COPYRIGHT NOTICE

This is the author's version of the work. The definitive version was published in *IEEE Transactions on Pattern Analysis and Machine Intelligence* (TPAMI), 2011.

DOI: `http://dx.doi.org/10.1109/TPAMI.2011.204`.

Prototype-based Domain Description for One-Class Classification

Fabrizio Angiulli

Abstract—This work introduces the Prototype-based Domain Description rule (PDD) one-class classifier. PDD is a nearest neighbor based classifier, since it accepts objects on the basis of their nearest neighbor distances in a reference set of objects, also called prototypes. For a suitable choice of the prototype set, the PDD classifier is equivalent to an other nearest neighbor based one-class classifier, namely the NNDD classifier. Moreover, it generalizes statistical tests for outlier detection. The concept of PDD consistent subset is introduced, which exploits only a selected subset of the training set. It is shown that computing a minimum size PDD consistent subset is in general not approximable within any constant factor. A logarithmic approximation factor algorithm, called the CPDD algorithm, for computing a minimum size PDD consistent subset is then introduced. In order to efficiently manage very large data sets, a variant of the basic rule, called Fast CPDD, is also presented. Experimental results show that the CPDD rule sensibly improves over the CNNDD classifier, namely the condensed variant of NNDD, in terms of size of the subset while guaranteeing a comparable classification quality, that it is competitive over other one-class classification methods and is suitable to classify large data sets.

Index Terms—One-class classification, novelty detection, nearest neighbor classification, data set condensation.

1 Introduction

When only data coming from a single class is available and the goal is to discriminate the objects belonging to that class from the other ones, data domain description classification techniques, also called one-class classifiers, are needed. Specifically, the task dealt with in one-class classification is the following: given a data set of objects, also called training set or reference set, belonging to a certain object space, determine a description of the data, that is a rule partitioning the object space into an accepting region (containing the objects belonging to the class represented by the training set) and a rejecting region, (containing all the other objects). One-class classification is also related to outlier or novelty detection, as the description of the data can be used to detect the objects deviating significantly from the training data. Indeed, domain description can be also regarded as a semi-supervised technique for outlier detection, in that it takes in input examples of normality in order to build its model, while no examples of abnormality are taken into account.

Several approaches to one-class classification have been presented in the literature [2], and some of them are briefly recalled next. One-class classification techniques based on *Support Vector Machines* (SVM) extend the SVM algorithm to the case of unlabelled data [3], [4], [5], [6]. The one-class SVM algorithm is a specialization, working in the presence of only positive data, of the standard two-class SVM algorithm, which, conversely, requires both positive and negative examples. Basically, the feature space is transformed via a kernel and then the origin of the transformed space is treated as the only member of the negative class. Thereafter, the standard two-class SVM algorithm is employed. The one-class SVM exploits the parameter $\nu \in (0, 1]$ in order to control the trade-off between the number of training-set examples accepted and the size of the support vector regularization term. The *k-center* method covers the data set with k balls with equal radii [7]. Ball centers are placed on training objects such that the maximum of all minimum distances between training objects and the centers is minimized. The nearest neighbor one-class classification method *NN-d* [8] accepts a test object p provided that the distance to its nearest neighbor q in the training set is less or equal than the distance from q to the nearest neighbor of q in the training set. This measure is comparable with the Local Outlier Factor [9] used to detect outliers.

Recently the Nearest Neighbor Domain Description rule (NNDD, for short) has been introduced [10], which is a one-class classifier accepting test objects whose nearest neighbor distances in a reference data set, assumed to model normal behavior, lie within a certain threshold. In particular, given a reference set and two parameters k and θ, the NNDD rule associates a feature vector $\delta(x) \in \mathbb{R}^k$ with each object x. The vector $\delta(x)$ is composed of the distances separating x from its first k nearest neighbors in the reference set. The classifier then accepts x if and only if $\delta(x)$ belongs to the hyper-sphere (according to one of the L_r Minkowski metrics, $r \in \{1, 2, \ldots, \infty\}$) centered in the origin of $\mathbb{R}^k$ and having radius θ, that is if and only if $\|\delta(x)\|_r \leq \theta$. The NNDD has some interesting properties, since its decision function encompasses different definitions of *distance-based outlier* [11], which, in their turn, under the infinity Minkowski metrics have been proved to generalize several *statistical tests* for outlier detection [12].

In this work a novel nearest neighbor based one-class classifier, called the *Prototype-based Nearest Neighbor* classifier (PDD, for short), is introduced. A *prototype set* is a set of objects x_i, also called *prototypes*, each of which is associated with a *radius* $R(x_i)$: Given parameter θ, a generic object y

F. Angiulli is with DEIS, University of Calabria, Italy, email: f.angiulli@deis.unical.it

A preliminary version of this article appears in the Proceedings of the 18th European Conference on Artificial Intelligence (ECAI'08) [1].

of the space is accepted by the PDD rule provided that y lies within distance $\theta - R(x_i)$ from some prototype x_i, or, equivalently, if $\mathrm{d}(x_i, y) + R(x_i) \leq \theta$. The contributions of the work are summarized in the following:

- The PDD one-class classifier is introduced. It is shown that the PDD classifier is in some sense equivalent to the NNDD one under the infinity Minkowski metric and also related to statistical definitions for outlier detection, for a suitable choice of the prototype set;
- The concept of PDD consistent subset is introduced, that is a subset of the original prototype set which is, loosely speaking, equivalent to the original prototype set. It is shown that computing a minimum size PDD consistent subset is a difficult task, since in general the problem is not approximable within any constant factor;
- A logarithmic approximation factor algorithm, called CPDD, for computing a minimum size PDD consistent subset is then introduced. The CPDD algorithm has some parameters which allows to tune the trade off between accuracy and size of the model. Also, the Fast CPDD algorithm is provided, which is designed to efficiently compute a PDD consistent subset when very large data sets are taken into account;
- Experimental results show that the CPDD rule sensibly improves over the CNNDD classifier in terms of size of the subset while guaranteeing a comparable classification quality, that the CPDD is competitive over other one-class classification techniques, and that Fast CPDD computes a consistent subset of comparable size and accuracy, but at a reduced computational cost, and is effective in classifying large data sets.

The rest of the work is organized as follows. Section 2 defines the Prototype-based Domain Description rule (PDD) and the concept of PDD consistent subset, and points out relationships with the CNNDD rule and with statistical definitions for outliers. Section 3 investigates the computational complexity of the problem of computing a minimum size PDD consistent subset. Section 4 introduces the CPDD rule and its properties. Subsequent Section 5 describes the Fast CPDD algorithm, analyzing its expected behavior on some distributions. Section 6 presents experimental results, including the sensitivity analysis of CPDD, comparison with CNNDD and other one-class classifiers, and performances of Fast CPDD. Finally, Section 7 presents conclusions of the work.

2 THE PROTOTYPE-BASED DOMAIN DESCRIPTION

This section introduces the Prototype-based Domain Description rule and states its main properties.

In the following, $\mathcal{U}$ denotes a set of objects, d a distance metrics on $\mathcal{U}$, D a set of objects from $\mathcal{U}$, k a positive integer number, θ a positive real number, and $r \in \{1, 2, \ldots, \infty\}$ a Minkowski metric L_r.

A *prototype set* P is a set of pairs

$$P = \{\langle x_1, r_1\rangle, \langle x_2, r_2\rangle, \ldots, \langle x_n, r_n\rangle\},$$

where each x_i $(1 \leq i \leq n)$ is an object of U, also called *prototype*, and each r_i is a real number, also called *prototype radius*. Given a prototype x_i, the prototype radius r_i associated with x_i is also denoted by $R(x_i)$.

Intuitively, a prototype set defines a set of hyper-spherical regions, each of which is centered in x_i and has radius r_i. Next the *Prototype-based Domain Description* one-class classifier is defined.

Definition 2.1: Given a prototype set P, the *Prototype-based Domain Description rule* according to P, d, and θ, is the function $\mathrm{PDD}_{P,\mathrm{d},\theta}$ (PDD, for short) from $\mathcal{U}$ to $\{-1, +1\}$ such that

$$\mathrm{PDD}(y) = \begin{cases} +1, & \exists x \in P : \mathrm{d}(x, y) + R(x) \leq \theta \\ -1, & \text{otherwise} \end{cases}$$

The PDD rule *accepts* the input object y (that is, it returns the value $+1$) if y lies within distance $\theta - R(x_i)$ from some prototype x_i. Otherwise, the PDD rule *rejects* the input object y (that is, it returns the value -1). Thus, the PDD rule accepts an object y if and only if the hyper-sphere centered in y and having radius θ contains at least one of the hyper-spheres associated with the prototypes in the set P.

2.1 Relationship with the NNDD rule

Next the definition of another one-class classifier, namely the NNDD rule, is recalled, and then the relationship between these two rules is pointed out.

First, the definitions of k-th nearest neighbor and of nearest neighbor distances vector are provided.

Given an object x of U, the *k-th nearest neighbor* $nn_{D,\mathrm{d},k}(x)$ of x in D according to d ($nn_k(x)$, for short) is the object y of D such that there exists exactly $k-1$ objects z of D with $\mathrm{d}(x, z) \leq \mathrm{d}(x, y)$. If x is a member of D, then it is its first nearest neighbor, that is $nn_1(x) = x$.

The *k nearest neighbors distances vector* $\delta_{D,\mathrm{d},k}(x)$ of x in D ($\delta_k(x)$, for short) is

$$\delta_k(x) = (\mathrm{d}(x, nn_1(x)), \ldots, \mathrm{d}(x, nn_k(x))),$$

that is the vector consisting of the distances separating x from its first k nearest neighbors in D.

Definition 2.2 ([10]): The *Nearest Neighbor Domain Description rule* according to D, d, k, θ, and r, is the function $\mathrm{NNDD}_{D,\mathrm{d},k,\theta,r}$ (NNDD, for short) from $\mathcal{U}$ to $\{-1, +1\}$ such that

$$\mathrm{NNDD}(y) = \mathrm{sign}(\theta - \|\delta(y)\|_r),$$

where $\mathrm{sign}(z) = -1$ if $z < 0$, and $\mathrm{sign}(z) = 1$ otherwise.

Intuitively, the NNDD rule accepts an input object y if and only if the norm of the associated nearest neighbor distances vector is not greater than the threshold θ. In particular, for $r = 1$, the rule requires that the sum of the distances separating y from its first k nearest neighbors in the reference set is not greater than θ, while for $r = \infty$, the object y is accepted provided that the distance from y to its k-th nearest neighbor is not greater than θ.[1]

1. Recall that $\|(y_1, \ldots, y_d)\|_1 = |y_1| + \ldots + |y_d|$ and, moreover, that $\|(y_1, \ldots, y_d)\| = \max\{|y_1|, \ldots, |y_d|\}$.

The following definition relates the PDD rule and the NNDD rule.

Definition 2.3: Given a set of objects D, the *prototype set* $P(D, \mathrm{d}, k, \theta)$ *associated with* D *with respect to* d*,* k*, and* θ is

$$\{\langle x, \mathrm{d}(x, nn_k(x))\rangle \mid x \in D \text{ and } \mathrm{d}(x, nn_k(x)) \leq \theta\}.$$

That is, the prototype set $P(D, \mathrm{d}, k, \theta)$ consists of all the pairs $\langle x_i, r_i\rangle$, with x_i belonging to D, such that the distance r_i separating x_i from its k-th nearest neighbor in D is not greater than θ.

Relationship between the two rules is clarified by subsequent Theorem 2.4, stating that if the reference set of the PDD rule is set to $P(D, \mathrm{d}, k, \theta)$ then on the objects x of the reference set D the output of the PDD rule is identical to the output of the NNDD rule for $r = \infty$.

Theorem 2.4: Given a set of objects D, a distance function d, and parameters k and θ, it holds that

$$(\forall x \in D)(\mathrm{NNDD}_{D,\mathrm{d},k,\theta,+\infty}(x) = \mathrm{PDD}_{P(D,\mathrm{d},k,\theta),\mathrm{d},\theta}(x)).$$

Proof. Let x be a generic object of D. First, consider the case $\mathrm{d}(x, nn_k(x)) \leq \theta$. Then $\mathrm{NNDD}(x) = \mathrm{sign}(\theta - \|\delta_k(p)\|_{+\infty}) = \mathrm{sign}(\theta - \mathrm{d}(x, nn_k(x))) = +1$. Furthermore, the pair $\langle x, \mathrm{d}(x, nn_k(x))\rangle$ belongs to $P(D, \mathrm{d}, k, \theta)$ and, hence, $\mathrm{d}(x, x) + R(x) = 0 + \mathrm{d}(x, nn_k(x)) \leq \theta$ and $\mathrm{PDD}_{P(D,\mathrm{d},k,\theta),\mathrm{d},\theta}(x) = +1$.

Consider now the case $\mathrm{d}(x, nn_k(x)) > \theta$. In this case $\mathrm{NNDD}(x) = -1$. By contradiction, assume that there exists a pair $\langle y, R(y))\rangle$ in $P(D, \mathrm{d}, k, \theta)$ such that $\mathrm{d}(x, y) + R(y) \leq \theta$. Then, since $R(y)$ is $\mathrm{d}(y, nn_k(y))$, within distance $r_y = \mathrm{d}(x, y) + \mathrm{d}(y, nn_k(y))$ from x there are at least $k+1$ objects of D and, hence, it holds that $\mathrm{d}(x, nn_k(x)) \leq r_y \leq \theta$, which contradicts the hypothesis. □

Thus, Theorem 2.4 states that from the point of view of the objects belonging to the data set D, the prototype set $P(D, \mathrm{d}, \theta, k)$ is the analogous for the PDD rule of the data set D for the NNDD rule.

2.2 PDD Consistent Subset

When the reference set D is large, space requirements to store D and time requirements to find the neighbors of an object in D increase. In the spirit of the reference set thinning problem for the k-NN-rule [13], [14], the concept of NNDD reference consistent subset was defined in [10] and, in the same spirit, next it is provided the definition of PDD consistent subset.

Definition 2.5: Let P be a prototype set and let S be a subset of P. The set S is said to be a PDD *consistent subset* of P with respect to d and θ, if the following relationship hold

$$(\forall \langle x, r\rangle \in P)(\mathrm{PDD}_{P,\mathrm{d},\theta}(x) = \mathrm{PDD}_{S,\mathrm{d},\theta}(x)).$$

It is of interest here to recall the concept of sample compression scheme. A *sample compression scheme* is defined by a fixed rule $\sigma : D \mapsto \sigma(D)$ for constructing a classifier from a given set of data. Given a training set D, it is compressed by finding the smallest subset (the compression set) $S \subseteq D$ for which the classifier $\sigma(S)$ correctly classifies the whole set D. It is known that the size of a sample compression scheme can be used to bound generalization [15], [16].

It can be concluded from the concept of sample compression scheme and from the discussion above that replacing the prototype set P with a consistent subset S of P may improve both generalization and response time.

Importantly, it also holds that a PDD consistent subset S of the set $P(D, \mathrm{d}, \theta, k)$ is the analogous for the PDD rule of the data set D for the NNDD rule, as accounted for in the following theorem.

Theorem 2.6: Given a set of objects D and a PDD consistent subset S of $P(D, \mathrm{d}, k, \theta)$, it holds that

$$(\forall x \in D)(\mathrm{NNDD}_{D,\mathrm{d},k,\theta,+\infty}(x) = \mathrm{PDD}_{S,\mathrm{d},\theta}(x)).$$

Proof. Let x be a generic object of D. If $\mathrm{d}(x, nn_k(x)) \leq \theta$, then $\langle x, \mathrm{d}(x, nn_k(x))\rangle \in P(D, k, \theta)$ and, hence, $\mathrm{PDD}_{S,\mathrm{d},\theta}(x) = \mathrm{PDD}_{P(D,k,\theta),\mathrm{d},\theta}(x) = \mathrm{NNDD}(x)$, by the definition of PDD consistent subset.

If $\mathrm{d}(x, nn_k(x)) > \theta$, then it holds that $\mathrm{NNDD}(x) = \mathrm{PDD}_{P(D,k,\theta),\mathrm{d},\theta}(x) = +1$, and there not exists a pair $\langle y, R(y)\rangle$ in $P(D, k, \theta)$ such that $\mathrm{d}(x, y) + R(y) \leq \theta$. The result then follows since S is a subset of $P(D, k, \theta)$. □

Thus, in order to build an efficient and effective one-class classifier, the task of interest here is to provide an algorithm for computing an as small as possible PDD consistent subset of the prototype set $P(D, \mathrm{d}, k, \theta)$, to be employed as a prototype set of the PDD classifier for discriminating the objects of the class associated with the data set D from the other ones. Due to the way the PDD classifier builds its decision boundary, its consistent subset is expected to be much smaller than the consistent subset associated with the CNNDD rule. Moreover, due to its relationship to the CNNDD rule and to statistical definitions for outliers (see next section), the PDD classifier is also expected to be very effective as far as the classification accuracy is concerned.

2.3 Relationship with statistical definitions

A characterizing point of the PDD rule is that it is related also with unsupervised methods for outlier detection, that are techniques for identifying the most deviating objects in an input data set. In particular, it has relationships with distance-based outlier detection methods, that were introduced in [11]: a point in a data set is a $DB(c, d)$-outlier with respect to parameters c and d, if at least fraction c of the points in the data set lies greater than distance d from it.

It can be shown that for a suitable choice of the parameters k and θ and of the prototype set, the PDD rule rejects the distance-based outliers in the data set. Indeed, let k set to $(1 - c)|D|$, let θ set to the distance value d, and let the prototype set P set to $P(D, d, k, \theta)$. The result follows by Theorem 2.4, after noticing that the distance-based definition coincides with the decision rule of the NNDD classifier under the infinity Minkowski metrics.

An interesting property of definition [11] is that it generalizes several *discordancy tests* to detect outliers given in statistics, other than being is suitable when the data set does not fit any standard distribution. In particular, given a statistical definition Def for outliers, it is said that the distance-based definition *unifies* the definition Def, provided that there exist

specific values c_0 and d_0 for the parameters c and d such that an object is an outlier for Def if and only if it is a $DB(c_0, d_0)$-outlier. This means that, given a data set complying with a certain data distribution for which there exists a test unifiable with the distance-based outlier definition, a suitable PDD classifier can be built which is able to reject the objects that are less probable to occur according to the distribution underlying the data. For example, consider a data set D which is *normally distributed*: it follows from [11] that the PDD classifier rejects the objects lying three standard deviations σ further the mean μ, that is the objects x such that $\left|\frac{x-\mu}{\sigma}\right| \geq 3$, provided that k is set to $k_0 = 0.0012|D|$, θ is set to $\theta_0 = 0.13\sigma$, and the prototype set is $P(D, \mathrm{d}, k_0, \theta_0)$.

For details on the concept of unification above recalled and on the relationship between statistical tests and distance-based outliers, the reader is referred to [11], [17].

Algorithm CPDD

1) for each object x_i in D, determine the distance r_i between x_i and its k-th nearest neighbor in D
2) set P to $\{x_i \in D \mid r_i \leq \theta\}$
3) for each object x_i in P, determine the set N_i composed of the objects y of D such that $\mathrm{d}(x_i, y) + r_i \leq \varrho\theta$
4) set S and C to the empty set
5) while $|C| \leq \eta|P|$ do
   a) determine the object x_j of P such that (break ties in favor of the object such that the value r_j is minimum)
   $$|N_j - C| = \max\{|N_i - C| : x_i \in P\}$$
   b) set S to $S \cup \{\langle x_j, r_j\rangle\}$, and C to $C \cup N_j$
6) return the set S

Fig. 1. The CPDD algorithm.

3 COMPUTATIONAL ANALYSIS

In this section the computational complexity of the problem of computing a minimum size PDD consistent subset is investigated. The reader is referred to [18] for basics on complexity theory, NP optimization problems, and approximation algorithms. Next it is shown that, in the general case, the problem of computing a minimum size PDD consistent subset is not in the APX complexity class, that is, loosely speaking, the class of the NP optimization problems whose optimal solution can be approximated in polynomial time within a fixed factor. Before providing the result, the PDD Consistent Subset Problem is introduced and then its evaluation and decision versions are defined.

Given a prototype set P, a distance metrics d, and a positive real number θ, the PDD *Consistent Subset Problem* $\langle P, \mathrm{d}, \theta\rangle$ is defined as follows: compute a PDD consistent subset S^* of P with respect to d and θ, also said a *minimum size* PDD *consistent subset*, such that, for each PDD consistent subset S of P with respect to d and θ, it holds that $|S^*| \leq |S|$.

The *evaluation version* $\langle P, \mathrm{d}, \theta\rangle_E$ of the problem $\langle P, \mathrm{d}, \theta\rangle$ is defined as follows: compute the size m^* of the minimum size PDD consistent subset of P with respect to d and θ.

Given a positive integer m, the *decision version* $\langle P, \mathrm{d}, \theta, m\rangle_D$ of the problem $\langle P, \mathrm{d}, \theta\rangle$ is defined as follows: reply "yes" if there exists a PDD consistent subset S of P with respect to d and θ such that $|S| \leq m$, and reply "no" otherwise.

Theorem 3.1: The $\langle P, \mathrm{d}, \theta\rangle$ problem (1) is NP-hard, and (2) is not in APX.

Proof. (Point 1) (Membership) Given a subset S of P, having size $|S| \leq m$, it can be checked in polynomial time that, for each $x \in P$, $\mathrm{PDD}_{P,\mathrm{d},\theta}(x) = \mathrm{PDD}_{S,\mathrm{d},\theta}(x)$.

(Hardness) The proof is by reduction from the *Dominating Set Problem* [18]. Let $G = (V, E)$ be an undirected graph, and let $m \leq |V|$ be a positive integer. The *Dominating Set Problem* is: is there a subset $U \subseteq V$, called *dominating set* of G, with $|U| \leq m$, such that for all $v \in (V - U)$ there exists $u \in U$ with $\{u, v\} \in E$?

Let $G = (V, E)$ be an undirected graph. Define the metric d_V on the set V of nodes of G as follows: $\mathrm{d}_V(u, v) = \theta$, if $\{u, v\} \in E$, and $\mathrm{d}_V(u, v) = 2\theta$, otherwise. Let P_V be the set $\{\langle v, 0\rangle \mid v \in V\}$. Next it is proved that G has a dominating set of size m if and only if $\langle P_V, \mathrm{d}_V, \theta, m\rangle_D$ is a "yes" instance.

($\Rightarrow$) Suppose that G has a dominating set U such that $|U| \leq m$. Then $S_U = \{\langle u, 0\rangle \mid u \in U\}$ is a PDD consistent subset of P_V with respect to d_V and θ. Indeed, let v a generic object of V. If $v \in U$, then $\langle v, 0\rangle \in P_V$ and $\mathrm{d}(v, v) + R(v) = 0 + 0 \leq \theta$. Otherwise, $v \notin U$ and there exists $u \in V$ such that $\{u, v\} \in E$; hence, $\langle u, 0\rangle \in S_U$ and $\mathrm{d}(v, u) + R(u) = \theta + 0 \leq \theta$.

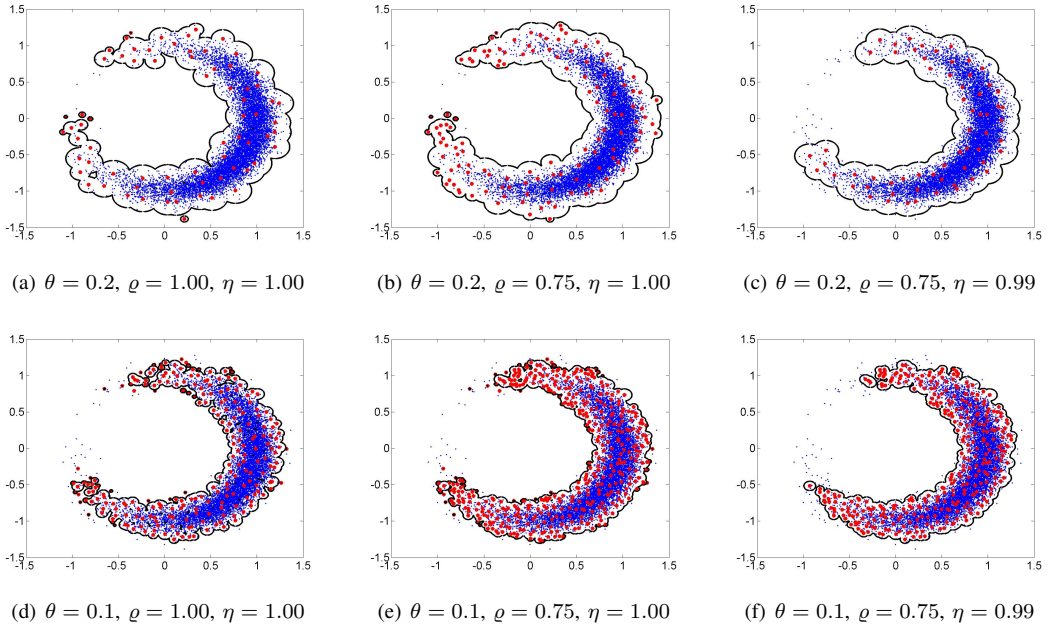
($\Leftarrow$) Suppose that there is a reference consistent subset S_U of P_V such that $|S_U| \leq m$. Then $U = \{u \mid \langle u, 0\rangle \in S_U\}$ is a dominating set of G. By contradiction, assume that there is $v \in (V - U)$ such that, for each $u \in U$, $\{v, u\} \notin E$. Then, there exists $v \in P_V$ such that, for each $\langle u, 0\rangle \in S_U$, $\mathrm{d}(v, u) + R(u) = 2\theta + 0 > \theta$, and S_U is not a PDD consistent subset of P_V. It follows immediately that U is a dominating set for G.

The NP-hardness of the $\langle P, \mathrm{d}, \theta\rangle$ problem follows immediately from the NP-completeness of its decision version.

(Point 2) It is known that the *Minimum Dominating Set Problem*, that is the problem of determining the size of the smallest dominating set of a graph, is not in APX [19].

Next we informally recall the concept of AP-reduction (the reader is referred to [18] for the formal definition). In the context of approximation algorithms, a reduction from a problem A to a problem B should guarantee that an approximate solution of B can be used to obtain an approximate solution for A. Thus, it is needed a function f_A mapping instances of A into instances of B and also a function f_B mapping back solutions of B into solutions of A. In order to preserve guaranteed approximation, the reduction has to satisfy the following property: for any instance x of A, if the performance ratio of the solution y of the instance $f_A(x)$ of B is at most r, then the performance ratio of the solution $f_B(x, y)$ is at most r', where r' depends only on r. An AP-reduction has the property of establishing a linear relation between the performance ratios r and r', thus preserving membership in all approximation classes.

We note that Point 1 of this theorem defines an AP-reduction from the *Minimum Dominating Set Problem* to the *Minimum* PDD *Consistent Subset Problem*. Indeed, the elements of the



(a) $\theta = 0.2$, $\varrho = 1.00$, $\eta = 1.00$ (b) $\theta = 0.2$, $\varrho = 0.75$, $\eta = 1.00$ (c) $\theta = 0.2$, $\varrho = 0.75$, $\eta = 0.99$

(d) $\theta = 0.1$, $\varrho = 1.00$, $\eta = 1.00$ (e) $\theta = 0.1$, $\varrho = 0.75$, $\eta = 1.00$ (f) $\theta = 0.1$, $\varrho = 0.75$, $\eta = 0.99$

Fig. 2. Examples of PDD consistent subsets computed by the CPDD algorithm.

sets U and S_U there defined are in one-to-one correspondence and, hence, the size of these two sets is always identical, that is $|U| = |S_U|$. As an immediate consequence of this reduction, the latter problem does not belong to APX. □

4 THE CPDD ALGORITHM

Figure 1 shows the algorithm CPDD. Given a data set D, the CPDD algorithm computes a PDD consistent subset of the prototype set $P(D, \mathrm{d}, k, \theta)$ associated with D.

The algorithm receives in input a data set D, parameters k a θ, and the additional parameters $\varrho, \eta \in (0, 1]$, whose semantics is discussed in the following (if not otherwise specified, it is assumed that ϱ and η are both set to 1).

Initially, for each object x_i of D, the algorithm determines the distance r_i to its k-th nearest neighbor (step 1). The set P, built in step 2, is composed of the objects occurring in the prototype set $P(D, \mathrm{d}, k, \theta)$, that are the objects x_i such that $r_i \leq \theta$. Hence, for each object x_i in P, the set N_i of the objects of D lying within distance $\theta - r_i$ from it is determined (step 3). Then the algorithm computes the consistent subset S following a greedy strategy (step 5). The set C consists of the objects of P which are accepted by using the current subset S. At each step, the object x_j maximizing the number of objects in $N_j - C$ is selected and inserted in S, until C contains at least the fraction η of the objects in P (until C covers P, if $\eta = 1$).

Next theorem shows that the the size of the solution returned by the algorithm has an approximation factor.

Theorem 4.1: The CPDD algorithm provides a solution having a $1 + \ln(n)$ approximation factor.

Proof. Assume that the parameter ϱ is set to one. We note that the set N_i precisely consists of all the prototypes in $P(D, \mathrm{d}, \theta, k)$ which are correctly recognized through the PDD rule if x_i is included in the PDD consistent subset S.

Given a finite set $\mathcal{S}$ and a collection $\mathcal{C}$ of subsets of $\mathcal{S}$, a *set cover* for $\mathcal{S}$ is a subset $\mathcal{C}'$ of $\mathcal{C}$ such that every element in $\mathcal{S}$ belongs to at least one member of $\mathcal{C}'$. It is clear that the PDD consistent subsets of P are in one-to-one correspondence with the set covers of $\{N_i \mid x_i \in P\}$. The result hence follows by noting that step 5 of the algorithm CPDD is analogous to the greedy algorithm for the *Minimum Set Cover Problem*, the problem of computing a set cover of minimum size, which achieves an approximation factor of $1 + \ln(n)$, where n is the size of the set to be covered. □

Note that steps 4-6 compute a PDD consistent subset of any arbitrary prototype set.

Figure 2 reports some examples of PDD consistent subsets computed by the CPDD algorithm. The data set (blue points) is composed of ten thousands points in the plane. The parameter k has been set to 5, while two distinct values for the parameters θ, ϱ and η have been considered, namely 0.1 and 0.2 for θ, 0.75 and 1.0 for ϱ, and 0.99 and 1.0 for η.

Stars (in red color) denote the prototypes belonging to the PDD consistent subset S, while the (black) curve denotes the decision boundary of the classifier $\mathrm{PDD}_{S,\mathrm{d},\theta}$. The relative size of the PDD consistent subset is (a) 0.7% (70 prototypes) for $\theta = 0.2$, $\varrho = 1$, and $\eta = 1$, (b) 1.3% (128 prototypes) for $\theta = 0.2$, $\varrho = 0.75$, and $\eta = 1$, (c) 0.6% (62 prototypes) for $\theta = 0.2$, $\varrho = 0.75$, and $\eta = 0.99$, (d) 2.3% (227 prototypes) for $\theta = 0.1$, $\varrho = 1$, and $\eta = 1$, (e) 4.4% (439 prototypes) for $\theta = 0.1$, $\varrho = 0.75$, and $\eta = 1$, and (f) 3.4% (337 prototypes) for $\theta = 0.1$, $\varrho = 0.75$, and $\eta = 0.99$.

From these figures it is clear that the smaller the value of the parameter θ, the closer to the data set shape the class boundary, the greater the number of data set objects rejected by the PDD rule, and the greater the number of prototypes belonging to the consistent subset. Moreover, the smaller the value of the parameter ϱ, the greater the number of prototypes

Algorithm Fast CPDD

1) set S, C, and E to the empty set, set D' to D, set $count^*$ to 0, $count$ to $k+1$, and σ to 1
2) while $count > (1-\eta)k$ do
   a) draw a random sample T from D' of size $m = ((1-a)\sigma + a)s$ ($a = 0.1$)
   b) for each object x_i in T, determine the distance r_i between x_i and its k-th nearest neighbor in D
   c) set P to $\{x_i \in T \mid r_i \leq \theta\}$
   d) for each object x_i in P, determine the set N_i composed of the objects y of D such that $\mathrm{d}(x_i, y) + r_i \leq \varrho\theta$
   e) set E to $E \cup P$
   f) do
      i) determine the object x_j of E such that (break ties in favor of the object such that the value r_j is minimum)
         $$|N_j - C| = \max\{|N_i - C| : x_i \in E\}$$
      ii) set $count$ to $|N_j - C|$ and $count^*$ to $\max\{count^*, count\}$
      iii) if $count > (1-\eta)k$ then set S to $S \cup \{\langle x_j, r_j\rangle\}$ and C to $C \cup N_j$
      while $x_j \notin P$ and $count > (1-\eta)k$
   g) set D' to $D' - (T \cup C)$, and σ to $\frac{\log(count)}{\log(count^*)}$
3) return the set S

Fig. 3. The Fast CPDD algorithm.

belonging to the consistent subset, the more accurate the form of the decision boundary, and the smaller the probability of rejecting objects belonging to the class represented by the data set. For example, in Figure 2(a) ($\varrho = 1$) there is a "hole", approximately centered in $(-0.78, -0.78)$, in the lower tail of the data set (but also other smaller "holes" exist along the data set shape), while these regions are covered by the prototypes in Figure 2(a) ($\varrho = 0.75$). Finally, the smaller the value of the parameter η, the smaller the number of prototypes belonging to the consistent subset, but the greater the probability of rejecting objects belonging to the class represented by the data set, since the most sparse regions of the feature space belonging to the class are left uncovered.

5 THE FAST CPDD ALGORITHM

The temporal cost of the CPDD algorithm is quadratic in the number of objects in the data set D, due to the need of computing all the pairwise data set object distances. Thus, when D is very large the associated computational effort may become heavy. To reduce this cost, it is here presented a variant of the basic algorithm, called Fast CPDD (or FCPDD, for short).

Figure 3 reports the algorithm Fast CPDD. The algorithm receives in input the parameters D, k, θ, ϱ, and η, already introduced, plus the novel parameter s, representing the size of a random sample of the data set. The semantics of the parameters ϱ and η has been already discussed. In particular, the parameter η is employed to reject the less probable objects of the data set. However, in the Fast CPDD algorithm this is accomplished by adopting a policy different from that of the CPDD algorithm, policy which is detailed next.

Let x_j be the object selected for insertion in the consistent subset S, and let $count$ denote the number $|N_j - C|$, where N_j is the set of the data set objects y such that $\mathrm{d}(x_j, y) + r_j \leq \varrho\theta$, with r_j the distance from x_j to its k-th nearest neighbor in D. Then, x_j is inserted in S if the condition $count > (1-\eta)k$ holds (see step 2.f.iii of the algorithm). This condition is also employed as stopping criterion of the algorithm, since the algorithm terminates if $count \leq (1-\eta)k$ (see step 2).

Thus, the consistent subset S contains only objects whose inclusion in S contributes to augment the size of the set C, which is the set containing the data set object accepted by using S as reference set of the classifier, of at least $(1-\eta)k$. Note that if η equals one (the default setting), all the data set objects accepted by the PDD rule are included in the set S, as it is the case also for the CPDD algorithm.

In order to alleviate the quadratic temporal cost involved in the computation of all the pairwise data set object distances, at each iteration (step 2) the algorithm Fast CPDD randomly selects a set T of data set objects among those in the set D' (step 2.a). The objects in D' are those not already selected in previous iterations which are not accepted by using the current consistent subset S (see step 2.g, where D' is updated to $D' - (T \cup C)$).

For each object x_i in the random sample T, the distance r_i to its k-th nearest neighbor in D is determined (step 2.b), and for each object x_i in P (that is such that $r_i \leq \theta$; see step 2.c), the set N_i composed of the objects y of D such that $\mathrm{d}(x_i, y) + r_i \leq \varrho\theta$ is computed (step 2.d).

The set E accumulates all the candidate prototypes selected during the various iterations, in that it is set to $E \cup P$ (step 2.e). The next objects to be inserted in S are then selected in the set E (see the cycle in step 2.f). Specifically, at each iteration the object x_j in E maximizing the number $count$ of objects in $N_j - C$ is selected for insertion in S (step 2.f.i). As already explained, the object x_j is inserted in S provided that the condition $count > (1-\eta)k$ is satisfied. The cycle terminates if the object x_j do not satisfy the condition $count > (1-\eta)k$ or if x_j comes from the set P, that is the set containing the last recently selected candidate prototype objects.

The motivation of this policy is the following. It is assumed that the size s of the random sample guarantees that a good approximation of the best object in D' (including the data set objects not currently accepted and not already selected for insertion in the previous iterations) to be added to the set S occurs in the set T and, hence, in the set P. Thus, at each iteration only the best object coming from P is inserted in S. As for the objects in the set $E - P$, that are the objects belonging to the sets P which were selected in the previous main iterations of the algorithm, they are inserted in S provided that they are better than the best object in the current set P, that is until one object of P is selected for insertion in S.

The termination condition of the main cycle of the algorithm (step 2) has been already illustrated. It remains to discuss on the size m of the random sample T. The parameter s specifies the maximum size allowed for a random sample T. In detail, at each iteration, m is set to the value $((1-a)\sigma + a)s$, where a ($a \in (0, 1)$) denotes the smallest value allowed for m, that

is $a \cdot s$ and $\sigma \in [0,1]$ represents a *reduction factor* ($a = 0.1$ by default, denoting the ten percent of the value of the parameter s; however, by setting a to 1, the size m of the random set is held fixed to s).

The value of σ is initially one, so that during the first iteration the size m of the random sample T is precisely s, that is the user-specified value. Let $count^*$ denote the maximum value assumed so far by the counter $count$ (see step 2.f.ii). At the end of each iteration, the reduction factor σ is set to the value $\frac{\log(count)}{\log(count^*)}$ (step 2.h). Thus, the larger the relative increase (with respect to the maximum one) of the set C in the last iteration, the closer to s the size m of the random sample to be drawn in the current iteration.

The rationale underlying this policy is that the size of the returned consistent subset is influenced mostly by the right selection of the objects in D coming from the most populated regions of the feature space. Intuitively, the value of σ is related to relative likelihood for the objects in $D-C$ to represent an outcome of the distribution of the class represented by the objects in D. Thus, since m is directly proportional to σ, the care the algorithm devotes to the selection of the next object to be inserted in S is directly proportional to the probability of the data set objects left uncovered by the current set S.

This completes the description of the algorithm. Assume η is set to 1. In this case, the algorithm terminates when in the set E and, hence, in the current set P, there are no more objects accepted by the PDD rule. This means that in the last iteration the algorithm selected a random sample T composed only of outliers. Although this does not rigorously imply that there are no more inliers in D', it is anyway the case that the occurrence probability associated with the residual inliers in D' is comparable to that of the outliers, and their inclusion in S does not pay much. Thus, the termination condition employed, that is $count \leq (1-\eta)k$, is a good replacement of the exact termination condition $D' = \emptyset$. However, by setting $\eta = 1$ and by substituting the condition in step 2 with the condition $D' \neq \emptyset$, the following property readily holds.

Theorem 5.1: The *Fast CPDD* algorithm computes a PDD consistent subset.

As for the differences of the Fast CPDD algorithm with respect to the CPDD algorithm, it must be pointed out the latter algorithm selects the next prototype to be added to the current set S from the set $D - S$, while the former one takes into account only the objects in the set E. Note that the set P selected at each iteration by the Fast CPDD is a subset of $D' = D - (E \cup C)$ (with C a superset of S), and not of $D - S$. This policy is needed by the Fast CPDD in order to accelerate convergence, since the set C grows more rapidly than the set S. Moreover, note that the current set E, from which Fast CPDD selects the next prototypes to be inserted in S, contains also objects coming from $C - S$ (recall that E is augmented with P; see step 2.e).

To conclude, the temporal cost of the algorithm is considered. Let m_i be the size of the random sample T at the generic iteration of the algorithm, and let M be $\sum_i m_i$. Since M cannot be greater than $|D|$, the number of distance computations of the algorithm FCPDD is $M \cdot |D| \leq |D|^2$. Thus, the temporal cost of the algorithm is quadratic, but in practice the number of distances to be computed could represent a small fraction of the worst case.

5.1 Analysis of the FCPDD algorithm

Let x^* be the object in D whose insertion in S will maximize the increment of the size of C. Moreover, assume the data set size is very large (potentially infinite), so that the distance to the k-th nearest neighbor can be approximated with a very small value (close to zero), and that the data set is distributed according to a pdf $f(x)$. Then, let p^* denote the probability that the Fast CPDD algorithm selects for insertion in S an object coming from the most populated sub-region $\mathcal{R}^* = I_\theta(x^*) - \mathcal{R}(S)$ of the accepting region left uncovered by the current set S, where $I_\theta(x^*)$ is the neighborhood of radius θ of the object x^* and $\mathcal{R}(S)$ denotes the set $\{y \in \mathcal{U} : \mathrm{PDD}_S(y) = +1\}$.

Thus, the probability p^* that the object selected by the Fast CPDD algorithm comes from the region $\mathcal{R}^*$ is

$$p^* = 1 - (1 - q^*)^s,$$

where q^* is the probability for an object of the accepting class to belong to the region $\mathcal{R}^*$, and $(1 - q^*)^s$ is the probability that no object in T belongs to $\mathcal{R}^*$.

At the first iteration, q^* is such that

$$q^* = \int_{\mathcal{R}^*} f(v)\,\mathrm{d}v,$$

while during the generic iteration of the algorithm, q^* is not smaller than

$$q^* = \left(\int_{\mathcal{R}^* \setminus \mathcal{R}(S)} f(v)\,\mathrm{d}v\right) / \left(\int_{\mathcal{U} \setminus \mathcal{R}(S)} f(u)\,\mathrm{d}u\right),$$

since q^* is the probability that a randomly selected object in $D - C$ belongs to $\mathcal{R}^*$.

As an example, consider the standard normal distribution with mean $\mu = 0$. It has been already pointed out that for $R = 0.13$ the 3‰ of the data set objects are rejected. Thus, let $\theta = R$, then the probability that a generic object comes from the most probable region $[\mu - R, \mu + R] = [-0.13, 0.13]$ of the data set is

$$q^* = \Phi(0.13) - \Phi(-0.13) \approx 0.1034,$$

and for $s = 100$,

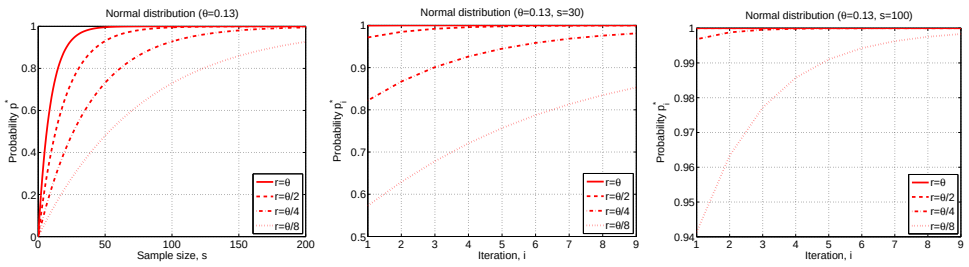
$$p^* = 1 - (1 - 0.1034)^{100} \approx 0.99998$$

regardless of the size of the data set.

Let $\mathcal{R}^*$ denote now the region of radius r centered in x^* (for $r = 0$ the region $\mathcal{R}^*$ contains only x^*). At the first iteration it holds that

$$p^* = 1 - (1 - (\Phi(r) - \Phi(-r)))^s.$$

Figure 4(a) reports the probability p^*, for various values of $r \leq \theta$ and sample sizes s. It is clear that the probability of selecting a close to optimal object rapidly increases with the sample size s.



(a) (b) (c)

Fig. 4. Analysis of the Fast CPDD algorithm on the normal distribution.

Moreover, let $i > 0$ denote the generic iteration at which the region $\mathcal{R}(S)$ associated with the set C is

$$\mathcal{R}(S) = [\mu - (2i-1)\theta, \mu + (2i-1)\theta].$$

Then, the probability p_i^* of selecting the next prototype in the optimal region $\mathcal{R}^*$ is upper bounded by

$$p_i^* = 1 - (1 - q_i^*)^s,$$

where q_i^* is

$$q_i^* = \frac{2(\Phi(2i\theta + r) - \Phi(2i\theta - r))}{1 - (\Phi((2i-1)\theta) - \Phi(-(2i-1)\theta)))}.$$

Figures 4(b) and (c) report the probability p_i^* in correspondence of various values of r for $s = 30$ and $s = 100$, respectively. For $s = 30$, representing a very small sample, the probability p_i^* is close to 1 for $r \geq \theta/2$ and above 0.9 for r greater than $\theta/4$. For $s = 100$, the probability p_i is very high in all cases considered. Moreover, interestingly the figures show that the probability to select an optimal object on this distribution even increases with the number iterations performed.

This analysis confirm that, provided that the employed random sample size s is sufficiently large, FCPDD is expected to behave very similarly to CPDD.

6 EXPERIMENTAL RESULTS

In this section, experiments involving the CPDD rule are presented. First of all, some of measures employed during experiments are described. The *empirical error* is the fraction of data set objects which are rejected by the CPDD classifier. The *True Positive Rate* (TPR, for short) is the fraction of normal objects accepted by the classifier, while the *False Positive Rate* (FPR, for short) is the fraction of abnormal objects accepted by the classifier. Dually, the *False Negative Rate* (FNR, for short) is the fraction of normal objects rejected by the classifier, while the *True Negative Rate* (TNR, for short) is the fraction of abnormal objects rejected by the classifier. It holds that FNR=1-TPR and FPR=1-TNR. The ROC curve is the plot of the FNR versus the TNR (or, specularly, TPR versus FPR), and the area under the ROC curve (AUC, for short) provides a summary to compare two classifiers. In order to determine the ROC curve, for a fixed value of k the parameter θ was varied from zero to a suitable large value and, then, the FNR and the TNR have been measured.

Experiments are organized as follows. Section 6.1 presents a sensitivity analysis of the CPDD rule. In Section 6.2 the ROC curves of the CPDD method are examined. Section 6.3 compares CPDD and CNNDD in presence of noise. Section 6.4 discusses differences with other SVM-based domain description algorithms. Finally, Section 6.5 presents results obtained by using the FCPDD algorithm.

6.1 Sensitivity analysis

In this section a sensitivity analysis of the CPDD rule is accomplished in order to understand the behavior of the method. Specifically, the parameter k has been varied in the range $[1, 32]$ and then the following combinations of values for ϱ and η have been considered: $\varrho = 1$ and $\eta = 1$, $\varrho = 0.9$ and $\eta = 1$, and $\varrho = 1$ and $\eta = 0.95$. For each combination of k, ϱ, and η, the AUC has been determined.

In the following, the results obtained on the *Image segmentation*, *Ionosphere*, *Satellite image*, and *Isolet* data sets [20] are described. In particular, for the *Image segmentation* data set (19 attributes) the *path* class (330 objects) forms the normal class, while the remaining 1,980 objects represent anomalies, for the *Ionosphere* data set (34 attributes) the *good* class (225 objects) is the normal one, while the objects of the *bad* class are the anomalies, for the *Satellite image* (36 attributes) the *red soil* class (1,533 objects) represent the normal class, while the remaining 3,902 objects are the anomalous ones, and for the *Isolet* data set (617 attributes) the class 1 (240 objects) represents the normal class, while 240 objects from the other classes form the anomalies.

Figure 5 shows the results. In all graphics, the abscissa reports the value of the parameter k. Figures 5(a)-(c) show the AUC of the CPDD method for the various combinations of parameters considered, together with the AUC of the CNNDD method for $r = +\infty$. The AUC of CPDD is inversely proportional to ϱ (see the curve for $\varrho < 1$), and directly proportional to η (see the curve for $\eta < 1$). Moreover, the AUC of CPDD and CNNDD are comparable and the best AUC value achieved by the two methods is about the same. The different behavior of the two methods can be better understood after having examined Figures 5(d)-(f), reporting the *size areas* of the methods, that is the area under the curve of the false negative rate versus the relative size of the CPDD reference subset, a measure which intends to provide a summary of the reference set size reduction achieved by the methods. As far as the CPDD method is concerned, the greater the

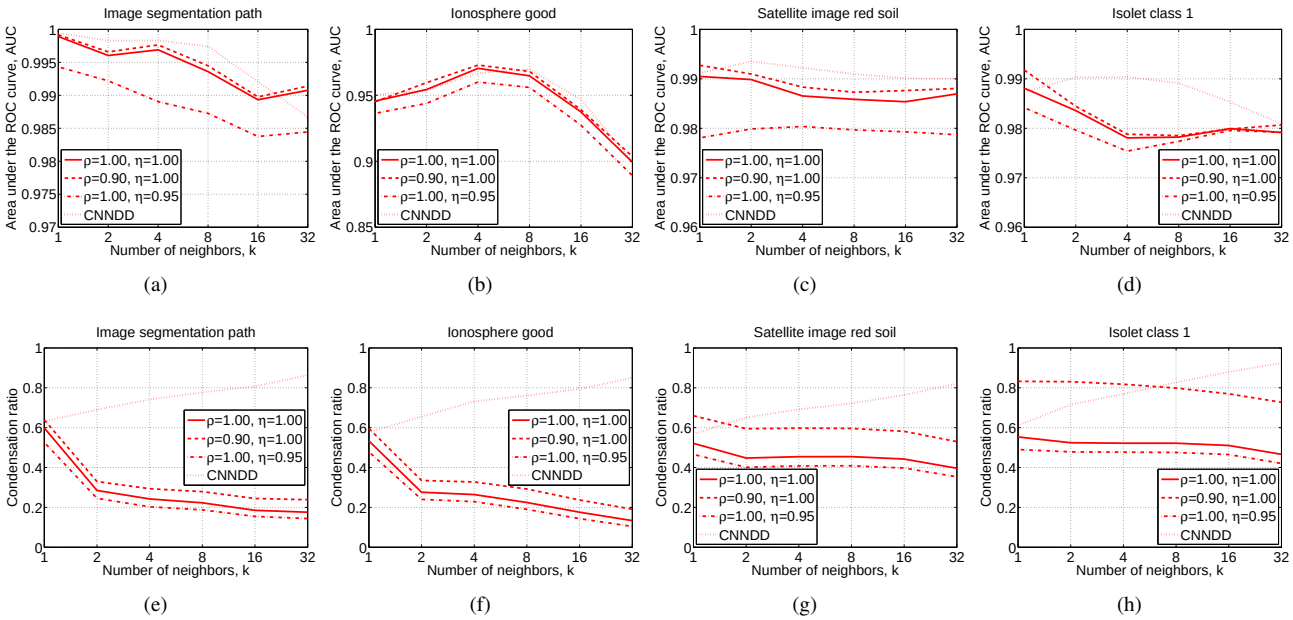


Fig. 5. Sensitivity analysis of the CPDD rule.

parameter k, the smaller the size area. On the contrary, the size noticeably increases with k for the CNNDD method. This can be explained since CNNDD retains in the reference subset more objects than CPDD.

Notice that, for $k = 1$ ($\varrho = 1$ and $\eta = 1$) the decision boundaries of the CPDD and CNNDD classifiers tend to coincide (since in this case it holds that $R(x) = 0$ for x in D), and this accounts for the close behavior of the two methods when $k = 1$. However, while the size of the CPDD consistent subset is inversely proportional to the value of the parameter k, exactly the opposite holds for the size of the CNNDD consistent subset. To better understand this point, consider Figure 6 showing how the FNR and TNR of CPDD and CNNDD vary with k on the *Ionosphere* and *Satellite image* data sets ($k \in \{1, 2, 4, 8\}$). As far as the CNNDD method is concerned, the curves highlight that when k increases, the FNR and TNR curves move towards the right side of the graphic, which means that in order to achieve the same combination of FNR and TNR values a larger and larger subset has to be employed. Differently, the same combination of FNR and TNR values can be achieved with a subset having the same size or even smaller, when the CPDD method is considered.

It can be concluded that in order to achieve small ratios $|S|/|D|$ the CNNDD rule has to be employed with little values of k, while the same constraint does not hold for the CPDD rule. This property represents a limitation in different scenarios, as discussed in the following.

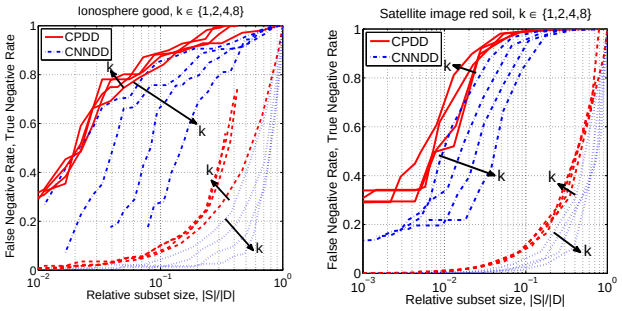


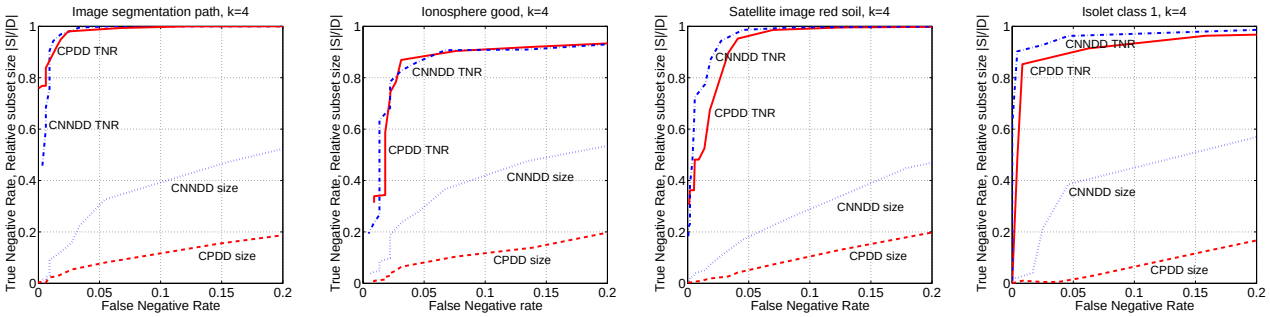
Fig. 6. FNR, TNR and subset size versus k.

6.2 ROC curves

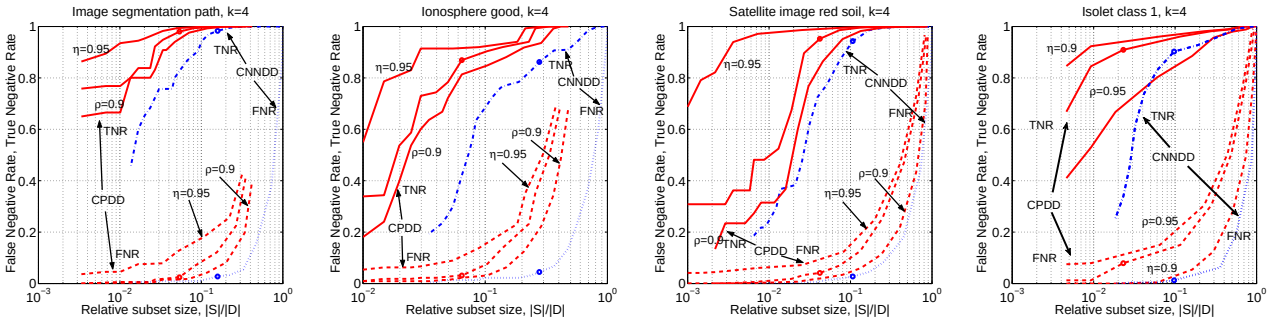
While in previous section the behavior of CPDD has been illustrated through the use of the AUC, here the ROC curves are considered. To obtain these curves, k as been set to 4 and θ has been varied from zero to a suitable large value. ROC curves are reported in Figure 7, which includes also the corresponding curves of the CNNDD method for comparison purposes (notice that, according to the analysis accomplished in the previous section, $k = 4$ is one of the best settings for CNNDD).

Figures 7(a)-(c) show the ROC curves of the CPDD (solid lines) and CNNDD (dash-dotted lines) methods and also the relative size $|S|/|D|$ of the corresponding consistent subsets S achieving the same value of FNR. From these curves it is clear that the the CPDD consistent subset (dashed lines) is much smaller than the corresponding CNNDD subset (dotted lines) guaranteeing the same FNR. Moreover, as already noticed, the AUCs of the two methods are very similar.

Figures 7(d)-(f) report the TNR (solid lines) ad FNR (dashed lines) of the CPDD method and the TNR (dash-dotted lines) and FNR (dotted lines) of the CNNDD method, as a function of the relative subset size $|S|/|D|$. For the CPDD method the pair of parameters $\varrho = 1, \eta = 0.95$ (upper curve), $\varrho = 1, \eta = 1$ (middle curve), and $\varrho = 0.9, \eta = 1$ (lower curve) have been considered. For the *Isolet* data set, the parameters $\eta = 0.90$ and $\varrho = 0.95$ have been employed in the upper and lower curve, respectively, due to the high dimensionality of the data.



(a) CPDD=0.997, CNNDD=0.998 (b) CPDD=0.970, CNNDD=0.967 (c) CPDD=0.986, CNNDD=0.992 (d) CPDD=0.978, CNNDD=0.990



(e) (f) (g) (h)

Fig. 7. Comparison between the CPDD and the CNNDD rule.

As far as the middle curves of the CPDD ($\varrho = 1, \eta = 1$) and the curves of the CNNDD is concerned, it can be noted that for the same value of FNR or TNR the subset of the CPDD is sensibly smaller than that of the CNNDD. As notable examples (highlighted by means of big points on the curves) compare (1) the CPDD subset of relative size 0.064, achieving FNR=0.031 and TNR=0.869, with the CNNDD subset of relative size 0.277, achieving FNR=0.045 and TNR=0.862, for the *Ionosphere* data set, (2) the CPDD subset of relative size 0.054, achieving FNR=0.024 and TNR=0.997, with the CNNDD subset of relative size 0.158, achieving FNR=0.027 and TNR=0.983, for the *Image segmentation* data set, (3) the CPDD subset of relative size 0.042, achieving FNR=0.041 and TNR=0.952, with the CNNDD subset of relative size 0.106, achieving FNR=0.027 and TNR=0.943, for the *Satellite image* data set, and (4) the CPDD subset of relative size 0.023, achieving FNR=0.079 and TNR=0.909, with the CNNDD subset of relative size 0.097, achieving FNR=0.012 and TNR=0.902, for the *Isolet* data set.

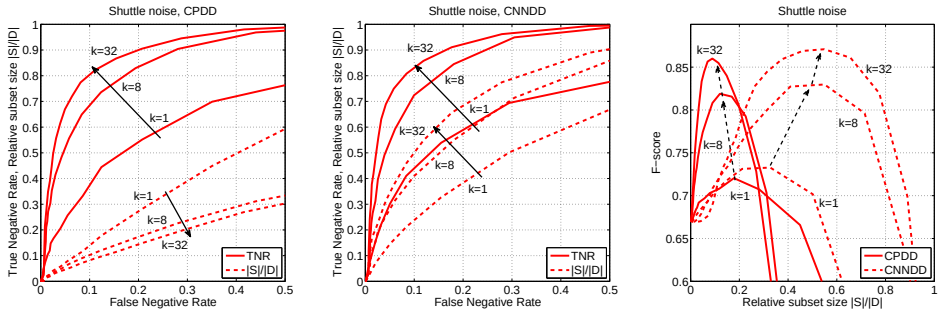
As far as the upper curves of the CPDD is concerned ($\varrho = 1, \eta < 1$), it can be noted that by decreasing the value of the parameter η, very high values of TNR are obtained in correspondence of very small subsets, but the associated FNR worsens with respect to the case $\eta = 1$. This can be explained since the smaller the parameter η, the greater the portion of the accepting region of the PDD rule which is leaved uncovered by the CPDD consistent subset. Conversely, as far as the lower curves of the CPDD is concerned ($\varrho < 1, \eta = 1$), it can be noted that by decreasing the value of the parameter ϱ, the FNR improves while the TNR gets worse. This can be explained since the smaller the parameter ϱ, the greater, and also the closer to each other, the number of prototypes composing the CPDD consistent subset.

Hence, by properly setting the parameters ϱ and η the user can tune the trade off between FNR and TNR and, simultaneously, between subset size and accuracy. The following table summarizes the AUCs of the CPDD for various combinations of the parameters ϱ and η, and $k = 4$.

Data set	ϱ	1.00	0.90	1.00	0.90
	η	1.00	1.00	0.95	0.95
Image segmentation		0.997	0.996	0.989	0.990
Ionosphere		0.970	0.972	0.956	0.967
Satellite image		0.986	0.989	0.981	0.987
Isolet		0.978	0.979	0.975	0.976

6.3 Treatment of noise

This section discusses an important scenario in which it is particularly profitable to be able to employ a large value for k. As already pointed out in [10], in presence of noise in the reference set, the CNNDD classifier can be made much more robust by increasing the parameter k. In order to compare CPDD and CNNDD, here it is considered the same experimental setting of [10] on the *Shuttle* data set [20]: the 34,108 points (9 attributes) of the *Rad Flow* class have been equally partitioned in a training set and an inlier test set (of 17,054 points each), whereas the 9,392 points of the other classes form an outlier test set. A noisy version of the training set has been obtained by adding the 5% (850 points) of mislabeled points (randomly selected points belonging to the outlier test set). Figure 8 shows the result of the experiment. Figure 8(a) reports the ROC curves (solid lines) of CPDD and the relative sizes of the consistent subsets (dashed lines), while Figure 8(b) reports the same information for CNNDD



(a) (b) (c)

Fig. 8. Experiments in presence of noise.

($k \in \{1, 8, 32\}$). By increasing k, both the two methods considerably increase their accuracy, though for the same k the AUC associated with CNNDD is slightly greater than that associated with CPDD. However, as far as the subset size is concerned, the difference between the two methods is marked. As a matter of fact, it can be observed that the size of the CNNDD subset rapidly grows with k, while exactly the opposite holds for the CPDD subset. For example, for $k = 32$, when $FNR \approx 0.1$ and $TNR \approx 0.8$, the ratio $|S|/|D|$ is about 0.5 for CNNDD, but less than 0.1 for CPDD. This behavior is better visualized in Figure 8(c), reporting the F-score associated with a certain value of relative subset size. The F-score is a combined measure of precision and recall, which are related to FNR and TNR. It can be noticed that the greater the F-score, the smaller (greater, resp.) the size of the CPDD (CNNDD, resp.) consistent subset. While CNNDD appears to be slightly more accurate than CPDD for the same value of k, CPDD is undoubtedly vastly more accurate when the same value of relative subset size is considered (see the curves for $|S|/|D| < 0.2$).

6.4 Comparison with other one-class classifiers

In this section the CPDD classifier is compared with the one-class SVM [21], [22] and the Core Vector Data Description (CVDD) classifiers[2] [5], [6], two well-known one-class classification methods, both based on the SVM algorithm. As for the one-class SVM, the radial basis function kernel has been employed. The parameter γ has been varied and the curve associated with the best AUC has been selected. To obtain the ROC curve for the CVDD method, the regularization parameter C has been varied from 1 to 10^6. The radial basis function (RBF) kernel with the parameter γ automatically determined by the algorithm has been used. As for the CPDD rule, the parameters employed are $k = 4$, $\varrho = 1$, and $\eta \in \{0.95, 1.00\}$.

Figure 9 compares the ROC curves of the methods. As far as the size of the model of the one-class SVM is concerned, for small FNRs the size of the CPDD subset is similar to the number of support vectors of the one-class SVM (for greater values of FNRs the former number is much smaller than the latter one). Moreover, by setting the parameter η to 0.95, the size of the CPDD subset can be further decreased. If we consider the results for $\eta = 0.95$ (see also the table in Section 6.2), CPDD performs better than SVM both in terms of AUC that in terms of size of the model on the first three data set. On the fourth one, that is *Isolet*, the AUC is about the same, but the model of CPDD is smaller. The AUCs of CVDD and CPDD were comparable, with CPDD performing better than CVDD in some experiments, and the vice versa in some others. However, as far as the size of the model is concerned, for low FNRs the CVDD classifier presents a noticeable number of support vectors, amounting to about the 20% of the data set size. As a result, the model of CVDD is much more large than that of CPDD, which is an undesirable property.

Very high-dimensional data sets. SVM-based algorithms are known to perform well with high-dimensional data. Hence, next it is considered the *Arcene* very-high dimensional data set [20]. The objects of this data set are composed of $10{,}000$ features, each feature indicating the abundance of proteins in human sera for a certain mass value. The training set and the inlier test set consist of 44 elements each, while the outlier test set contains 112 elements. Figure 10 reports the ROC curves of CPDD (marked with circles; $k = 1$), one-class SVM (marked with triangles; RBF kernel with γ automatically selected), and CVDD (marked with squares; RBF kernel with $\gamma = 10^{-9}$) and the corresponding size of the models (dashed curves). As for the accuracy, CPDD performed very well. Compare the AUCs of the different methods, that are 0.812 for CPDD, 0.615 for one-class SVM, and 0.778 for CVDD. Moreover,

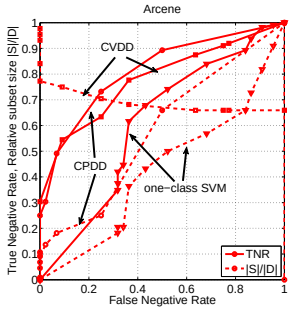
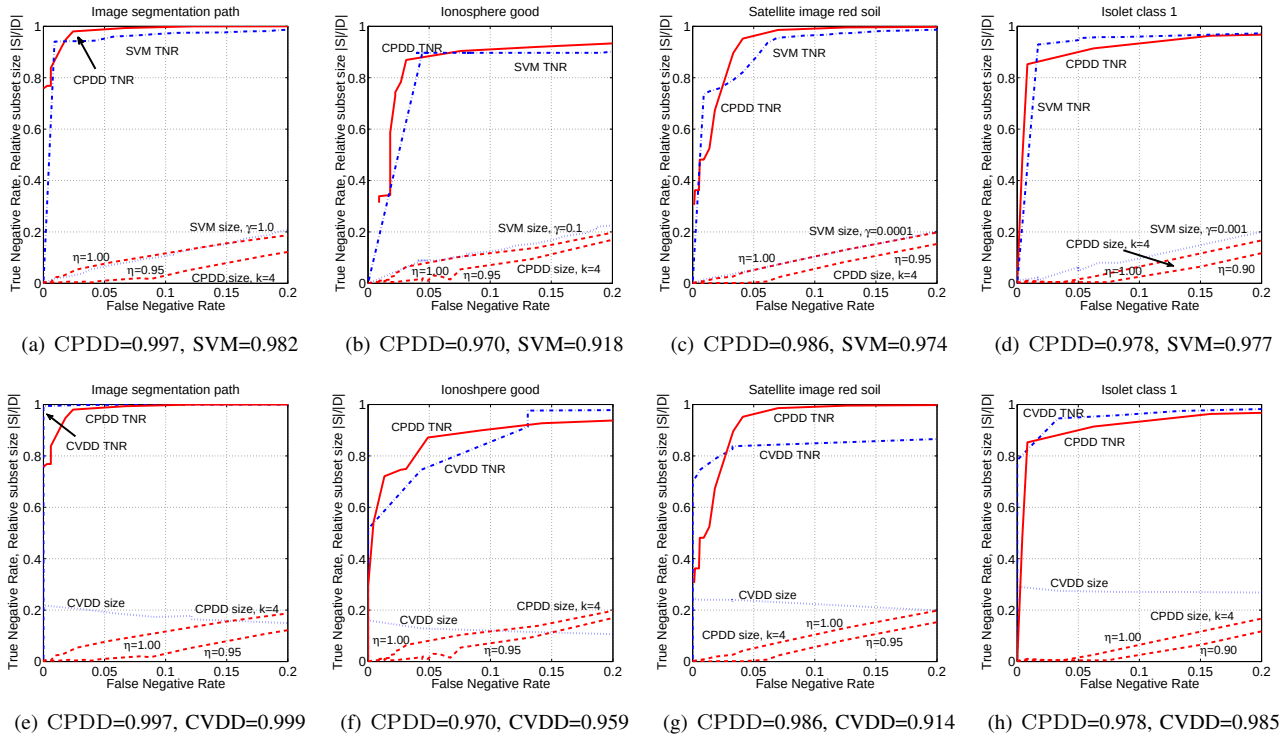


Fig. 10. Experiment on a high-dimensional data set.

2. The LibCVM implementation of CVDD available at `http://www.cse.ust.hk/~ivor/cvm.html` has been used.



(a) CPDD=0.997, SVM=0.982 (b) CPDD=0.970, SVM=0.918 (c) CPDD=0.986, SVM=0.974 (d) CPDD=0.978, SVM=0.977

(e) CPDD=0.997, CVDD=0.999 (f) CPDD=0.970, CVDD=0.959 (g) CPDD=0.986, CVDD=0.914 (h) CPDD=0.978, CVDD=0.985

Fig. 9. Comparison between the CPDD rule and the CVDD.

as for the size of the model, CVDD confirmed the behavior already observed, in that the number of its support vectors corresponds to a considerable fraction of the training set size. In this case, one-class SVM presents a model which is smaller than that of CPDD. However, it must be noticed that in this experiment the ROC curve of one-class SVM is undefined for FNRs in the range $[0, 0.3]$ and that its accuracy is not very good compared with the other two methods. Summarizing, CPDD exhibited the best combination of accuracy and size of the model.

6.5 Experiments using the Fast CPDD method

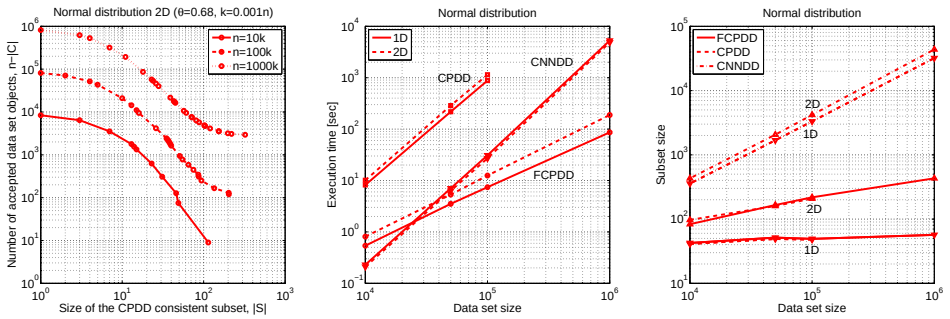
This section presents experiments conducted by using the Fast CPDD algorithm. FCPDD has been compared with CPDD and CNNDD. In all the experiments, the sample size s has been set to $s = 100$. Moreover, if not otherwise stated, the value of the parameter θ has been set so that the corresponding empirical error is approximatively $3‰$.

Scalability analysis. In order to study the scalability behavior of the FCPDD algorithm, in this section experiments on two families of synthetically generated data sets, that are the *Normal 1D* (a one-dimensional normal standard distribution) and *Normal 2D* (a two-dimensional normal standard distribution), are reported. The data sets of the same family differ for the number n of objects, with n varying from ten thousands to one million. The parameter θ has been set to 0.13 for the *1D* data set, and to 0.68 for the *2D*, corresponding about to an empirical error of the $3‰$. The parameter k has been set to about the $1‰$ of n. Results of FCPDD are averaged over ten runs.

Figure 11(a) reports the number of accepted data set objects (the value $n - |C|$) in correspondence of the size $|S|$ of the consistent subset S determined during the execution of the FCPDD algorithm for the *2D* data set. Each circle on the curve represents the size of $|S|$ at the end of a generic main iteration of the algorithm (step 2 of Figure 3). The FCPDD algorithm converges in few iterations and produces a subset composed of a very small fraction of data set objects. It is below the $1‰$ in all cases.

Figure 11(b) shows the execution time of the FCPDD, CPDD, and CNNDD algorithms. It is clear that FCPDD is faster than CPDD of some orders of magnitude and, moreover, that the relative difference between the two methods increases with the size of the data set. As for the CNNDD method, it is faster than FCPDD for small data sets (when absolute execution times are of few seconds), but FCPDD scales much better than CNNDD and, for large data sets, the former method clearly outperforms the latter one. E.g. compare the 86.5 (188.4, resp.) seconds employed by FCPDD on the *1D* (*2D*, resp.) data set, with the $5{,}282.8$ ($4{,}856.3$, resp.) seconds employed by CNNDD on the same data set.[3] To better understand the time savings achievable by using FCPDD instead of CPDD, consider the *relative number of distances*, that is to say the actual number of distances computed by FCPDD divided by the worst case quadratic one. On the *Normal 1D* data set, the relative number of distances is 7.73% for 10K objects, 1.31% for 100K, and 0.10% for 1,000K, while on the *Normal 2D* data set, is 7.94% for 10K, 1.46% for 100K, and 0.17% for 1,000K.

3. Experiments have been performed on a computer having a 2.40GHz Core 2 Duo CPU and 4GB of main memory.



(a) (b) (c)

Fig. 11. Sensitivity analysis of the Fast CPDD algorithm.

Figure 11(c) shows the subset size of the FCPDD, CPDD, and CNNDD algorithms. Interestingly, despite the fact that FCPDD does not consider all the pairwise object distances, it is able to return a subset of size practically identical to that computed by CPDD, but at a reduced computational cost. It can be also noted that the subset computed by CNNDD grows more rapidly with respect to CPDD/FCPDD. E.g. compare the 57 (431, resp.) objects employed by FCPDD on the *1D* (*2D*, resp.) data set, with the 32,015 (43,704, resp.) objects employed by CNNDD on the same data set.

Accuracy with respect to CPDD. Next the accuracy of the CPDD and FCPDD methods is compared on the *Shuttle* data set (the normal class consists of 34,108 objects with 9 attributes, while there are 9,392 abnormal test objects). The following table reports the results for $k = 8$ and $\theta \in [10, 30]$.

		$\theta = 10$	$\theta = 20$	$\theta = 30$
Subset size	FCPDD	1,160	90	59
	CPDD	911	106	57
Exec. time [sec]	FCPDD	10.2	3.5	2.9
	CPDD	123.4	163.1	204.9
Emp. err. [%]	FCPDD	0.9	0.7	0.5
	CPDD	0.9	0.6	0.5
TNR [%]	FCPDD	92.9	81.3	69.5
	CPDD	92.2	82.6	68.6

Results confirm that the size of the subset computed by FCPDD is close to that of the subset computed by CPDD. As for the execution time, FCPDD is one or two orders of magnitude faster than CPDD. Importantly, as far as the empirical error and the TNR are concerned, they are always similar. The results agree with those previously illustrated. It can be concluded that the Fast CPDD computes a subset of size and accuracy comparable to that associated with the subset returned by the CPDD, while guaranteeing great time savings.

Experiments on very large data sets. In this experiment the *Forest Cover Type* very large data set, consisting of 10 quantitative variables, has been considered. The most populated class, that is the *Lodgepole Pine*, was designated as the normal class and partitioned in a training set of 254,971 objects and test set of 28,330 objects. The parameter k has been varied between 150 and 1,500. The size of the consistent subset is reported in Figure 12(a). The smaller k, the larger the consistent subset. For $k = 150$ the relative size of the subset is very small, less than the 3%. Figure 12(b) shows the empirical error (the curve displayed corresponds to the value 100 minus the actual value of empirical error) and the true negative rate on the classes *Cottonwood/Willow* and *Krummholz*, which are the two classes best separated from the normal one. For $k = 150$ the classifier appears to be very effective. Moreover, the following table reports the absolute and relative execution times of the FCPDD algorithm.

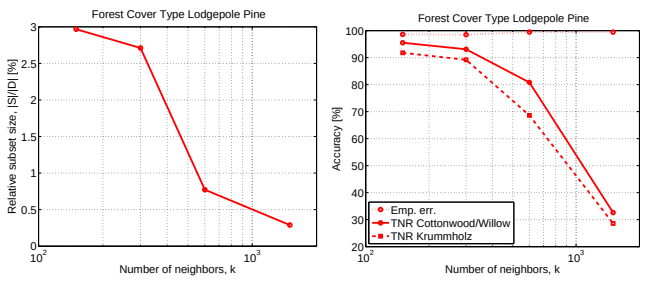
	$k = 150$	$k = 300$	$k = 600$	$k = 1{,}500$
Exec. time [sec]	540.1	480.3	186.4	92.7
Rel. exec. time [%]	6.3	5.7	2.1	1.0

Thus, FCPDD was able to determine an high quality consistent subset with great time savings, since in all cases only a small fraction of the pairwise distances were computed.

7 DISCUSSION AND CONCLUSIONS

One-class classification methods are the natural option when only data from one single class is available and the goal is to discriminate between objects belonging to the class represented by the available data and objects that do not belong to that class. The assumption is that one is able of modeling only normal behavior and everything deviating from it should be rejected. In principle, this kind of approach could be used in a multi-class environment, but it is not the standard setting for one-class classifiers, which are semi-supervised learners.

In this work the Prototype-based Domain Description one-class classifier has been presented. It has been shown that the PDD classifier is equivalent to the NNDD classifier and that it generalizes statistical tests for outlier detection. Moreover, the concept of PDD consistent subset has been introduced and it has been shown that computing a minimum size PDD



(a) (b)

Fig. 12. Experiments on the *Forest Cover Type* data set.

consistent subset is not approximable within any constant factor. Two algorithms for computing a PDD consistent subset have been presented: CPDD guarantees a logarithmic approximation factor, while FCPDD efficiently manages very large data sets. Interestingly, despite the fact that FCPDD does not consider all the pairwise object distances, it has been shown FCPDD is able to return a subset of size practically identical to that returned by CPDD, but at a reduced computational cost.

The PDD classifier is nearest neighbor based, but it clearly differs from the nearest neighbor classification rule. In that respect, the same considerations drawn in [10] concerning differences between NNDD and the k nearest neighbor rule can be applied to PDD.

Other peculiarities of the novel method and differences with existing ones are summarized next. As already pointed out in [10], the CNNDD method can be very accurate, but in order to achieve reasonable condensation ratios it has to be employed with small values for the parameter k. However, this characteristic may represent a limitation in different scenarios. As a matter of fact, using a small k increases the risk of incorporating noise in the model, thus lowering the quality of the classifier. Moreover, the notion of unification with statistical definitions introduced in Section 2.3 requires to employ values of k which are related to the size of the training set. For $k = 1$ CPDD tends to coincide with CNNDD and, in general, CPDD retains an interesting relationship with the CNNDD rule (see Theorem 2.6). Despite these facts, the larger k, the smaller the size of the CPDD consistent subset, while exactly the opposite holds for CNNDD. And, indeed, experiments concerning the treatment of noise (see Section 6.3) highlight that CPDD is undoubtedly much more accurate than CNNDD when the same value of relative subset size is considered. Also, experiments concerning the scalability analysis employing values for the parameters suggested by the statistical theory (Section 6.5), have shown that the subset computed by CNNDD grows much more rapidly with respect to that computed by CPDD. The same experiment shows that FCPDD scales much better than CNNDD, with the former method clearly outperforming the latter one in terms of execution time.

Comparison with SVM-based one-class classification methods is also informative. In particular, CPDD is more accurate than the one-class SVM and presents a model of smaller size. The accuracy of CPDD and CVDD are comparable, but as far as the size of the model is concerned, the model of CVDD is sensibly larger than that of CPDD since for low values of false negative rate the CVDD classifier presents a notable number of support vectors. Despite the fact that SVM-based methods are known to perform well in high-dimensions, CPDD exhibited the best combination of accuracy and size of the model on this kind of data.

To conclude, the CPDD method has solid theoretical foundations, both from the point of view of the underlying notion of normality than from the algorithmic point of view. CPDD overcomes some limitations of the CNNDD rule while retaining its good accuracy properties, is competitive over other one-class classification algorithms, and is also effective in classifying large data sets.

REFERENCES

[1] F. Angiulli, "Prototype-based domain description," in *European Conference on Artificial Intelligence*, 2008, pp. 107–111.

[2] D. M. J. Tax, "One-class classification," Ph.D. dissertation, Delft University of Technology, June 2001.

[3] B. Schölkopf, C. Burges, and V. Vapnik, "Extracting support data for a given task," in *Proc. of the Int. Conf. on Knowledge Discovery & Data Mining*, Menlo Park, CA, 1995, pp. 251–256.

[4] D. Tax and R. Duin, "Data domain description using support vectors," in *Proc. of the European Symp. on Artificial Neural Networks*, Bruges (Belgium), April 1999, pp. 251–256.

[5] I. Tsang, J. Kwok, and P.-M. Cheung, "Core vector machines: Fast svm training on very large data sets," *Journal of Machine Learning Research*, vol. 6, pp. 363–392, 2005.

[6] I. Tsang, A. Kocsor, and J. Kwok, "Simpler core vector machines with enclosing balls," in *International Conference on Machine Learning (ICML)*, 2007, pp. 911–918.

[7] A. Ypma and R. Duin, "Support objects for domain approximation," in *Proc of the ICANN*, 1998.

[8] D. Tax and R. Duin, "Data descriptions in subspaces," in *Proc. of Int. Conf. on Pattern Recognition*, 2000, pp. 672–675.

[9] M. Breunig, H. Kriegel, R. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *Proc. of the ACM Int. Conf. on Management of Data*, 2000.

[10] F. Angiulli, "Condensed nearest neighbor data domain description," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 10, pp. 1746–1758, 2007.

[11] E. Knorr and R. Ng, "Algorithms for mining distance-based outliers in large datasets," in *Proc. of the Int. conf. on Very Large Databases*, 1998, pp. 392–403.

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

[13] P. Hart, "The condensed nearest neighbor rule," *IEEE Trans. on Information Theory*, vol. 14, pp. 515–516, 1968.

[14] F. Angiulli, "Fast nearest neighbor condensation for large data sets classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 11, pp. 1450–1464, 2007.

[15] N. Littlestone and M. Warmuth, "Relating data compression and learnability," University of California, Santa Cruz, Tech. Rep., 1986.

[16] S. Floyd and M. Warmuth, "Sample compression, learnability, and the vapnik-chervonenkis dimension," *Machine Learning*, vol. 21, no. 3, pp. 269–304, 1995.

[17] F. Angiulli and F. Fassetti, "Dolphin: an efficient algorithm for mining distance-based outliers in very large datasets," *ACM Transactions on Knowledge Discovery from Data*, vol. 3, no. 1, p. Article 4, 2009.

[18] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi, *Complexity and Approximation*. Berlin: Springer-Verlag, 1999.

[19] M. Bellare, S. Goldwasser, C. Lund, and A. Russeli, "Efficient probabilistically checkable proofs and applications to approximations," in *STOC*, 1993, pp. 294–304.

[20] C. B. D.J. Newman, S. Hettich and C. Merz, "UCI repository of machine learning databases," 1998. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html

[21] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. Smola, and R. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[22] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001, software available at `http://www.csie.ntu.edu.tw/∼cjlin/libsvm`.