COPYRIGHT NOTICE

This is the author's version of the work. The definitive version was published in *IEEE Transactions on Knowledge and Data Engineering* (TKDE), 18(2):145-160, February 2006.

DOI: `http://dx.doi.org/10.1109/TKDE.2006.29`.

Distance-Based Detection and Prediction of Outliers

Fabrizio Angiulli, Stefano Basta, Clara Pizzuti

Abstract—A distance-based outlier detection method that finds the top outliers in an unlabelled data set and provides a subset of it, called outlier detection solving set, that can be used to predict the outlierness of new unseen objects, is proposed. The solving set includes a sufficient number of points that permits the detection of the top outliers by considering only a subset of all the pairwise distances from the data set. The properties of the solving set are investigated, and algorithms for computing it, with sub-quadratic time requirements, are proposed. Experiments on synthetic and real data sets to evaluate the effectiveness of the approach are presented. A scaling analysis of the solving set size is performed, and the false positive rate, that is the fraction of new objects misclassified as outliers using the solving set instead of the overall data set, is shown to be negligible. Finally, to investigate the accuracy in separating outliers from inliers, ROC analysis of the method is accomplished. Results obtained show that using the solving set instead of the data set guarantees a comparable quality of the prediction, but at a lower computational cost.

Index Terms—Distance-based outliers, Outlier detection, Outlier prediction, data mining.

1 Introduction

Outlier detection in large data sets is an active research field in data mining [1] that has many applications in all those domains that can lead to illegal or abnormal behavior, such as fraud detection [2], network intrusion detection [3], [4], insurance fraud, medical diagnosis, marketing or customer segmentation. Many supervised approaches to outlier mining first learn a model over example data already labelled as exceptional or not [2], [3], and then evaluate a given input as normal or outlier depending on how well it fits the model. Unsupervised methods, instead, have the task to discriminate each datum as normal or exceptional when the training examples are not labelled. Among the unsupervised approaches, *distance-based* outlier detection methods distinguish an object as outlier on the base of the distance to its nearest neighbors [5], [6], [7], [4], [8]. These approaches differ in the way the distance measure is defined, but in general, given a data set D of objects, an object p can be associated with a weight or score, that is a function of the k nearest neighbors distances. Intuitively, the weight measures how an object is dissimilar from its neighbors. Let w^* be the n-th greatest weight of an object in D. An outlier w.r.t. D is an object scoring a weight w.r.t. D greater than or equal to w^*.

It is worth to point out that research on distance-based outlier detection has mainly aimed at developing methods to detect outliers in an input data set, rather than developing methods able to learn a model for predicting outliers in new incoming data. In these methods prediction on a new object p, to establish if it is an outlier, can be realized by computing the distances of p to all the objects in the data set and then comparing the obtained weight with that of the n-th outlier.

In this paper we focus on unsupervised distance-based outlier detection and prediction and we distinguish between these two tasks. *Outlier detection* consists in finding the top n outliers in D, that is the n objects of D having greatest weights. *Outlier prediction* corresponds to deciding if an incoming object is an outlier, that is if its weight w.r.t. D is greater than or equal to w^*.

We introduce the concept of *outlier detection solving set* S, a subset of D that includes a sufficient number of points from D to allow us to consider only the distances among the pairs in $S \times D$ to obtain the top n outliers. We then present an algorithm that computes the solving set and obtains the top n outliers in D and the weight w^*, by avoiding to calculate the distance of an object to each other to obtain its k nearest neighbors. Finally we show that the solving set S, besides containing the top n outliers in D, allows to effectively classify each new unseen object as outlier or not by approximating its weight w.r.t. D with its weight w.r.t. S. In fact, each new object can be classified as an outlier (w.r.t. D) if its weight w.r.t. S is above w^*. From this perspective, the solving set S is a *learned model* and can be seen as a *compressed representation of* D, analogously to sample compression schemes for a concept class [9] or to support vector extraction for classification tasks [10].

The contributions of this work can be summarized as follows:

- the concept of *outlier detection solving set*, a subset of the input data set representing a model that can be used to predict outliers, is defined;
- the computational complexity of computing a minimum cardinality solving set, showing that the problem is in general intractable, is analyzed;
- algorithms that compute with sub-quadratic time

The authors are with the ICAR-CNR Institute of High Performance Computing and Networking of the Italian National Research Council, Via Pietro Bucci 41C, 87036 Rende (CS), Italy. Email: {angiulli,basta,pizzuti}@icar.cnr.it

requirements a solving set and the top n outliers are provided;
- experimental evidence that the solving set is a fraction of the overall data set, and that the false positive rate obtained using the solving set is negligible is given;
- the ROC analysis of the method to investigate its accuracy in separating outliers from inliers is performed. Results obtained show that using the solving set instead of the data set to predict outliers is efficient and effective.

The paper is organized as follows. In the next section an overview on unsupervised distance-based outlier detection approaches is given. In Section 3 the problems that will be treated are formally defined. In Section 4 we define the concept of solving set and explain how to exploit it to solve the introduced problems. In Section 5 the complexity analysis of the task of computing a solving set is done. In Section 6 methods for computing the top n outliers and the solving set are described. Finally, Section 7 reports experimental results and discusses the choice of the parameters and the applicability of the approach proposed.

2 Related Work

In this section an overview of unsupervised distance-based outlier mining methods is given, though different approaches have been proposed [11], [12], [13], [14], [15].

The concept of *distance-based* outlier relies on the notion of neighborhood of a point, typically the k nearest neighbors, and has been first introduced by Knorr and Ng [5], [16]. Distance-based outlier are those points for which there are less than k points within the distance δ in the input data set. This definition does not provide a ranking of outliers and needs to determine an appropriate value of the parameter δ. The authors present two algorithms, the first one is a nested loop algorithm that runs in $O(dN^2)$ time, while the second one is a cell-based algorithm that is linear with respect to N, the number of points of the data set, but exponential in d, the number of dimensions of the data set. This last method is fast only if $d \leq 4$. On the other hand, the nested loop approach is impractical when outliers in large data sets have to be mined. Thus, efforts for developing efficient algorithms that scale to large real data sets, have been recently made.

Ramaswamy et al. [6] modified the definition of outlier introduced by Knorr and Ng and consider as outliers the top n points p whose distance to their k-th nearest neighbor is greatest. To detect outliers, a partition-based algorithm is presented that, first, partitions the input points using a clustering algorithm, and then prunes those partitions that cannot contain outliers. The experiments, up to 10 dimensions, show that the method scales well with respect to both data set size and dimensionality. This definition, however, does not take into account the information contained in the k-neighborhood of a point and thus it could not properly distinguish between dense or sparse neighborhood.

In [7] a new definition of distance-based outlier that takes into account the whole neighborhood by considering for each point p the sum of the distances from its k nearest neighbors, is proposed. This sum is called the $weight$ of p, $\Omega_k(p)$, and it is used to rank the points of the data set. Outliers are those points having the largest values of weight. $\Omega_k(p)$ is a more accurate measure of how much a point p can be considered an outlier, because it takes into account the sparseness of the neighborhood of a point. In order to compute these weights, the k nearest neighbors of each point are found in a fast and efficient way by linearizing the search space using the *Hilbert space filling curve*. The algorithm is able to deal with high dimensional data sets and scales near linearly with respect to both the dimensionality and the size of the data set.

An analogous definition of outlier based on the k-nearest neighbors has been used in [4] for unsupervised anomaly detection to detect intrusions in unlabelled data. Data elements are mapped in a feature space and anomalies are detected by determining which points lie in sparse regions of the feature space. Experiments on data sets of network connections and system call traces showed that the algorithms were able to find intrusions over unlabelled data.

More recently, Bay and Schwabacher [8] in order to find the top-n distance-based outliers of an input data set, augmented the naive distance-based nested loop algorithm, that finds the k nearest neighbors of each data set point, with a simple pruning rule and randomization obtaining a near linear scaling on real, large, and high dimensional data sets. The algorithm is sensitive to the order and to the distribution of the data set. If the data is sorted or correlated, the performance could be poor.

A detailed discussion on the usefulness, the meaning, and the intensional knowledge of distance-based outliers, as well as a description of the real life application domains for which this notion of outlier is relevant, can be found in [17], [16], [6], [4].

3 Problem Formulation

In this section we formally give the definition of the problems that will be treated. An *object* is represented by a set of d measurements (also called attributes or features).

Given a set D of objects, an object p, a distance dist on $D \cup \{p\}$, and a positive integer number i, the *i-th nearest neighbor* $nn_i(p, D)$ of p w.r.t. D is the object q of D such there exist exactly $i-1$ objects r of D (if p is in D then p itself must be taken into account) such that $\mathrm{dist}(p, q) \geq \mathrm{dist}(p, r)$. Thus, if p is in D then $nn_1(p, D) = p$, otherwise $nn_1(p, D)$ is the object of D closest to p.

Given a set D of N objects, a distance dist on D, an object p of D, and an integer number k, $1 \leq k \leq N$, the *weight* $w_k(p, D)$ of p in D (w.r.t. k) is $\sum_{i=1}^{k} \mathrm{dist}(p, nn_i(p, D))$.

Intuitively, the notion of weight captures the degree of dissimilarity of an object with respect to its neighbors, that is, the lower its weight is, the more similar its neighbors are. We denote by $D_{i,k}$, $1 \leq i \leq N$, the object of D scoring the i-th largest weight w.r.t. k in D, i.e. $w_k(D_{1,k}, D) \geq w_k(D_{2,k}, D) \geq \ldots \geq w_k(D_{N,k}, D)$.

Given a set D of N objects, a distance dist on D, and two integer numbers n and k, $1 \leq n, k \leq N$, the *Outlier Detection Problem* $ODP\langle D, \mathrm{dist}, n, k\rangle$ is defined as follows: find the n objects of D scoring the greatest weights w.r.t. k, i.e. the set $D_{1,k}, D_{2,k}, \ldots, D_{n,k}$. This set is called the *solution set* of the problem.

Given a set U of objects, a subset D of U having size N, an object q of U, called *query object* or simply *query*, a distance dist on U, and two integer numbers n and k, $1 \leq n, k \leq N$, the *Outlier Prediction Problem* $OPP\langle D, q, \mathrm{dist}, n, k\rangle$ is defined as follows: is $w_k(q, D) \geq w_k(D_{n,k}, D)$?

An object such that its weight w.r.t. D is equal to or greater than the weight of $D_{n,k}$ is said to be an *outlier* w.r.t. D (or equivalently a *D-outlier*), otherwise it is said to be an *inlier* w.r.t. D (or equivalently a *D-inlier*).

The ODP can be solved in $\mathcal{O}(|D|^2)$ time by computing all the distances $\{\mathrm{dist}(p, q) \mid (p, q) \in D \times D\}$, while a comparison of the query object q with all the objects in D, i.e. $\mathcal{O}(|D|)$ time, suffices to solve the OPP. Real life applications deal with data sets of hundred thousands or millions of objects, and thus these approaches are both not applicable.

4 Solving ODP and OPP

In this section we introduce the concept of *outlier detection solving set* and explain how to exploit it to solve both the ODP and the OPP.

Definition 4.1: Given a set D of N objects, a distance dist on D, and two integer numbers n and k, $1 \leq n, k \leq N$, an *outlier detection solving set* for the $ODP\langle D, \mathrm{dist}, n, k\rangle$ is a subset S of D such that:

1) $|S| \geq \max\{n, k\}$, and
2) Let $lb(S)$ denote the n-th greatest element of $\{w_k(p, D) \mid p \in S\}$. Then, for each $q \in (D - S)$, $w_k(q, S) < lb(S)$.

In the following the outlier detection solving set will be referred simply as solving set. Intuitively, a solving set S is a subset of D such that the distances $\{\mathrm{dist}(p, q) \mid p \in S, q \in D\}$ are sufficient to state that S contains the solution set of the ODP.

Let $n^* \geq n$ denote the positive integer such that $w_k(D_{n,k}, D) = w_k(D_{n^*,k}, D)$ and $w_k(D_{n^*,k}, D) > w_k(D_{n^*+1,k}, D)$. We call the set $\{D_{1,k}, \ldots, D_{n^*,k}\}$ the *extended solution set* of the $ODP\langle D, \mathrm{dist}, n, k\rangle$. The following Proposition proves that $S \supseteq \{D_{1,k}, \ldots, D_{n^*,k}\}$ and, hence, that $lb(S)$ is equal to $w_k(D_{n,k}, D)$.

Proposition 4.2: Let S be a solving set for the ODP $\langle D, \mathrm{dist}, n, k\rangle$. Then $S \supseteq \{D_{1,k}, \ldots, D_{n^*,k}\}$.

Proof: The proof is by contradiction. Assume that there exists i, $1 \leq i \leq n^*$, such that $D_{i,k} \notin S$. Then $D_{i,k} \in (D - S)$ and $w_k(D_{i,k}, S) \geq w_k(D_{i,k}, D) \geq lb(S)$, thus S is not a solving set. □

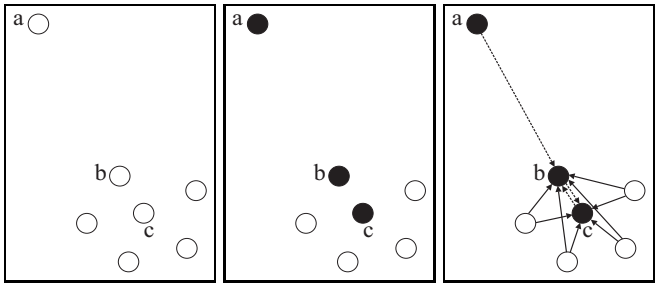

Fig. 1. An example of solving set for $n = 1$ and $k = 2$.

Clearly, D is a solving set, and the proposition states that a solving set contains at least the extended solution set $\{D_{1,k}, \ldots, D_{n^*,k}\}$. In general, an extended solution set does not suffice to form a solving set. For example, consider Figure 1, where a data set D of 7 points of $\mathbb{R}^2$ is shown. The top outlier for $k = 2$ is the point a. The black filled points in the figure on the center represent a solving set $S = \{a, b, c\}$ for $n = 1$ and $k = 2$. In the figure on the right, the solid arrows link points in $D - S$ to their 1-st and 2-nd nearest neighbor in S, while the dashed arrows link points in S to their 2-nd nearest neighbor in D (the 1-st nearest neighbor of each of these points in D is itself; note that, in general, the 2-nd nearest neighbor could not belong to S). As $k = 2$, the sum of the length of the solid arrows outgoing each point p of $D - S$, that is $w_2(p, S)$, is an upper bound to the weight of these points, while the length of the dashed arrows corresponds to the weight of each point of S. In particular, the distance from a to b represents $lb(S)$: S is a solving set as $lb(S) > w_2(p, S)$ for each point $p \in (D - S)$. S, of course, must contain a, the top outlier of D. Furthermore, neither the set $\{a, b\}$, nor the set $\{a, c\}$ are sufficient to form a solving set.

We propose to use the solving set S to solve the outlier prediction problem OPP in a fast way by computing the weight of the new objects with respect to the solving set S instead of the complete data set D. Hence, finding a solving set for ODP can be seen as a training phase in which the learned model is the solving set S. In practice, given $\mathcal{P} = ODP\langle D, \mathrm{dist}, n, k\rangle$ we first solve $\mathcal{P}$ by finding a solving set S for $\mathcal{P}$, and then we answer any $OPP\langle D, q, \mathrm{dist}, n, k\rangle$ in the following manner: reply "no" if $w_k(q, S) < lb(S)$ and "yes" otherwise. We say that q is an *S-outlier* if $w_k(q, S) \geq lb(S)$, and *S-inlier* otherwise.

Figure 2 shows an example of application of this approach. In Figure 2(a) there is a data set D of 1,000 points of $\mathbb{R}^2$. In figure 2(b) a solving set S (96 points) for $ODP\langle D, \mathrm{dist}, 10, 6\rangle$, where dist is the Euclidean distance, is depicted (the stars represent the solution set). Figure 2(c) shows the outlier prediction obtained using the solving set S for over 10,000 queries. In particular, the black points represent queries detected as S-inliers, while gray points are queries classified as S-outliers. Finally, in Figure 2(d) the lightness of the 10,000 points q

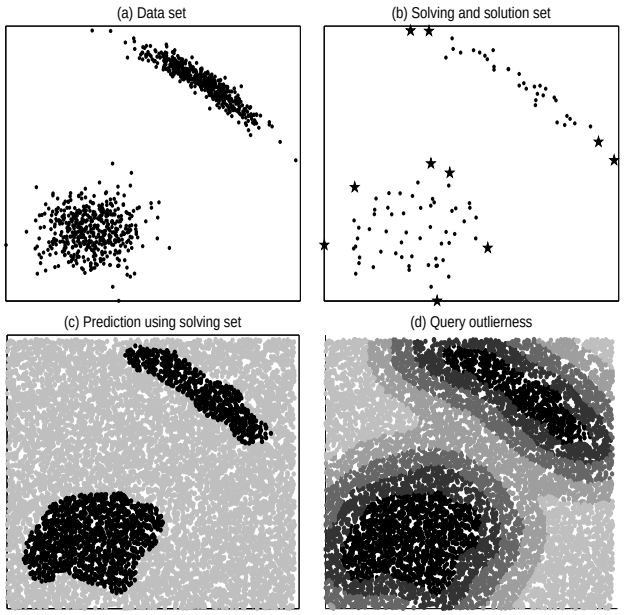

Fig. 2. An example of outlier prediction using solving sets.

is proportional to the ratio $\frac{w_k(q,S)}{lb(S)}$. The figure shows that more distant a query object is from the data set, higher is its weight.

		Predicted label w.r.t. S	
		S-inlier	S-outlier
Label w.r.t. D	D-inlier	*true negative*	*false positive*
	D-outlier	*false negative*	*true positive*

TABLE 1
Classification of the query objects.

To evaluate the quality of the solution that can be obtained for the OPP using the solving set, we adopt two standard measures developed for intrusion detection systems [4], [18]: *false positive rate* and *detection rate*. These measures are based on the classification of the query objects reported in Table 1. The false positive rate is the ratio between the number of queries that are incorrectly misclassified as outliers (i.e the false positives) and the total number of normal data (the D-inliers, i.e. both the false positives and the true negatives). The detection rate is the ratio between the number of correctly detected outliers (i.e. the true positives) and the total number of outliers (the D-outliers, i.e. both the true positives and the false negatives). In our approach the detection rate is always 1 as we cannot have false negatives. Indeed, for each query object q, $w_k(q,S) \geq w_k(q,D)$, being $S \subseteq D$. This guarantees that, if a query object is a D-outlier, the solving set correctly predicts it. On the other hand the solving set could misclassify a query object as outlier, though it is a normal one.

To be effective a solving set S should satisfy the following desiderata:

- *efficiency*: S must be efficiently computable. We provide an algorithm that solves the ODP and, at the same time, calculates a solving set with subquadratic time requirements;
- *smallness*: the size of S must be small w.r.t. the size of D. We present a thorough analysis of the solving set size, on both real and synthetic data, showing that in practice S is composed by a small percentage of the objects of the data set;
- *meaningfulness*: the outliers detected using the solving set S must be comparable with those obtained by using the entire data set D. We show that the false positive rate obtained is negligible;
- *consistency*: S must guarantee to predict the correct label for all the objects in D.

S actually does not guarantee this latter property for those objects belonging to $(S - \{D_{1,k}, \ldots, D_{n^*,k}\})$. To this aim, next we introduce the concept of *robust solving set* that assures consistency.

Definition 4.3: Given a solving set S for the $ODP\langle D, \mathrm{dist},\ n, k\rangle$ and an object $p \in D$, we say that p is *S-robust* if the following holds: $w_k(p,D) < lb(S)$ iff $w_k(p,S) < lb(S)$. We say that S is *robust* if, for each $p \in D$, p is S-robust.

Thus, if S is a robust solving set for the $ODP\langle D, \mathrm{dist}, n, k\rangle$, then an object p, occurring in the data set D, is a D-outlier for the $OPP\langle D, p, \mathrm{dist}, n, k\rangle$ iff it is an S-outlier. Hence, from the point of view of the data set objects, the sets D and S are equivalent.

In the next section we analyze the complexity of the task of computing a solving set and a robust solving set.

5 On the complexity of computing an ODP solving set

In general, given a set of objects D and fixed n and k, there can be many different solving sets for D. We are interested in finding a minimal solving set or, possibly, the smallest solving set.

Definition 5.1: A solving set S for $\mathcal{P} = ODP\langle D, \mathrm{dist}, n, k\rangle$ is *minimal* if every proper subset S' of S is not a solving set for $\mathcal{P}$. A *minimum solving set* S for $\mathcal{P}$ is a solving set S for $\mathcal{P}$ such that, for each solving set S' for $\mathcal{P}$, $|S| \leq |S'|$.

Proposition 5.2: A solving set S is minimal iff there does not exist $S' \subseteq S$, $|S'| = |S| - 1$, such that S' is a solving set.

Proof: We note that every superset of a solving set is a solving set itself. Thus, if S is not minimal, it properly contains a solving set S'', then it must contain also a solving set $S' \supseteq S''$ such that $|S'| = |S| - 1$. □

A minimal solving set can be computed by exploiting Proposition 5.2 as follows: let $D = \{p_1, \ldots, p_N\}$; (i) set S to D; (ii) for $i = 1, \ldots, N$, if $S - \{p_i\}$ is a solving set, then set $S = S - \{p_i\}$. When the cycle terminates, the set S contains a minimal solving set.

It is relevant to our task to characterize the complexity of the problem of finding a minimum solving set. This

is closely related to the complexity of the following decision problem.

Problem 5.3: Given a set D of N objects, a distance dist on D, and three integer numbers n, k, and t, $1 \leq n, k \leq N$, $\max\{n,k\} \leq t \leq N$, the *ODP Solving Set Problem* $SSP\langle D, \mathrm{dist}, n, k, t\rangle$ is the following: does there exist a solving set S for the $ODP\langle D, \mathrm{dist}, n, k\rangle$ such that $|S| \leq t$?

Theorem 5.4: Let dist be a metric and $k = 2$. Then $SSP\langle D, \mathrm{dist}, n, k, t\rangle$ is NP-complete.

Proof: Membership is straightforward. As for the hardness, the proof is by reduction to a variant of the well-known NP-complete problem *Dominating Set Problem* [19], variant that we called *k-Dominating Set Problem*. Let $G = (V, E)$ be an undirected graph, and let $k, t \leq |V|$ be two positive integers. The *k-Dominating Set Problem* is defined as follows: is there a subset $U \subseteq V$, called *k-dominating set* of G, with $|U| \leq t$, such that for all $v \in (V - U)$ there are distinct $u_1, \ldots, u_k \in U$ with, $\forall i : 1 \leq i \leq k$, $\{v, u_i\} \in E$? The Dominating Set Problem corresponds to the 1-Dominating Set Problem. The k-Dominating Set Problem can be proved to be NP-complete by reducing the Dominating Set Problem to it through the transformation described in Proposition 3.2 of [20]. Let U be a k-dominating set of G. For each node $v \in (V - U)$, let $\pi_U^1(v), \ldots, \pi_U^k(v)$ denote k distinct nodes of U such that, $\forall i : 1 \leq i \leq k$, $\{v, \pi_U^i(v)\} \in E$.

Let $G = (V, E)$ be an undirected graph, and let $t \leq |V|$ be a positive integer. Without loss of generality, suppose that each node of G has degree at least 2. Let D_G be the set of objects $V \cup O$, where O is the set of new objects $\{o_1, \ldots, o_n\}$. Let $0 < \epsilon \leq \frac{1}{3}$. Define the distance dist_G on D_G as follows:

- $\forall \{u, v\} \in E$, $\mathrm{dist}_G(u, v) = 1 - \epsilon$;
- $\forall u, v \in V$, $u \neq v$, $\{u, v\} \notin E$, $\mathrm{dist}_G(u, v) = 1 + \epsilon$;
- $\forall u \in V \cup O$, $\forall v \in O$, $u \neq v$, $\mathrm{dist}_G(u, v) = \mathrm{dist}_G(v, u) = 2$;
- $\forall u \in D_G$, $\mathrm{dist}_G(u, u) = 0$.

We note that $\epsilon \leq \frac{1}{3}$ implies that $1 + \epsilon \leq 2(1 - \epsilon)$. This proves that dist_G is a metric. Indeed, the constraint $2 \leq 2(1-\epsilon)$, although not satisfied, is not relevant as for each distinct $u, v, w \in D_G$ such that $\mathrm{dist}_G(u, v) = 2$ and $\mathrm{dist}_G(v, w) = 1 - \epsilon$, it holds that $\mathrm{dist}_G(u, w) = 2$, and, thus, $\mathrm{dist}_G(u, v) \leq \mathrm{dist}_G(u, w) + \mathrm{dist}_G(w, v)$.

Furthermore, We note that O is the solution set of the $ODP\langle D_G, \mathrm{dist}_G, n, 2\rangle$. In fact, (i) for each $o \in O$, $w_2(o, D_G) = 0 + 2 = 2$, and, (ii) for each $u \in V$, $w_2(u, D_G) = 0 + (1 - \epsilon) = 1 - \epsilon < 2$, as, having u degree at least 2, there exists an object $v \in V$ such that $\mathrm{dist}_G(u, v) = 1 - \epsilon$. Now we prove that G has a 2-dominating set of size t iff $SSP\langle D_G, \mathrm{dist}_G, n, 2, n + t\rangle$ is a YES instance.

($\Rightarrow$) Suppose that G has a 2-dominating set U such that $|U| \leq t$. We show that $S = O \cup U$ is a solving set for $ODP\langle D_G, \mathrm{dist}_G, n, 2\rangle$. First, we note that $lb(S) = 2$, as $S \supseteq O$. For each $v \in (D_G - S) = (V - U)$, as $S \supseteq \{\pi_U^1(v), \pi_U^2(v)\}$ then $w_2(v, S) \leq (1-\epsilon)+(1-\epsilon) = 2-2\epsilon < 2 = lb(S)$. Thus S is a solving set for $\langle D_G, \mathrm{dist}, n, 2\rangle$.

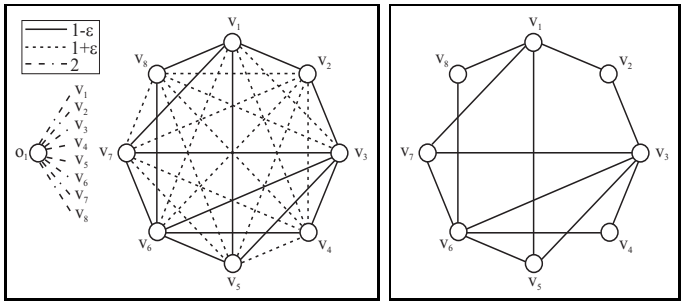

Fig. 3. A graph of eight nodes (left) and an example of reduction described in Theorem 5.4, for $n = 1$ (right).

($\Leftarrow$) The proof is by contradiction. Suppose that there exists a solving set S for the ODP $\langle D_G, \mathrm{dist}_G, n, 2\rangle$ such that $|S| \leq n + t$. By Proposition 4.2, $S \supseteq O$. Assume that there exists $v \in (V - S)$ such that, for each distinct $v_1, v_2 \in (S \cap V) = U$, either $\{v, v_1\} \notin E$ or $\{v, v_2\} \notin E$. Then $w_2(v, S) \geq (1 - \epsilon) + (1 + \epsilon) = 2 \geq lb(S)$, and S is not a solving set: contradiction. It follows immediately that U is a 2-dominating set for G such that $|U| \leq t$. □

Figure 3 shows an example of the reduction described in Theorem 5.4, for $n = 1$. The matrix associated with the distance dist_G is the following:

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \\ o_1 \end{array} \begin{array}{c} \begin{array}{ccccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 & o_1 \end{array} \\ \begin{pmatrix} 0 & 1-\epsilon & 1+\epsilon & 1+\epsilon & 1-\epsilon & 1+\epsilon & 1-\epsilon & 1-\epsilon & 2 \\ 1-\epsilon & 0 & 1-\epsilon & 1+\epsilon & 1+\epsilon & 1+\epsilon & 1+\epsilon & 1+\epsilon & 2 \\ 1+\epsilon & 1-\epsilon & 0 & 1-\epsilon & 1-\epsilon & 1-\epsilon & 1-\epsilon & 1+\epsilon & 2 \\ 1+\epsilon & 1+\epsilon & 1-\epsilon & 0 & 1+\epsilon & 1-\epsilon & 1+\epsilon & 1+\epsilon & 2 \\ 1-\epsilon & 1+\epsilon & 1-\epsilon & 1+\epsilon & 0 & 1-\epsilon & 1+\epsilon & 1+\epsilon & 2 \\ 1+\epsilon & 1+\epsilon & 1-\epsilon & 1-\epsilon & 1-\epsilon & 0 & 1-\epsilon & 1-\epsilon & 2 \\ 1-\epsilon & 1+\epsilon & 1-\epsilon & 1+\epsilon & 1+\epsilon & 1-\epsilon & 0 & 1+\epsilon & 2 \\ 1-\epsilon & 1+\epsilon & 1+\epsilon & 1+\epsilon & 1+\epsilon & 1-\epsilon & 1+\epsilon & 0 & 2 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 0 \end{pmatrix} \end{array}$$

The set $U = \{v_1, v_3, v_6\}$ is a 2-dominating set of G, thus, by Theorem 5.4, $\{v_1, v_3, v_6, o_1\}$ is a solving set S for the ODP $\langle D_G, \mathrm{dist}, 1, 2\rangle$, and, hence, the following sub-matrix suffices to solve this problem:

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \\ o_1 \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_3 & v_6 & o_1 \end{array} \\ \begin{pmatrix} 0 & 1+\epsilon & 1+\epsilon & 2 \\ 1-\epsilon & 1-\epsilon & 1+\epsilon & 2 \\ 1+\epsilon & 0 & 1-\epsilon & 2 \\ 1+\epsilon & 1-\epsilon & 1-\epsilon & 2 \\ 1-\epsilon & 1-\epsilon & 1-\epsilon & 2 \\ 1+\epsilon & 1-\epsilon & 0 & 2 \\ 1-\epsilon & 1-\epsilon & 1-\epsilon & 2 \\ 1-\epsilon & 1+\epsilon & 1-\epsilon & 2 \\ 2 & 2 & 2 & 0 \end{pmatrix} \end{array}$$

Theorem 5.5: Let dist be a metric and let $k > 2$. Then $SSP\langle D, \mathrm{dist}, n, k, t\rangle$ is NP-complete.

Proof: Membership is straightforward. As for the hardness the proof is by reduction to the *k-Dominating Set Problem*. Let $G = (V, E)$ be an undirected graph, and let $2 < k \leq |V|$ and $t \leq |V|$ be two positive integers. Without loss of generality, suppose that each node of G has degree at least k. Let D_G be the set of objects $V \cup O$, where O is the set of new objects $\{o_1, \ldots, o_m\}$,

with $m = \max\{n, k\}$. Let

$$a = 1,\ b = 2c,\ \text{and}\ c = \frac{2k-2}{2k+1}.$$

Define the distance dist_G on D_G as follows:

- $\forall\{u, v\} \in E$, $\text{dist}_G(u, v) = c$;
- $\forall u, v \in V$, $u \neq v$, $\{u, v\} \notin E$, $\text{dist}_G(u, v) = b$;
- $\forall u, v \in O$, $u \neq v$, $\text{dist}_G(u, v) = a$;
- $\forall u \in V$, $\forall v \in O$, $\text{dist}_G(u, v) = \text{dist}_G(v, u) = b$;
- $\forall u \in D_G$, $\text{dist}_G(u, u) = 0$.

We note that $\frac{4}{7} \leq c < a = 1 < b = 2c < 2$, and, thus, (i) $b \leq 2c$ and (ii) $b \leq 2a$. This proves that dist_G is a metric. Indeed, the constraint (iii) $a \leq 2c$, although not satisfied, is not relevant since there not exist $u, v, w \in D_G$ such that $\text{dist}_G(u, v) = a$ and $\text{dist}_G(v, w) = c$.

We note that O contains the solution set of the $ODP\langle D_G,\ \text{dist}_G, n, k\rangle$. In fact, (i) for each $o \in O$, $w_k(o, D_G) = 0 + a(k-1) = k-1$, and, (ii) for each $u \in V$, $w_k(u, D_G) = 0 + (k-1)c = \frac{2(k-1)^2}{2k+1} < k-1$, as, having u degree at least k, there exist $k-1$ objects $v_1, \ldots, v_{k-1} \in V$ such that $\forall i : 1 \leq i \leq k-1$, $\text{dist}_G(u, v_i) = c$.

Now we prove that G has a k-dominating set of size t iff $SSP\langle D_G, \text{dist}_G, n, k, m+t\rangle$ is a YES instance.

($\Rightarrow$) Suppose that G has a k-dominating set U such that $|U| \leq t$. Now we show that $S = O \cup U$ is a solving set for $ODP\langle D_G, \text{dist}_G, n, k\rangle$. First, we note that $lb(S) = k-1$, as $S \supseteq O$. For each $v \in (D_G - S) = (V - U)$, as $S \supseteq \{\pi_U^1(v), \ldots, \pi_U^k(v)\}$ then $w_k(v, S) \leq kc = k\frac{2k-2}{2k+1} < k-1 = lb(S)$. Thus S is a solving set for $\langle D_G, \text{dist}, n, k\rangle$.

($\Leftarrow$) The proof is by contradiction. Suppose that there exists a solving set S for the $ODP\ \langle D_G, \text{dist}_G, n, k\rangle$ such that $|S| \leq m + t$. By Proposition 4.2, $S \supseteq O$. Assume that there exists $v \in (V - S)$ such that, for each distinct $u_1, \ldots, u_k \in (S \cap V) = U$, it exists $i : 1 \leq i \leq k$, with $\{v, u_i\} \notin E$. Then $w_k(v, S) \geq (k-1)c + b = (k+1)c = \frac{2k+2}{2k+1}(k-1) > k-1 = lb(S)$, and S is not a solving set: contradiction. It follows immediately that U is a k-dominating set for G such that $|U| \leq t$. □

We note that the solving set $S = \{v_1, v_3, v_6, o_1\}$ of the previous example is robust, as $lb(S) = 2$, $w_2(v_1, D_G) = 1 - \epsilon$, $w_2(v_3, D_G) = 1 - \epsilon$, $w_2(v_6, D_G) = 1 - \epsilon$, while $w_2(v_1, S) = 1 + \epsilon < lb(S)$, $w_2(v_3, S) = 1 - \epsilon$, and $w_2(v_6, S) = 1 - \epsilon < lb(S)$, as it can be verified by using the following matrix of distances:

$$\begin{array}{c} \\ v_1 \\ v_3 \\ v_6 \\ o_1 \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_3 & v_6 & o_1 \end{array} \\ \begin{pmatrix} 0 & 1{+}\epsilon & 1{+}\epsilon & 2 \\ 1{+}\epsilon & 0 & 1{-}\epsilon & 2 \\ 1{+}\epsilon & 1{-}\epsilon & 0 & 2 \\ 2 & 2 & 2 & 0 \end{pmatrix} \end{array}$$

It is easy to verify that, in the construction of the Theorem 5.4, if there exists a solving set then it is robust. Thus, searching for a robust solving set is as hard as searching for a solving set.

Theorem 5.6: Let $k > 1$ and let dist be a metric. Then the *robust* $SSP\langle D, \text{dist}, n, k, t\rangle$ is NP-complete.

Proof: The proof is analogous to that of Theorems 5.4 and 5.5. □

Thus, from the complexity analysis above done the following theorem finally holds.

Theorem 5.7: Let $k > 1$ and let dist be a metric. Then computing a minimum cardinality (robust) solving set is NP-hard.

6 Algorithms for Computing ODP Solving Sets

In this section we describe three algorithms that compute, respectively, a solving set, a robust solving set, and a minimal robust solving set.

SolvingSet algorithm. The SolvingSet algorithm solves the Outlier Detection Problem and calculates a solving set for it. It computes the weights of the data set objects by comparing each object with a selected small subset of the overall data set, called $Cand$, and storing its k neighbors found so far with respect to $Cand$. The current weight of an object is thus an upper bound to its true weight. The objects having weight lower than the n-th greatest weight so far calculated are called *non-active*, while the others are called *active*. At the beginning $Cand$ contains randomly selected objects from D, while, at each step, it is built by selecting, among the active points of the data set not already inserted in $Cand$ during previous steps, a mix of random objects and objects having the maximum current weights.

During the execution, if an object becomes non active, then it will not be considered any more for insertion in the set $Cand$, because it can not be an outlier. As the algorithm processes new objects, more accurate weights are computed and the number of non-active objects increases more quickly. The algorithm stops when no other object can be examined, i.e. all the objects not yet inserted in $Cand$ are non-active, and thus $Cand$ becomes empty. The solving set is the union of the sets $Cand$ computed during each step.

The algorithm SolvingSet (see figure 4) receives in input the data set $Data$, containing N objects, the distance dist on $Data$, the number k of neighbors to consider for the weight calculation, the number n of top outliers to find, an integer $m \geq k$, and a rational number $r \in [0, 1]$. In particular, m represents the number of objects to select from the data set for insertion in the set $Cand$ at each step of the algorithm, while r specifies the trade-off between the number of random objects ($(1-r)m$ objects) and the number of objects having the greatest current weights (rm objects) to select for insertion in $Cand$.

Each object p of the data set is associated with a heap $NN[p]$, whose elements are pairs $\langle q, \delta\rangle$, where q is an object and δ a distance. Each heap stores the pairs having the k smallest distances to p.

The function $\texttt{Sum}(NN[p])$, for a generic p, returns the sum of the distances δ in the heap $NN[p]$, that is, an *upper bound to the weight of the object*.

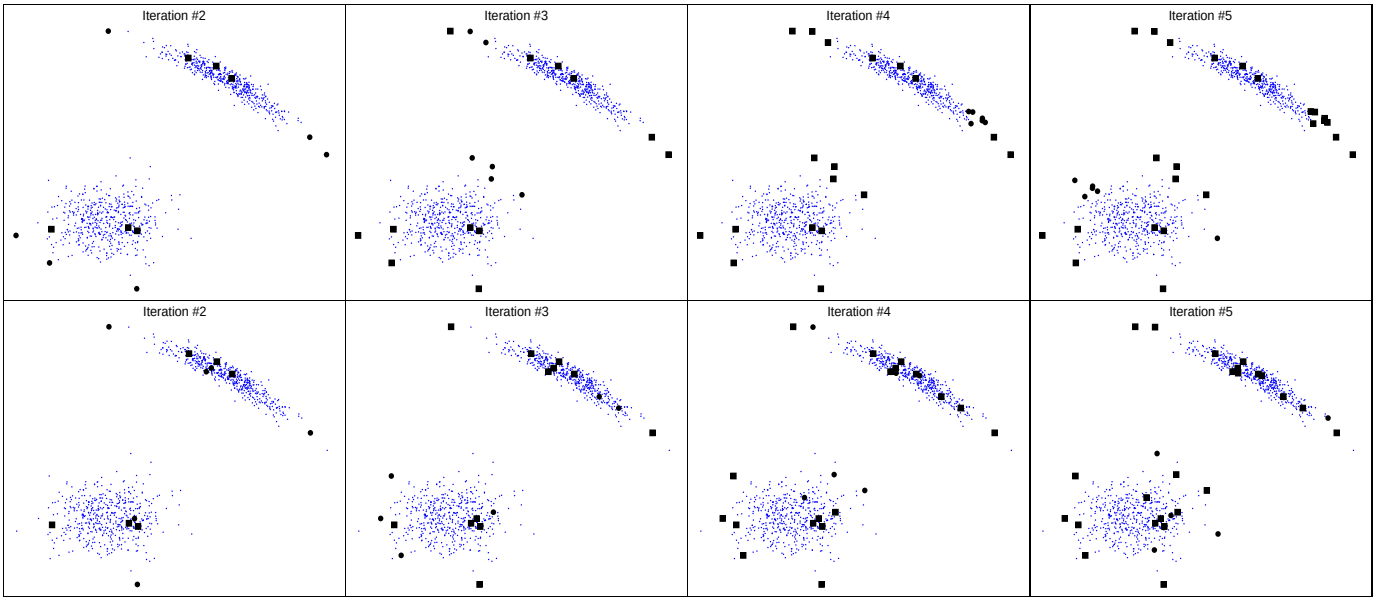

Fig. 5. Two examples of execution of the SOLVINGSET algorithm for values of the parameter $r = 1$, top figures, and $r = 0.5$, bottom figures. Black squares constitute points of $SolvSet$, while black circles constitute the set $Cand$.

```
SOLVINGSET(Data, dist, n, k, m, r) {
  SolvSet = ∅;
  Top = ∅;
  Cand = RandomSelect(Data, m);
  while (Cand ≠ ∅) {
    SolvSet = SolvSet ∪ Cand;
    Data = Data − Cand;
    for each (p in Cand)
      for each (q in Cand) {
        δ = dist(p, q);
        UpdateMin(NN[p], ⟨q, δ⟩);
        if (p ≠ q) UpdateMin(NN[q], ⟨p, δ⟩);
      }
    NextCand = ∅;
    for each (p in Data) {
      for each (q in Cand)
        if max{Sum(NN[p]), Sum(NN[q])} ≥ Min(Top)) {
          δ = dist(p, q);
          UpdateMin(NN[p], ⟨q, δ⟩);
          UpdateMin(NN[q], ⟨p, δ⟩);
        }
      UpdateMax(NextCand, ⟨p, Sum(NN[p])⟩);
    }
    for each (q in Cand)
      UpdateMax(Top, ⟨q, Sum(NN[q])⟩);
    Cand = CandSelect(NextCand, Data − NextCand, r);
  }
}
```

Fig. 4. The algoritm SOLVINGSET.

The function $\texttt{UpdateMin}(NN[p], \langle q, \delta\rangle)$ updates the heap associated with p, by substituting the pair $\langle s, \sigma\rangle$, where σ is the maximum distance in $NN[p]$, with $\langle q, \delta\rangle$, in the case $\delta < \sigma$.

Furthermore, the algorithm uses two heaps Top and $NextCand$, of, respectively, n and m data set objects. The objects stored in Top are those having the greatest true weights computed so far, while those contained in $NextCand$ are the objects of $Data$ having the greatest weight upper bound. Both the heaps thus store pairs $\langle p, \sigma\rangle$, where p is an object and σ is $\texttt{Sum}(NN[p])$.

The function $\texttt{Min}(Top)$ returns the smallest value σ associated with a pair stored in the heap Top. Thus it is a *lower bound to the weight of the n-th outlier of $Data$*. This means that the objects q of $Data$ having weight upper bound $\texttt{Sum}(NN[q])$ less than $\texttt{Min}(Top)$ cannot belong to the solution set.

The function $\texttt{UpdateMax}(H, \langle p, \texttt{Sum}(NN[p])\rangle)$ updates the heap H, by substituting the pair $\langle s, \sigma\rangle$, where σ is the minimum weight upper bound in H, with $\langle p, \texttt{Sum}(NN[p])\rangle$, in the case $\texttt{Sum}(NN[p]) > \sigma$.

$Cand$ is initialized by picking at random m elements from the data set and it must contain those points of $Data$ for which the exact weight has been computed. To this end the objects of $Cand$ are compared with all the points in $Cand \cup Data$. This is realized by splitting the comparison with the objects in $Cand$ and in $Data - Cand$ in two steps in order to avoid useless distance computations. Thus, the first double cycle compares the objects of $Cand$ with themselves and, for each of them, updates the associated heap through the function $UpdateMin$. In the second double cycle the distance between a point p in $Data$ and a point q in $Cand$ is computed only if at least one of the two can be an outlier, that is when the maximum between their upper bounds $Sum(NN[p])$ and $Sum(NN[q])$ to their weights is greater than the lower bound $Min(Top)$. Thus, at the end of this double cycle, the objects in $Cand$ have weight upper bound equal to their true weight, and they can be inserted in the heap Top.

It remains to see how the function `CandSelect` populates the set $Cand$ for the next iteration. This function builds a set composed by rm active objects in $NextCand$

and by $(1-r)m$ active objects in $Data$ but not in $NextCand$. We note that $NextCand$ contains the m objects of $Data$ having the greatest weight upper bound.

If there are no more active points, then `CandSelect` returns the empty set and the algorithm stops: Top contains the solution set and $SolvSet$ is a solving set for the ODP.

The algorithm SOLVINGSET has worst case time complexity $\mathcal{O}(|D|^2)$, but practical complexity $\mathcal{O}(|D|^{1+\beta})$, with $\beta < 1$. Indeed, let S be the solving set computed by the algorithm, then it performed $|D|^{1+\beta} = |D| \cdot |S|$ distance computations, and thus $\beta = \frac{\log |S|}{\log |D|}$.

Figure 5 shows two examples of execution of the algorithm SOLVINGSET on the data set of Figure 1. The first row reports the iterations 2 to 5 of the algorithm for $n = 10$, $k = m = 6$, and $r = 1.0$, while the second row reports the iterations 2 to 5 for $n = 10$, $k = m = 6$, and $r = 0.5$. Big filled circles represent the points of the set $Cand$ during the current iteration, while big filled squares represent the points composing the set $SolvSet$ at the beginning of the current iteration (see Figure 4). During the first iteration, the algorithm SOLVINGSET randomly selects six points (the squares in the left top and bottom figures), and then the six points to be inserted in $Cand$ during the next iteration are computed (these points are the circles in the left top and bottom figures). Notice that the points to be inserted in $Cand$ during the next iteration selected for $r = 1.0$ (top row) are always those more distant from the current solving set $SolvSet$, while for $r = 0.5$ (bottom row) they are a mix of randomly selected points and points whose distance to $SolvSet$ is maximum. For example, in the top left figures, the six circles are among the most outlying objects in the data set, while in the bottom left figure only three circles are among these objects, the other three circles are picked within the two distributions. We will discuss in the following section how the parameter r affects the convergence of the algorithm, but here we can observe that higher values of r bias the algorithm to insert outliers early in the solving set $SolvSet$.

RobustSolvingSet algorithm. The solving set S computed by the SOLVINGSET algorithm is not robust. Now we show how a robust solving set R containing S can be computed with no additional asymptotic time complexity. The following proposition states a necessary and sufficient condition that a solving set S must satisfy in order to be robust.

Proposition 6.1: A solving set S for the $ODP\langle D, \text{dist}, n, k\rangle$ is robust iff for each $p \in (S - \{D_{1,k}, \ldots, D_{n^*,k}\})$, $w_k(p, S) < lb(S)$.

Proof: We note that each object in $\{D_{1,k}, \ldots, D_{n^*,k}\} \cup (D - S)$ is S-robust by definition of solving set. Thus, as the objects p in $(S - \{D_{1,k}, \ldots, D_{n^*,k}\})$ are such that $w_k(p, D) < lb(S)$, then it must be the case that $w_k(p, S)$ is less than $lb(S)$. □

By Proposition 6.1 we have to verify that, for each object p of S, if the weight of p in D is less than $lb(S)$, then the weight of p in S remains below the same threshold. Thus, the algorithm ROBUSTSOLVINGSET does the following: (i) initialize the robust solving set R to S; (ii) for each $p \in S$, if $w_k(p, D) < lb(S)$ and $w_k(p, R) \geq lb(S)$, then select a set C of neighbors of p coming from $D - S$, such that $w_k(p, R \cup C) < lb(S)$, and set R to $R \cup C$. We note that the objects C added to R are certainly R-robust, as they are S-robust by definition of solving set and $R \supseteq S$. Furthermore, we note that the objects C satisfying the above property can be efficiently retrieved, as they are always stored in the heap $NN[p]$.

```
ROBUSTSOLVINGSET(Data, dist, n, k, m, r) {
  SolvSet = SOLVINGSET(Data, dist, n, k, m, r);
  RobuSolvSet = SolvSet;
  for each (p in SolvSet)
    if (Sum(NN[p]) < Min(Top)) {
      RNN = ∅;
      for each (q in RobuSolvSet)
        UpdateMin(RNN, ⟨q, dist(p, q)⟩);
      if (Sum(RNN) ≥ Min(Top)) {
        DNN = Sort(NN[p] − RNN);
        j = 1;
        while (Sum(RNN) ≥ Min(Top)) {
          ⟨q, δ⟩ = DNN[j];
          UpdateMin(RNN, ⟨q, δ⟩);
          RobuSolvSet = RobuSolvSet ∪ {q};
          j = j + 1;
        }
      }
      NN[p] = RNN;
    }
  return(RobuSolvSet);
}
```

Fig. 6. The algorithm ROBUSTSOLVINGSET.

In particular, the algorithm (see Figure 6) uses the heap RNN to store, for each object p of the solving set $SolvSet$, such that (`Sum`$(NN[p]) <$ `Min`(Top)), its k nearest neighbors in the robust solving set $RobuSolvSet$. If `Sum`(RNN) is greater than `Min`(Top) then p is not $RobuSolvSet$-robust. In this case, the pairs $\langle q, \delta\rangle$ occurring in $NN[p]$ but not in RNN (i.e. the nearest neighbor of p in $Data$ but not in $SolvSet$) are stored in the array DNN and then sorted w.r.t. the field δ, and, finally, the set $RobuSolvSet$ is augmented picking one object q at time from DNN until $w_k(p, RobuSolvSet)$ decreases below the threshold `Min`(Top). Notice that at the end of ROBUSTSOLVINGSET, in the matrix NN there are only objects coming from the robust solving set.

The size of the set R, computed by the ROBUSTSOLVINGSET algorithm, is upper bounded by $k|S|$, even if it is unlikely that this size is reached in practice, as confirmed by experimental results described in the next section. If we ignore the contribution of the subprogram SOLVINGSET, then ROBUSTSOLVINGSET has worst case time complexity $\mathcal{O}(|R| \cdot |S|)$. Thus, as $|R| \leq |D|$ (and we expect that $|R| \ll |D|$ in practice), the time complexity of ROBUSTSOLVINGSET is dominated by the complexity of the subprogram SOLVINGSET.

```
MINIMALROBUSTSOLVINGSET(Data, dist, n, k, m, r) {
  RobuSolvSet=ROBUSTSOLVINGSET(Data, dist, n, k, m, r);
  MinRobuSolvSet = RobuSolvSet;
  for each (p in RobuSolvSet)
    if (Sum(NN[p]) < Min(Top)) {
      RemoveObject = true;
      for each (q in Data)
        if (Sum(NN[q]) < Min(Top) and Member(NN[q], p))
        {
          MNN = ∅;
          for each (r in (MinRobuSolvSet − {p}))
            UpdateMin(MNN, ⟨r, dist(q, r)⟩);
          if (Sum(MNN) < Min(Top))
            NN[q] = MNN;
          else {
            RemoveObject = false;
            break;
          }
        }
      if (RemoveObject)
        MinRobuSolvSet = MinRobuSolvSet − {p};
    }
  }
  return(MinRobuSolvSet);
}
```

Fig. 7. The algorithm MINIMALROBUSTSOLVINGSET.

MinimalRobustSolvingSet algorithm. The set R computed by ROBUSTSOLVINGSET is robust, but not minimal. Thus, optionally, we can minimize the robust solving set R to obtain a minimal robust solving set M.

The algorithm MINIMALROBUSTSOLVINGSET is shown in Figure 7 and computes the set M. First, M is set equal to R. Then, for each object p in R having weight upper bound less than $lb(S)$ (recall that, by Proposition 4.2, a solving set must contain the solution set), the algorithm checks if $M - \{p\}$ is a robust solving set, i.e. if, for each q of $Data$ such that $w_k(q, M) \leq lb(S)$, it holds that $w_k(q, M - \{p\}) \leq lb(S)$. If it is the case, then p is removed from M. Notice that the algorithm avoids the computation of $w_k(q, M - \{p\})$ for the objects q that do not have p occurring in $NN[q]$. During these operations the heap $NN[q]$ is updated to the heap MNN, containing the k nearest neighbors of q in $M - \{p\}$, even if we are not sure that p will be removed from M. We point out that this early update does not lead to any inconsistency, as, to work properly, the algorithm requires that the property "$\texttt{Sum}(NN[q]) < lb(M)$ iff $w_k(q, D) < lb(M)$" holds, and the property is preserved by this update. The worst case complexity of MINIMALROBUSTSOLVINGSET is $\mathcal{O}(|R|^2 \cdot |D|)$, so this is the most expensive task among the three considered, but it is worth to note that it is unlikely that this number of operations is needed in practice.

7 EXPERIMENTAL RESULTS

In this section we present experiments on synthetic and real data sets to evaluate the effectiveness of our approach. In particular, first we consider a two dimensional synthetic data set to permit the graphical visualization of the various concepts introduced, to study the influence of the parameter r on the method, and to compare the prediction quality obtained answering the OPP using solving sets with those obtained exploiting random samples of the data set (Section 7.1). Then, we use a synthetic and four real high-dimensional data sets to study how the size of the solving set scales w.r.t. the parameters n and k (Section 7.2), and to analyze the false positive rate (Section 7.3). Finally, we use four real labelled data sets to compare the accuracy of a solving set and of the overall corresponding data set in separating data from a certain class – assumed to represent normal data, from data of the other classes – assumed to represent exceptions (Section 7.4). Some considerations arguing the applicability of the approach close the section (Section refdisc).

r	$\lvert S\rvert$	$\lvert R\rvert$	$\lvert M\rvert$
0.00	13,378	13,778	5,939
0.25	5,410	6,455	5,543
0.50	5,092	6,091	5,505
0.75	5,123	6,080	5,567
1.00	5,588	6,519	5,665

TABLE 2
Solving set size $|S|$, robust solving set size $|R|$, and minimal robust solving set size $|M|$ for the example of Figure 8.

7.1 Two dimensional example

The data set, shown in Figure 8(a) together with the top $n = 200$ outliers for $k = 20$ (the dark stars), is composed by $N = 100{,}000$ points in the square $[-2.5, 2.5]^2$. We computed these outliers using the SOLVINGSET algorithm. In particular, we executed the algorithm for $r \in \{0, 0.25, 0.5, 0.75, 1.0\}$ and $m = 20$.

Figure 8(b) shows the value $\texttt{Min}(Top)$ per iteration, while Figure 8(c) shows the number of active points per iteration of the five executions considered. It is clear that the value $\texttt{Min}(Top)$ approaches $lb(S)$ as quickly as r approaches the value 1. Nevertheless, we obtained the best results, in terms of number of iterations and, hence, of solving set size $|S|$, for values of r belonging approximatively to the range $[0.5, 0.75]$.

Table 2 shows the solving set size $|S|$, the robust solving set size $|R|$ computed using ROBUSTSOLVINGSET, and the minimal robust solving set size $|M|$ computed using MINIMALROBUSTSOLVINGSET.

Figures 9(a), 9(b) and 9(c) depict the solving set points (the grey) for $r = 0$, $r = 0.5$, $r = 1.0$, respectively. The dark points represent the points added to the solving set to make it robust. We note that these points are distributed along the boundaries of the data set region. From these figures we can observe that, for $r = 0.5$

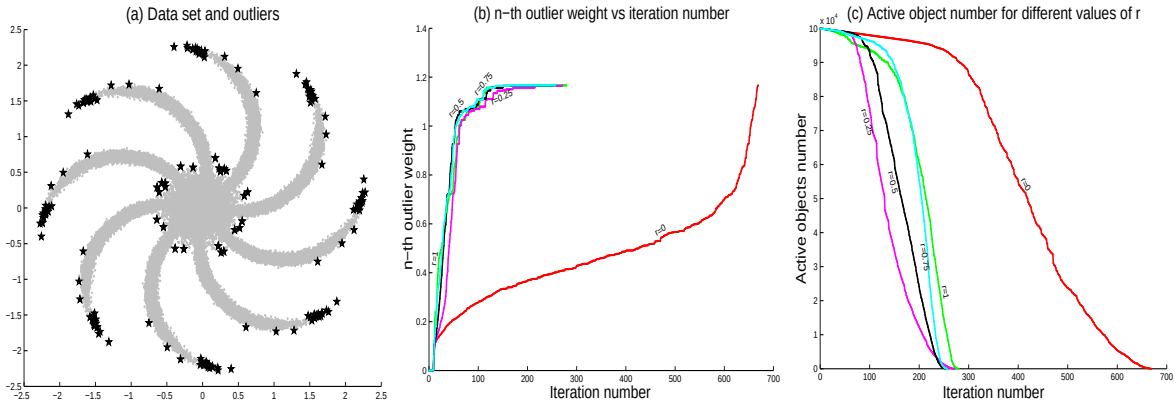

Fig. 8. The example data set.

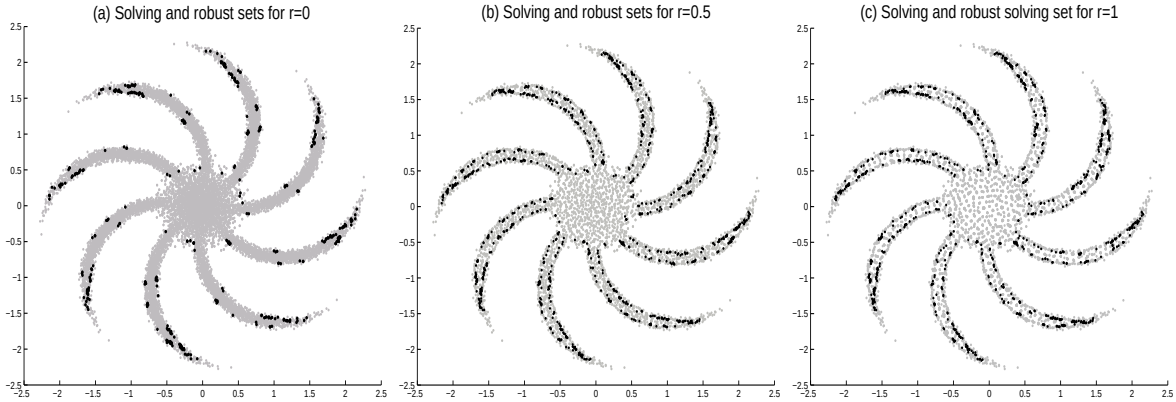

Fig. 9. Solving sets (gray points) and robust solving sets (both gray and dark points) found for the example data set.

the solving set points appear to be near uniformly distributed on the data set shape, for $r = 0$ their density appears to be proportional to the density of the data set points in the same regions, while for $r = 1.0$ they appear to form a lot of small clusters having size related to the value of m, separated by empty regions having diameter related to the value $lb(S)$.

To test the quality of the approximation obtained using a solving set instead of the entire data set, we generated $20{,}000$ random queries in the square $[-2.5, 2.5]^2$. We compared the robust solving set R ($|R| = 6{,}091$, about the 6.1% of the data set) computed for $r = 0.5$, with a set T of $|R|$ points extracted at random by D. In particular, in Figure 10(a), black points are true negatives (both D-inliers and R-inliers), light-gray points are true positives (both D-outliers and R-outliers), while the gray points are false positives (D-inliers but R-outliers). It is clear from the figure that the false positives accumulate along the boundaries of the accepting region, thus they represent points belonging to the region of transition between the inside and the outside of the data set distribution. In Figure 10(b), black points are both D-inliers and T-inliers, light-gray points are both D-outliers and T-outliers, while gray points are D-inliers but T-outliers. This figure shows the superiority of the approach based on the computation of a solving set w.r.t. using as reference set a random sample of the data set. Finally, Figure 10(c) shows the weight of these queries computed w.r.t. the data set D (black curve), the robust solving set R (gray curve), and the random set T (light gray curve). The closeness of the first two curves points out that the weight of the queries computed w.r.t. R is almost the same of that computed w.r.t. D. The curve relative to the minimal robust solving set is very near to that of the robust solving set, so we do not report this curve for clarity.

In the two dimensional data set example, we considered a query set containing a number of outliers much greater than the number of inliers in order to uniformly fill the square $[-2.5, 2.5]^2$. In real situations outliers are only a small percentage of the query set. Thus, in this example it is not meaningful to consider the false positive rate. However, we note that the false positive obtained using the robust solving set R are 590, i.e. the 2.95% of the queries, the false positive obtained using the minimal robust solving set M (for $r = 0.5$, $|M| = 5{,}505$, about the 5.5% of the data set) are 655, i.e. the 3.27% of the queries, while the false positive obtained using the random set T are $2{,}011$, i.e. the 10.6% of the queries.

Furthermore, the value $w_k(q, S)$ approximates the value $w_k(q, D)$ much better than $w_k(q, T)$. In particular,

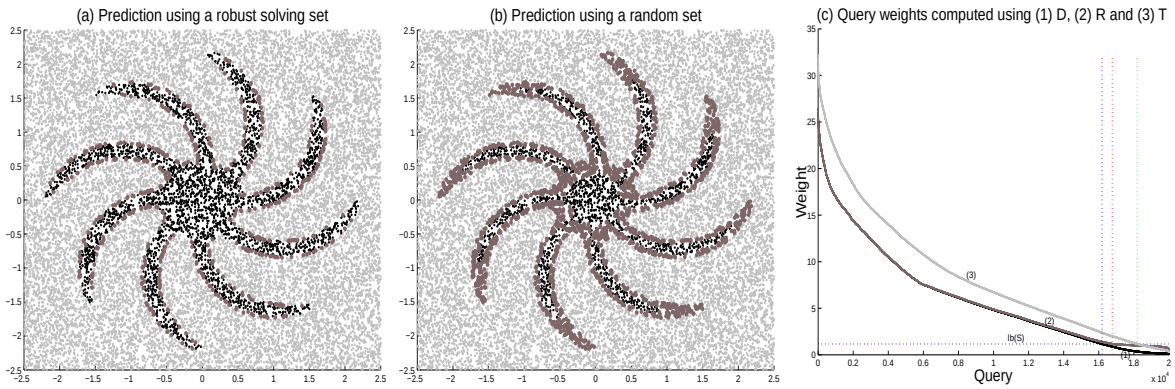

Fig. 10. Outlier prediction on the example data set.

$w_k(q, S)$ is very close to $w_k(q, D)$ when q is an outlier, while the difference between these two values becomes notable when q is an inlier. We note that this behavior is exactly the expected one. Indeed, the data set reduction operated computing a solving set is obtained at the expense of a poor approximation of the weight of the objects belonging to the data set that are not outliers. Obviously, this is not a problem, as we want to use the solving set to detect outliers. Moreover, the random set T behaves better than R on the inliers, as most of its points come from the highly populated regions of D, but its weight approximation is very poor on the outliers and on the inliers coming from the low density populated regions of D.

7.2 Solving set size

For this and the following experiments we considered five data sets. The $Gaussian$ data set G is a collection of $1{,}000{,}000$ points on the plane generated from a normal distribution having standard deviation 1. The $ColorHistogram$ CH, *CoocTexture* CT, and *ColorMoments* CM data sets are composed by $68{,}040$ points of $\mathbb{R}^{32}$, $\mathbb{R}^{16}$, $\mathbb{R}^{9}$ respectively, and represent image features extracted from a Corel image collection[1]. The $Landsat$ data set L is a collection of $275{,}465$ points of $\mathbb{R}^{60}$ representing normalized feature vectors associated to tiles of a collection of large aerial photos[2]. Figures 11(a)-(e) show the sizes $|S|$ of the solving sets (dashed curves) and the sizes $|R|$ of the robust solving sets (solid curves) obtained considering the five data sets for various values of n and k ($m = k$ and $r = 0.5$ in all the experiments) and several values $|D|$ of the data set size, using the Euclidean distance. We are going to discuss next the size of the minimal robust solving set. To vary the data set size, we considered in each experiment the data set obtained by picking distinct objects at random from the original data set. It is clear from these figures that the ratios $\frac{|S|}{|D|}$ and $\frac{|R|}{|D|}$ decrease dramatically with increasing values of $|D|$.

1. See `http://kdd.ics.uci.edu/databases/CorelFeatures/CorelFeatures.html` for more information.

2. See `http://vision.ece.ucsb.edu` for a detailed description.

Furthermore, we note that in these experiments the size $|R|$ of the robust solving set is less than two times the size $|S|$ of the associated solving set, even when $k = 100$, i.e. much less than the worst case $k|S|$ previously stated.

Figure 12 depicts the execution times of the algorithms SOLVINGSET and ROBUSTSOLVINGSET on the data of Figure 11. The figures show that the time trend of the algorithms roughly follows the product $|S| \cdot |D|$, as expected according to the analysis done in the previous section. Furthermore, we note that the time required to compute R from S is negligible.

Both SOLVINGSET and ROBUSTSOLVINGSET algorithms return a (robust) solving set nondeterministically chosen among all the possible (robust) solving sets. But, a data set has an exponential number of (robust) solving sets, thus it is interesting to study how the size of the (robust) solving set computed varies among different executions of the algorithm on the same values for the input parameters.

Figure 13 on the left shows minimum and maximum sizes of the solving set ($|S|_{min}$ and $|S|_{max}$), of the robust solving set ($|R|_{min}$ and $|R|_{max}$), and of the minimal robust solving set ($|M|_{min}$ and $|M|_{max}$), obtained on random samples of the *ColorHistogram* data set, with parameters $n = 100$, $k = 25$, $m = 25$, $r = 0.5$. Each execution is repeated ten times, thus minimum and maximum values are taken over the ten executions. We note that the sizes of the computed solving sets are within the $\pm 1\%$ of the mean size. These results suggest that the proposed algorithms guarantee to compute a solving set whose size is stable. As for the size of the minimal robust solving sets we note that it is reasonably smaller than that of the robust solving set, though greater than the solving set. We will compare the classification accuracy of these sets in the following.

Figure 13 on the right, reports the mean execution times of the algorithms SOLVINGSET, ROBUSTSOLVINGSET, and MINIMALROBUSTSOLVINGSET on the experiments on the left. The time required to compute the minimal robust solving set, as expected, can be sensibly higher than the time required to compute a robust solving set. Nevertheless, we can note that the

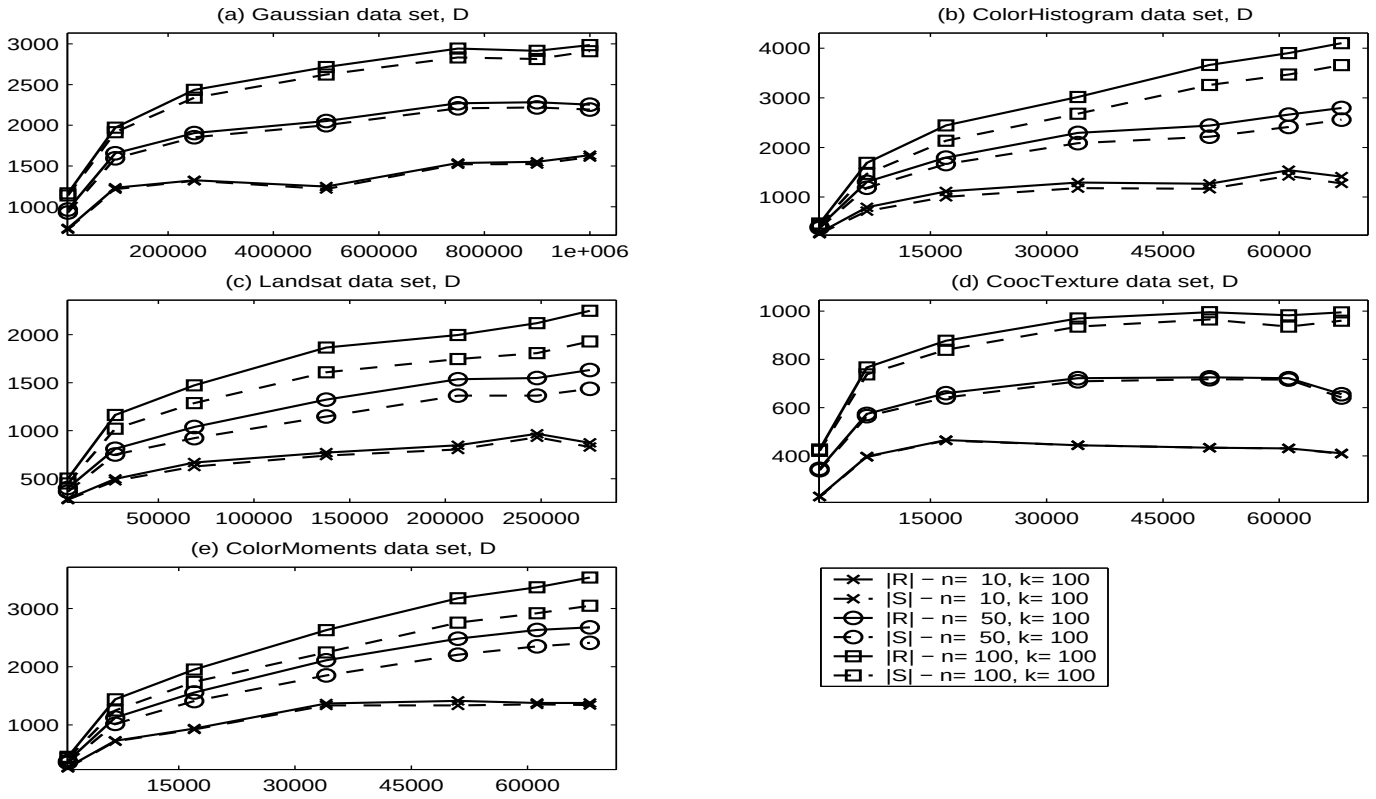

Fig. 11. Scaling analysis of the ODP solving sets sizes. The x-axis represents the size of the data set, while the y-axis the size of the (robust) solving set.

actual execution time of MINIMALROBUSTSOLVINGSET is clearly better than the worst case stated in the previous section. Indeed, if T_S is the time needed to compute the solving set, the time needed to compute the minimal robust solving set is roughly $|R| \cdot T_S$, that is for the data set considered about three orders of magnitude greater than T_S, while the measured execution time is much less than the worst case, though significantly greater than T_S.

7.3 False positive rate

To test the quality of the outliers detected using the solving set, for each of the five data sets described above, we considered as data set (D in the following) a random sample composed by the 90% of the original data set and then we used the remaining 10% as query set (Q in the following). We computed a solving set S, a robust solving set R, and a minimal robust solving set M of each data set, for $n \in \{10, 50, 100\}$, $k \in \{10, 50, 100\}$, $m = k$, $r = 0.5$, and dist the Euclidean distance. Table 3 reports the cardinality $|S|$ of the solving set S, the cardinality $|R|$ of the robust solving set R, the cardinality $|M|$ of the minimal robust solving set M, and the false positive rates obtained using S, R, and M respectively, to answer the $OPP\langle D, q, \mathrm{dist}, n, k\rangle$ on the objects q of Q. As the table shows, the size of the robust solving set is slightly higher than that of the solving set. Clearly, the minimal robust solving set is always smaller than the robust solving set, while it seems there is no relation between the size of the minimal robust solving set and that of the solving set. It is worth to note that the difference between the sizes of the minimal robust and robust solving sets is, in some cases, very small.

The differences of the false positive rates obtained for all the three cases are negligible, though the robust solving set gives better results. The table points out that the minimal robust solving set is smaller than the robust one, but the latter guarantees a better accuracy. By this observation we can conclude that the choice of computing the minimal robust solving set is reasonable only if the space requirements are more important than the time requirements, considering that the robust solving set provides a good compromise between accuracy and size. Finally we can observe that the table shows that the approach ensures a very good quality of the outliers obtained by using one of the three types of solving sets. In fact, for all the data sets considered, the false positive rate is very low and in one case, the *CoocTexture* data set when $n = 10$, it is always zero.

7.4 Evaluation through ROC Analysis

The outlier detection technique described in this work is unsupervised as it can be applied to training data that has no labels associated. Nevertheless, as the solving set represents a model of the overall data set, it makes

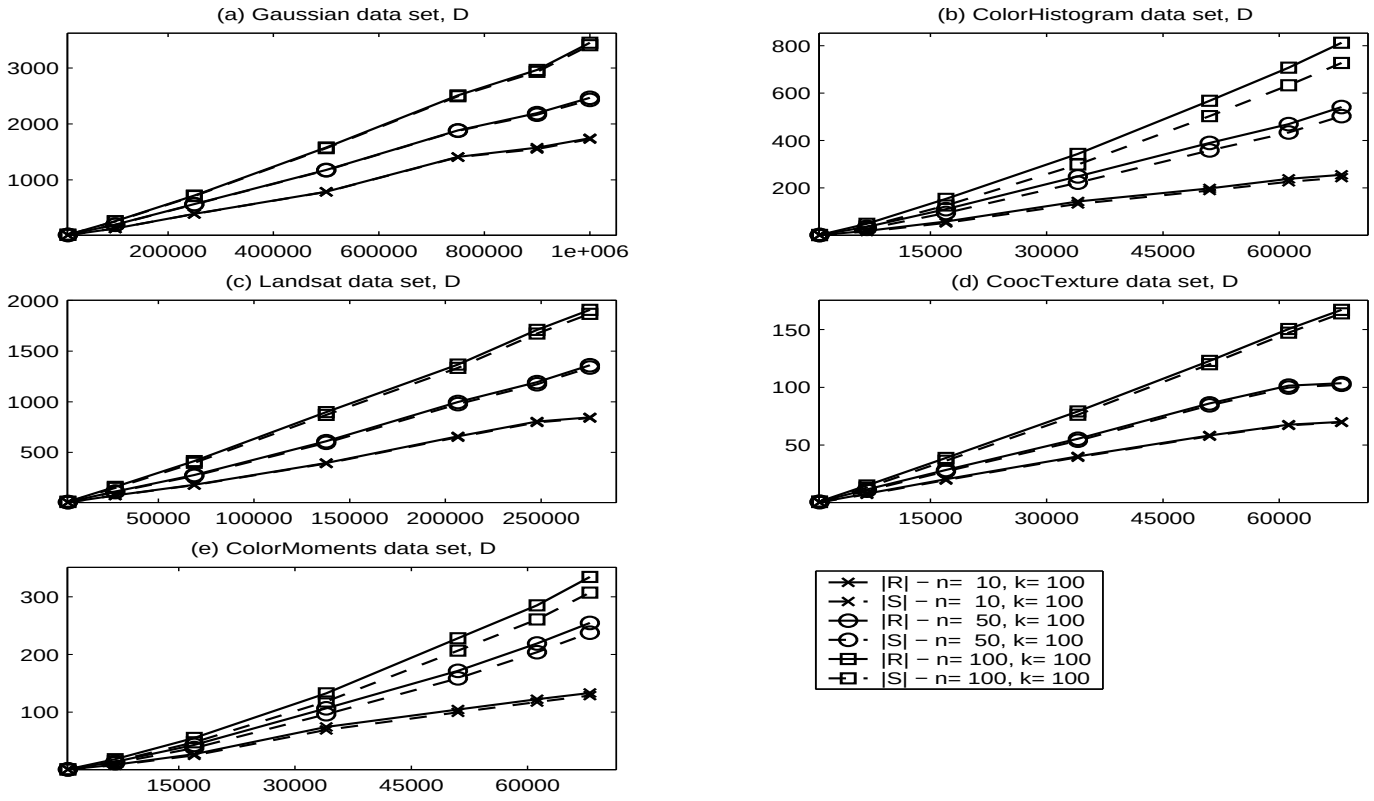

Fig. 12. Scaling analysis of the algorithms SOLVINGSET and ROBUSTSOLVINGSET. The x-axis represents the size of the data set, while the y-axis the execution time in seconds.

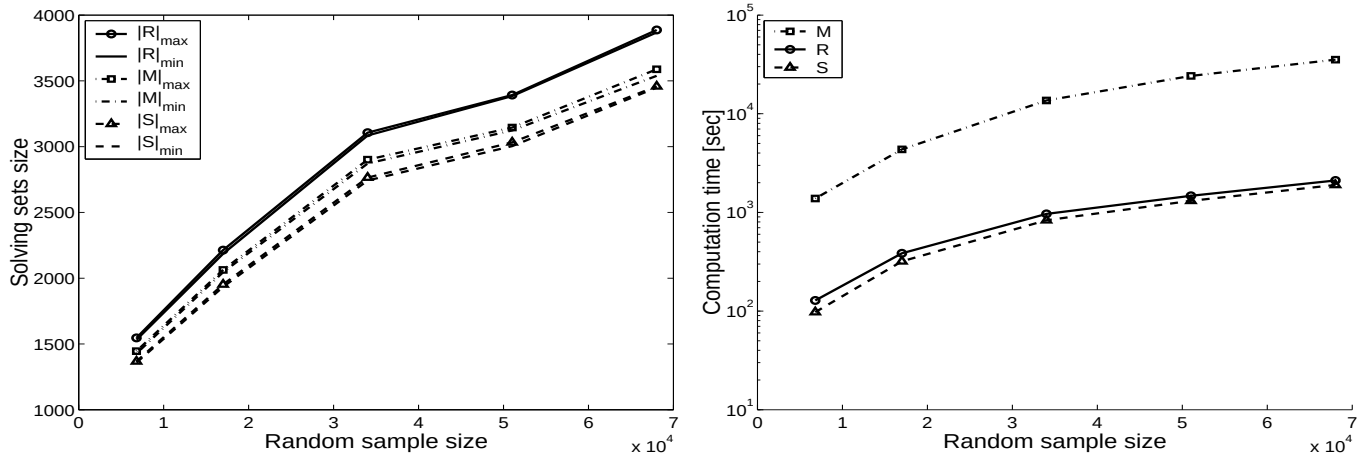

Fig. 13. On the left: minimum and maximum sizes of the solving set ($|S|_{min}$ and $|S|_{max}$), of the robust solving set ($|R|_{min}$ and $|R|_{max}$), and of the minimal robust solving set ($|M|_{min}$ and $|M|_{max}$). On the right: time required to compute the solving sets on the left.

	n	k	$\|S\|$	$f.p.$ [%]	$\|R\|$	$f.p.$ [%]	$\|M\|$	$f.p.$ [%]
	10	10	510	0.00	521	0.00	412	0.00
	10	50	963	0.00	977	0.00	710	0.00
	10	100	1525	0.01	1550	0.00	1172	0.01
	50	10	1160	0.01	1177	0.00	995	0.01
G	50	50	1572	0.01	1609	0.00	1321	0.01
	50	100	2208	0.01	2267	0.00	1851	0.01
	100	10	1970	0.01	1992	0.01	1692	0.01
	100	50	2159	0.01	2232	0.01	1911	0.01
	100	100	2852	0.02	2951	0.01	2519	0.01
	10	10	1657	1.09	1732	0.94	1548	1.22
	10	50	1234	0.24	1313	0.16	1207	0.29
	10	100	1424	0.22	1540	0.18	1437	0.21
	50	10	3009	2.23	3167	1.75	2805	2.43
CH	50	50	2437	1.06	2699	0.65	2529	0.75
	50	100	2414	1.09	2661	0.54	2601	0.56
	100	10	3972	3.04	4218	2.34	3717	3.35
	100	50	3317	2.25	3783	1.09	3606	1.25
	100	100	3471	1.87	3902	0.93	3796	1.00
	10	10	769	0.08	791	0.07	670	0.13
	10	50	658	0.03	697	0.02	610	0.02
	10	100	832	0.03	882	0.02	764	0.02
	50	10	1599	0.30	1677	0.24	1453	0.35
L	50	50	1206	0.10	1368	0.05	1259	0.07
	50	100	1425	0.10	1603	0.05	1511	0.05
	100	10	2200	0.45	2314	0.38	1991	0.45
	100	50	1633	0.27	1883	0.12	1780	0.13
	100	100	1822	0.25	2118	0.10	2054	0.11
	10	10	1142	0.47	1200	0.40	1081	0.60
	10	50	1107	0.35	1215	0.16	1150	0.24
	10	100	1522	0.06	1540	0.06	1229	0.09
	50	10	2143	1.13	2252	0.97	2011	1.35
CM	50	50	1951	0.88	2208	0.51	2121	0.59
	50	100	2326	0.76	2623	0.41	2595	0.41
	100	10	3012	1.75	3157	1.42	2838	1.99
	100	50	2598	1.55	2986	0.78	2890	0.85
	100	100	2935	1.40	3372	0.75	3355	0.75
	10	10	95	0.01	96	0.01	68	0.04
	10	50	228	0	230	0	129	0
	10	100	428	0	428	0	200	0
	50	10	305	0.07	322	0.04	250	0.09
CT	50	50	468	0	478	0	373	0
	50	100	721	0.01	729	0.01	524	0.01
	100	10	663	0.38	692	0.29	585	0.43
	100	50	801	0.29	844	0.24	768	0.25
	100	100	988	0.22	1020	0.21	920	0.24

TABLE 3
Solving set sizes and false positives rates ($f.p.\%$).

sense to compare the accuracy of the solving set with the accuracy of the data set in separating data from a certain class C – assumed to represent normal behavior – from data from other classes – assumed to represent abnormality. Thus, in this section we deal with labelled data set. We note that, in this novel context, true/false negative/positive objects must be defined by comparing the labels predicted w.r.t. S (or D) with the labels L associated with the objects, as shown in Table 4.

Accordingly to the new definitions, a false positive rate and a detection rate value can be associated with the dataset D, while the detection rate of the solving set has, in general, a value in the interval [0,1].

		Predicted label w.r.t. S (or D)	
		S(or D)-inlier	S(or D)-outlier
Class label L	$L = C$	*true negative*	*false positive*
	$L \neq C$	*false negative*	*true positive*

TABLE 4
Classification of the query objects.

We computed the robust solving set for four real data sets and then compared the false positive rate for normal query objects and the detection rate for outlier query objects when using the overall data set against using the robust solving set to determine the weight of each query. We considered four labelled real data sets well known in the literature: *Wisconsin Diagnostic Breast Cancer* [21], *Shuttle* [22], *DARPA 1998*, and the *KDD CUP 99* data sets. The first two data sets are used for classification tasks, thus we considered the examples of one of the classes as the normal data and the others as the exceptional data. The last two come from the *DARPA 1998 Intrusion Detection Evaluation Data* [23] and have been extensively used to evaluate intrusion detection algorithms. Next we briefly describe the characteristics of these data sets:

- *Breast cancer:* The *Wisconsin Diagnostic Breast Cancer* data set is composed by instances representing features describing characteristics of the cell nuclei present in the digitalized image of a breast mass. Each instance has one of two possible classes: *benign*, that we assumed as the normal class, or *malignant*.
- *Shuttle:* The *Shuttle* data set was used in the European StatLog project which involves comparing the performances of machine learning, statistical, and neural network algorithms on data sets from real-world industrial areas [22]. This data set contains 9 attributes all of which are numerical. The data is partitioned in 7 classes, namely, *Rad Flow*, *Fpv Close*, *Fpv Open*, *High*, *Bypass*, *Bpv Close*, and *Bpv Open*. Approximately 80% of the data belong to the class *Rad Flow*, that we assumed as the normal class.
- *DARPA:* The DARPA 1998 Intrusion Detection Evaluation Data consists of network connection records of several intrusions simulated in a military network environment. The TCP connections have been elaborated to construct a data set of 23 features, one of which identifying the kind of attack: DoS, $R2L$, $U2R$, and $PROBING$. We used the TCP connections from 5 weeks of training data.
- *KDD Cup:* This data set comes from the 1998 DARPA Intrusion Detection Evaluation Data [23] but the records are described by 41 characteristics, of which 3 categorical, which we excluded.

Breast Cancer and *Shuttle* data sets are composed by a training set and a test set. We merged these two sets obtaining a unique labelled data set. From each labelled data set, we extracted three unlabelled sets: a set of *normal examples* and a set of *normal queries*, both

Data set name	Attributes	Normal examples	Normal queries	Exceptional queries
Shuttle	9	40,000	500	1,244
Breast cancer	9	400	44	239
DARPA	23	24,051	9,620	18,435
KDD Cup	38	97,277	3,029	3,235

TABLE 5
The data used in the experiments.

Data set name	k	ROC area data set	ROC area solving set
Shuttle (*High* class)	25	0.974	0.99
	100	0.959	0.964
Shuttle (other classes)	25	0.996	0.994
	100	0.996	0.995
Breast cancer	5	0.987	0.976
	10	0.986	0.975
DARPA	500	0.883	0.883
	1000	0.894	0.894
KDD Cup	10	0.939	0.941
	25	0.954	0.956

TABLE 6
The areas of the ROC curves of Figure 14 .

containing data from the normal class, and a set of *exceptional queries*, containing all the data from the other classes. Table 5 reports the sizes of these sets.

We then used the normal examples to find the robust solving set R for the $ODP\langle D, \mathrm{dist}, n, k\rangle$ and the normal and exceptional queries to determine the false positive rate and the detection rate when solving $OPP\langle D, q, \mathrm{dist}, n, k\rangle$ and $OPP\langle R, q, \mathrm{dist}, n, k\rangle$. We computed a robust solving set for values of the parameter n ranging from $0.01|D|$ to about $0.10|D|$ or $0.20|D|$ (i.e. from the 1% to about the 10% or 20% of the normal examples set size) and using $r = 0.75$ and $m = k$ in all the experiments and the Euclidean distance as metric dist.

The performance of the method was measured by computing the ROC (Receiver Operating Characteristic) curves [24] for both the overall data set of normal examples, denoted by D, and the robust solving set. The ROC curves show how the detection rate changes when specified false positive rate ranges from the 0% to the 100% of the normal examples set size. We have a ROC curve for each possible value of the parameter k. Let Q_I denote the normal queries associated with the normal example set D, and Q_O the associated exceptional queries. For a fixed value of k, the ROC curve is computed as follows. Each point of the curve for the sets D and R resp. is obtained as $(\frac{|I_O|}{|Q_I|}, \frac{|O_O|}{|Q_O|})$, where I_O is the set of queries q of Q_I that are outliers for the $OPP\langle D, q, \mathrm{dist}, n, k\rangle$ and $OPP\langle R, q, \mathrm{dist}, n, k\rangle$ resp., and O_O is the set of queries of Q_O that are outliers for the same problems.

Figure 14 reports the ROC curves together with the curves of the robust data set sizes, i.e. the curves composed by the points (false positive rate,$|R|$). For each set we show two curves, associated with two different values of k. Dashed lines are relative to the normal examples set (there called data set), while solid lines are relative to the robust solving set.

The ROC curve shows the tradeoff between the false positive rate and the detection rate. The closer the curve follows the left and the top border of the unit square, the more accurate the method. Indeed, the area under the curve is a measure of the accuracy of the method in separating the outliers from the inliers. Table 6 reports the approximated areas of all the curves of Figure 14 (in order to compute the area we interpolated the missing points exploiting the fact that the curve starts in $(0, 0)$ and stops in $(1, 1)$). It is worth to note that the best values obtained using the robust solving set vary from 0.894 to 0.995, i.e. they lye in the range of values characterizing excellent diagnostic tests.

We recall that, by definition, the size $|R|$ of the robust solving set R associated with a data set D is greater than n. Furthermore, in almost all the experiments $n/|D| \simeq$ false positive rate. Thus, the relative size of the robust solving set $|R|/|D|$ is greater than the false positive rate. We note that in the experiments reported, $|R|$ can be roughly expressed as αn, where α is a small constant whose value depends on the data set distribution and on the parameter k. The curves show that using the robust solving set to predict outliers, not only improves the response time of the prediction over the entire data set, since the size of the solving set is a fraction of the size of the data set, as figures 14 point out, but also guarantees the same or a better response quality than the overall data set.

Next we briefly discuss the experiments on each data set.

As regard the *Breast cancer* data set, we note that the robust solving set for $k = 5$ composed by about the 10% (18% resp.) of the data set, reports a false positive rate 4.5% (6.8% resp.) and detection rate 97.5% (100% resp.).

For the *Shuttle* data set, the robust solving set for $k = 25$ composed by about the 12% of the data set, guarantees a false positive rate 1% and a detection rate of the 100% on the classes *Fpv Close*, *Fpv Open*, *Bypass*, *Bpv Close*, and *Bpv Open*. Furthermore, the robust solving set sensibly improves the prediction quality over the data set for the class *High*.

These two experiments point out that the method can behave very well also for binary classification problems when one of the two classes is assumed normal and the other abnormal, and for the detection of rare values of the attribute class in a completely unsupervised manner, which differs from the supervised approaches like [25] that searches for rare values in a set of labelled data.

As regard the *DARPA* data set, a 90% of detection rate is obtained by allowing the 10% of false positive. In this case we note that the size of the solving set is not small since higher values of the parameter k are needed to

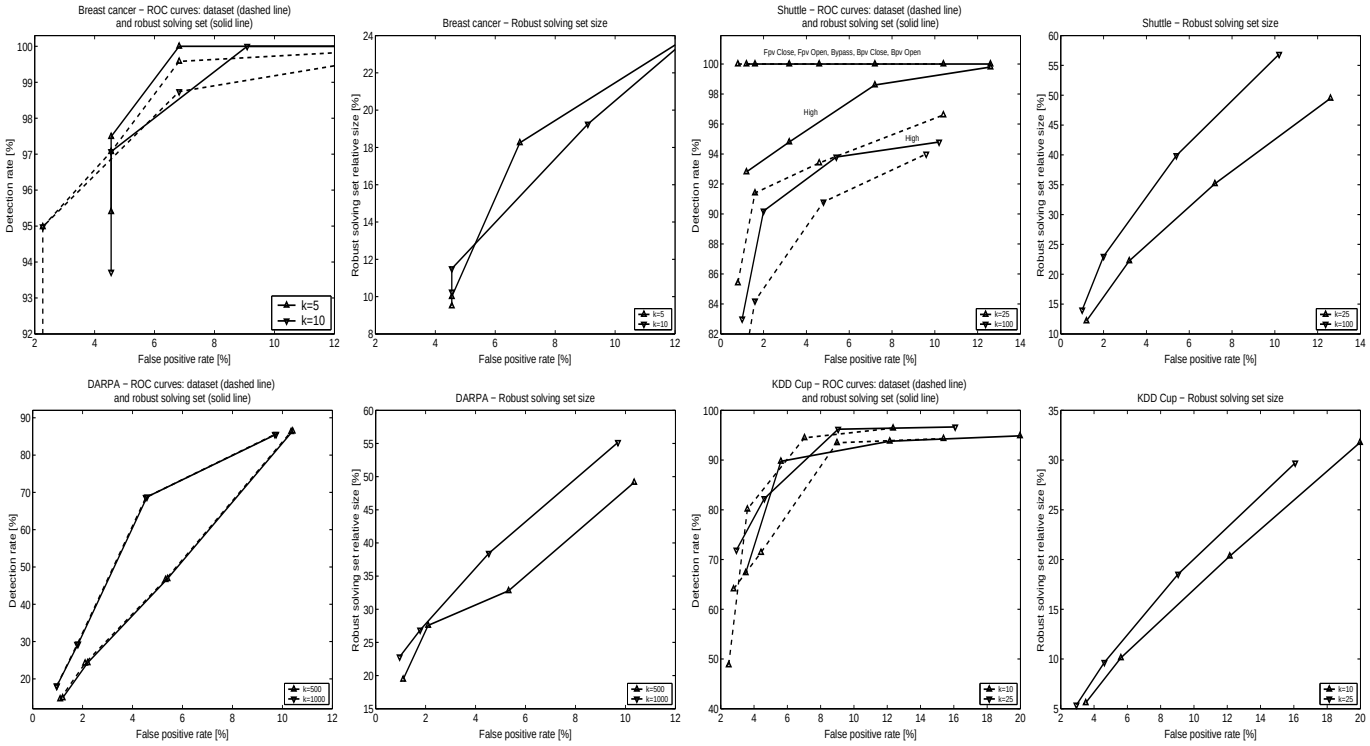

Fig. 14. ROC curves and robust solving set sizes.

obtain good prediction results because of the peculiarity of this data set in which inliers and outliers overlap. In particular, some relatively big clusters of inliers are close to regions of the feature space containing outliers. As a consequence a value of $k \geq 500$ is needed to "erase" the contribution of these clusters to the weight of an outlier query object and, consequently, to improve the detection rate.

Finally, for the *KDD Cup* data set, when $k = 25$, a detection rate of 90% can be obtained if we allow a 6% of false positives, while a detection rate of 95% can be obtained if we allow a 9% of false positives. We notice that, if we use the overall data set the detection rate is lower, while, as the figure shows, the size of the robust solving set is below the 10% of the overall data set in the former case, and below the 20% in the latter. This result strengthens the worthiness of the approach proposed.

7.5 Discussion

In this section we draw some considerations concerning the applicability of the method here presented.

It is known that the perception of what is an outlier is rather subjective. The task of detection is to find those objects that seem to behave differently from the others, the human analyst with expertise in the application domain has then the task to decide if the obtained objects can be effectively considered as outliers. For example, the dark stars in Figure 2 are those objects that most deviate (in this case more distant) from the others. However, in the detection context it is important to tune the trade-off between the detection rate and the false positive rate. For example, the increase of the number of normal objects to be considered outliers, allows to have a higher detection rate in the prediction phase.

As regards the choice of the parameters n and k, when the labels of the data set are known, they can be set by tuning the trade-off between false positive rate and detection rate. This tuning can be obtained with a trial-and-error process. Different values of n and k are fixed and the trade-off between false positive rate and detection rate is studied: a higher false positive rate is accepted if the detection rate increases sensibly. However, in general the labels are not known, but we observed that, when the believed overlapping degree between the normal and abnormal classes is low, then a small value of k is sufficient. This behavior is noticed, for example, in the experiments reported in Figure 14, where for the DARPA data set high values of k are necessary because normal and abnormal data are not separated. As regards the choice of n, it is more related to the expectation on the percentage of abnormal objects present in the data to be processed.

As for the applicability of the method in finding outliers, we experimented that, if the distributions of the normal and abnormal data are such that the two classes either do not overlap or have a low degree of overlapping in the feature space adopted, the method behaves well. When the method is not able to discriminate, we can assume that either the feature space or the distance notion used are not suitable for that application domain.

Finally, although the prediction phase can obviously take advantage of optimized techniques to search for

the k nearest neighbors [26], testing such techniques in our methods is not the primary aim of our proposal. As regards the prediction time, it depends on the dimension of the solving set and on the algorithm used to find the k nearest neighbors (i.e. exhaustive search, k-d-tree, pivots, etc.). However, both the temporal and spatial complexities are directly proportionate to the size of the input data set on which the search must be done. Thus the substitution of the whole data set with a subset of it is in any case advantageous from the point of view of computational resources to be employed.

8 Conclusions

This work enhances the state of distance-based outlier detection research, by giving the first proposal of distance-based outlier prediction method. The method introduced is based on the notion of outlier detection solving set, a subset of the data set, that can be used to predict if new unseen objects are outliers. The scalings analysis of the solving set size points out that it is composed by a fraction of the overall data set, the false positive rate is shown to be negligible, and ROC analysis of the method demonstrates that using the solving set instead of the data set guarantees a comparable accuracy, but at a lower computational cost.

References

[1] J. Han and M. Kamber, *Data Mining, Concepts and Technique*. Morgan Kaufmann, San Francisco, 2001.

[2] T. Fawcett and F. Provost, "Adaptive fraud detection," *Data Mining and Knowledge Discovery*, vol. 1, pp. 291–316, 1997.

[3] W. Lee, S. Stolfo, and K. Mok, "Mining audit data to build intrusion detection models," in *Proc. Int. Conf on Knowledge Discovery and Data Mining (KDD-98)*, 1998, pp. 66–72.

[4] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo, "A geometric framework for unsupervised anomaly detection : Detecting intrusions in unlabeled data," in *Applications of Data Mining in Computer Security, Kluwer*, 2002.

[5] E. Knorr and R. Ng, "Algorithms for mining distance-based outliers in large datasets," in *Proc. Int. Conf. on Very Large Databases (VLDB98)*, 1998, pp. 392–403.

[6] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," in *Proc. Int. Conf. on Managment of Data (SIGMOD'00)*, 2000, pp. 427–438.

[7] F. Angiulli and C. Pizzuti, "Outlier mining in large high-dimensional data sets," *IEEE Transaction on Knowledge and Data Engineering*, vol. 2, no. 17, pp. 203–215, February 2005.

[8] S. D. Bay and M. Schwabacher, "Mining distance-based outliers in near linear time with randomization and a simple pruning rule," in *Proc. Int. Conf. on Knowledge Discovery and Data Mining (KDD'03)*, 2003.

[9] S. Floyd and M. Warmuth, "Sample compression, learnability, and the vapnik-chervonenkis dimension," *Machine Learning*, vol. 211, no. 3, pp. 269–304, 1995.

[10] B. Schölkopf, C. Burges, and V. Vapnik, "Extracting support data for a given task," in *Proc of the 1st Int. Conf. on Knowledge Discovery & Data Mining*, 1995, pp. 252–257.

[11] V. Barnett and T. Lewis, *Outliers in Statistical Data*. John Wiley & Sons, 1994.

[12] A. Arning, C. Aggarwal, and P. Raghavan, "A linear method for deviation detection in large databases," in *Proc. Int. Conf. on Knowledge Discovery and Data Mining (KDD'96)*, 1996, pp. 164–169.

[13] M. M. Breunig, H. Kriegel, R. Ng, and J. Sander, "Lof: Identifying density-based local outliers," in *Proc. Int. Conf. on Managment of Data (SIGMOD'00)*, 2000.

[14] W. Jin, A. Tung, and J. Han, "Mining top-n local outliers in large databases," in *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'01)*, 2001.

[15] K. Yamanishi and J. Takeuchi, "Discovering outlier filtering rules from unlabeled data," in *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'01)*, 2001, pp. 389–394.

[16] E. Knorr, R. Ng, and V. Tucakov, "Distance-based outlier: algorithms and applications," *VLDB Journal*, vol. 8, no. 3-4, pp. 237–253, 2000.

[17] E. Knorr and R. Ng, "Finding intensional knowledge of distance-based outliers," in *Proc. Int. Conf. on Very Large Databases (VLDB99)*, 1999, pp. 211–222.

[18] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava, "A comparative study of anomaly detection schemes in network intrusion detection," in *Proc. SIAM Int. Conf. on Data Mining (SIAM-03)*, 2003.

[19] M. R. Garey and D. S. Johnson, *Computer and Intractability*. New York: W. H. Freeman and Company, 1979.

[20] D. Peleg, G. Schechtman, and A. Wool, "Randomized approximation of bounded multicovering problems," *Algorithmica*, vol. 18, no. 1, pp. 44–66, 1997.

[21] L. Mangasarian and W. H. Wolberg, "Cancer diagnosis via linear programming," *SIAM News*, vol. 25(5), pp. 1–18, 1990.

[22] C. Feng, A. Sutherland, S. King, S. Muggleton, and R. Henery, "Comparison of machine learning classifiers to statistics and neural networks," in *AI & Stats Conf. 93*, 1993.

[23] D. A. R. P. A. DARPA, "Intrusion detection evaluation," in *http://www.ll.mit.edu/IST/ideval/index.html*.

[24] F. Provost, T. Fawcett, and R. Kohavi, "The case against accuracy estimation for comparing induction algorithms," in *Proc. Int. Conf. on Machine Learning (ICML'98)*, 1998.

[25] L. Torgo and R. Ribeiro, "Predicting outliers," in *Proc. Int. Conf. on Principles of Data Mining and Knowledge Discovery (PKDD'03)*, 2003, pp. –.

[26] V. Gaede and O. Günther, "Multidimensional access methods," *ACM Comput. Surv.*, vol. 30, no. 2, pp. 170–231, 1998.