function, Ann. Math. Stat. 36 (1965) 1049–1051.
[23] S. Marsland, On-line novelty detection through self-organisation, with application to inspection robots. Ph.D. Thesis, University of Manchester, 2001.
[24] F. Meinecke, S. Harmeling, K.-R. Müller, Robust ICA for super-Gaussian sources, in: Proceedings of the International Workshop on Independent Component Analysis and Blind Signal Separation (ICA2004), 2004.
[25] F. Meinecke, S. Harmeling, K.-R. Müller, Inlier-based ICA with an application to super-imposed images, IJIST Special Issue on BSS and Blind Deconvolution 2005, to be published.
[26] C.E. Metz, Basic principles of ROC analysis, Semin. Nucl. Med. VIII (4) (1978).
[27] S.M. Omohundro, Efficient algorithms with neural network behaviour, J. Complex Systems 1 (2) (1987) 273–347.
[28] S. Ramaswamy, R. Rastogi, K. Shim, Efficient algorithms for mining outliers from large data sets, in: ACM SIGMOD International Conference on Management of Data, Dallas, Texas, 2000, pp. 427–438.
[29] P.J. Rousseeuw, K. Van Driessen, A fast algorithm for the minimum covariance determinant estimator, Technometrics 41 (3) (1999) 212–223.
[30] S. Roweis, L. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326.
[31] C.M. Santos-Pereira, A.M. Pires, Detection of outliers in multivariate data, a method based on clustering and robust estimators, in: Proceedings in Computational Statistics, Physica-Verlag, 2002, pp. 291–296.
[32] B. Schölkopf, J. Platt, J. Shawe-Taylor, A.J. Smola, R.C. Williamson, Estimating the support of a high-dimensional distribution, Neural Computation 13 (7) (2001) 1443–1471.
[33] D.M.J. Tax, One-class classification. Ph.D. Thesis, Delft University of Technology, http://www.ph.tn.tudelft.nl/~davidt/thesis.pdf, June 2001.
[34] D.M.J. Tax, R.P.W. Duin, Uniform object generation for optimizing one-class classifiers, J. Mach. Learn. Res. (2001) 155–173.
[35] J.B. Tenenbaum, V. de Silva, J.C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (5500) (2000) 2319–2323.
[36] D. Titterington, A. Smith, U. Makov, Statistical Analysis of Finite Mixture Distributions, Wiley, New York, 1995.
[37] J.K. Uhlmann, Satisfying general proximity/similarity queries with metric trees, Inf. Process. Lett. 40 (1991) 175–179.
[38] J. Weston, O. Chapelle, I. Guyon, Data cleaning algorithms with applications to micro-array experiments, Technical Report, BIOwulf Technologies, 2001.
[39] A. Ypma, P. Pajunen, 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, January 1999, pp. 37–42.
[40] A. Ypma, D.M.J. Tax, R.P.W. Duin, Robust machine fault detection with independent component analysis and support vector data description, in: Y.-H. Hu, J. Larsen, E. Wilson, S. Douglas (Eds.), Neural Networks for Signal Processing, vol. IX, August 1999, pp. 67–76.

Stefan Harmeling studied mathematics with a specialization in mathematical logic at Universität Münster (Germany) and received in 1998 the "Diplommathematiker" degree (cmp. to Master in mathematics). After that he pursued a Master program in computer science with an emphasis on artificial intelligence at Stanford University (USA) which he completed in 2000 (supported by the DAAD). Since then he is a member of the IDA group at Fraunhofer FIRST (former GMD FIRST) in Berlin. In 2004, he obtained the "Dr. rer. nat" (cmp. to Ph.D) in computer science at Universität Potsdam (Germany) with the thesis "Independent Component Analysis and beyond". Dr. Harmeling is interested in all aspects of machine learning and signal processing. In particular, he has worked on nonlinear blind source separation, outlier detection, model selection and kernel methods.

Guido Dornhege was born in Werne, Germany, in 1976. He received the Diploma degree in mathematics 2002 from University of Münster, Germany. He conducted studies of Maass cusp forms. Since 2002 he is member of the intelligent data analysis (IDA) group at Fraunhofer-FIRST in Berlin working in the Berlin Brain-Computer Interface (BBCI) project. His scientific interests are in the field of analysis of biomedical data by machine learning techniques.

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 Ph.D. 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.

Frank C. Meinecke received the Diploma degree in physics from the University of Potsdam, Germany, in 2003. He is currently working towards the Ph.D. degree in the Intelligent Data Analysis Group at Fraunhofer FIRST in Berlin, Germany.

His research interest is focused on nonlinear dynamics and signal processing, especially blind source separation and independent component analysis.

Klaus-Robert Müller received the Diploma degree in mathematical physics 1989 and the Ph.D. in theoretical computer science in 1992, both from University of Karlsruhe, Germany. From 1992 to 1994 he worked as a Postdoctoral fellow at GMD FIRST, in Berlin where he started to build up the intelligent data analysis (IDA) group. From 1994 to 1995 he was a European Community STP Research Fellow at University of Tokyo in Prof. Amari's Lab. From 1995 on he is department head of the IDA group at GMD FIRST (since 2001 Fraunhofer FIRST) in Berlin and since 1999 he holds a joint associate Professor position of GMD and University of Potsdam. In 2003 he became full professor at University of Potsdam. He has been lecturing at Humboldt University, Technical University Berlin and University of Potsdam. In 1999 he received the annual national prize for pattern recognition (Olympus Prize) awarded by the German pattern recognition society DAGM. He serves in the editorial board of Computational Statistics, IEEE Transactions on Biomedical Engineering and in program and organization committees of various international conferences. His research areas include statistical physics and statistical learning theory for neural networks, support vector machines and ensemble learning techniques. He contributed to the field of signal processing working on time-series analysis, statistical denoising methods and blind source separation. His present application interests are expanded to the analysis of biomedical data, most recently to brain computer interfacing and genomic data analysis.