Contents lists available at ScienceDirect

Pattern Recognition

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

Approximate polytope ensemble for one-class classification

Pierluigi Casale [a,*], Oriol Pujol [b], Petia Radeva [b]

[a] ACTLab, Signal Processing Group, TU Eindhoven, Den Dolech 2, Eindhoven, 5612AZ, The Netherlands
[b] Department de Matemàtica Aplicada i Anàlisi, Universitat de Barcelona, Gran Via 585, Barcelona 08007, Spain

ARTICLE INFO

Article history:
Received 5 November 2012
Received in revised form
22 July 2013
Accepted 4 August 2013

Keywords:
One-class classification
Convex hull
High-dimensionality
Random projections
Ensemble learning

ABSTRACT

In this work, a new one-class classification ensemble strategy called approximate polytope ensemble is presented. The main contribution of the paper is threefold. First, the geometrical concept of convex hull is used to define the boundary of the target class defining the problem. Expansions and contractions of this geometrical structure are introduced in order to avoid over-fitting. Second, the decision whether a point belongs to the convex hull model in high dimensional spaces is approximated by means of random projections and an ensemble decision process. Finally, a tiling strategy is proposed in order to model non-convex structures. Experimental results show that the proposed strategy is significantly better than state of the art one-class classification methods on over 200 datasets.

1. Introduction

In pattern recognition, a particular typology of problems is defined when only data related to one target class are available. These problems are known as one-class classification problems. These classification tasks naturally arise when target data can be effortlessly collected while counterexamples are scarce or difficult to obtain [1]. Typical one-class problems are the prediction of mean time before a machinery failure [1,2] or the problem of banknotes verification [3]. In the former case, examples of non-regular operations can only be found in the presence of cracks and malfunctions. In the latter case, while it is possible to easily provide all the examples necessary to model the examples of valid banknotes, it is hard to define a proper sampling of examples belonging to the counterexample set. Effective one-class classification strategies use density estimation methods and boundary methods to model the target class. Gaussian Model, Mixture of Gaussian Model and Parzen Density Estimation are density estimation methods widely used. Density estimation methods work well when there exists *a-priori* knowledge of the problem at hand and a big amount of data is available. On the other hand, boundary methods only intend to model the boundary of the problem without focusing on the complete description of the underlying distribution. Well known approaches to boundary methods are k-centers [4], Nearest Neighbors [2]) and extensions of Support Vector Machines (SVM) to the one-class setting [5–7]. Support Vectors Data Description (SVDD) [2] represents the state of the art in one-class classification. SVDD computes the minimum hypersphere containing all the data in a multi-dimensional space, providing an elegant and intuitive understanding about the solution of the classification problem. Indeed, many classification problems can be efficiently solved when addressed from the geometrical point of view. In particular, when the geometrical framework is taken into account, the *convex hull*, i.e., the smallest polytope containing the full set of points, may represent an even more general structure than the hypersphere.

The convex hull has always been considered a powerful tool in geometrical pattern recognition [8–11]. Bennet et al. [12,13] showed that there exists a geometrical interpretation of the SVM related to the convex hull, i.e., finding the maximum margin between two classes is equivalent to finding the nearest neighbors in the convex hull of each class when classes do not overlap.

Nevertheless, using the convex hull in real applications is limited by the fact that its computation in high dimensional spaces has an extremely large computational cost. Although advanced solutions have been proposed to overcome the limitation of building the convex hull in high dimensional spaces [14,15], there exists a wide range of pattern recognition problems where the use of the convex hull is still unsuitable specially when limited computational resources are considered. Hence, theoretical methodologies for building a convex hull in constrained scenarios are worth to be investigated. In addition, detecting whether a point lies inside or outside of the convex hull in high dimensional spaces still remains an open problem. Random projections and high dimensional geometry lie at the heart of several important approximation algorithms [16]. Random projections have been widely used in pattern recognition applications as a tool for dimensionality

[*] Corresponding author. Tel.: +31 402473547.
E-mail address: p.casale@tue.nl (P. Casale).

0031-3203/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.patcog.2013.08.007

reduction [17–19]. The random projections technique is based on the idea that high dimensional data can be projected into a lower dimensional space without significantly losing the structure of the data. This data structure preservation has been proved by Johnson and Linderstrauss (JL) [20] and it is ensured with high probability if data are projected into a destination space having dimensionality proportional to the logarithm of the cardinality of the dataset. The capability to reduce the dimensionality of the problem without a significant computational effort and loss of structure allows to create very simple and powerful learning techniques. The simplest learning algorithm based on random projections [21] consists in two steps: project the data in a random subspace and run the learning algorithm in that space. Some works suggest that using random projections is equivalent to using the '*kernel trick*' [22]. Other works [23] suggest that random projections can help in the feature selection process and can also provide specific insights in the construction of large margin classifiers. Moreover, even picking a random separator on data projected down to a line, there is a reasonable chance to get a valid weak hypothesis.

In this work,[1] three main contributions are proposed in the context of one-class classification:

1. The geometric structure of the convex hull is proposed for modeling the boundary of the one-class classification problem. Shrunken or enlarged versions of the baseline convex hull of the training data are used to avoid over-fitting and to find the optimal operating point of the classifier. These versions are called *extended convex polytopes* and their growth is governed by a parameter α. Using this model, a new data point is said to belong to the target class if it lies inside the extended convex polytope. The creation of the extended convex polytope is limited by the fact that its computation is unfeasible in high dimensional spaces.
2. This limitation is circumvented by approximating the D-dimensional expanded convex polytope decision by an ensemble of decisions in very low-dimensional spaces $d \ll D$. This new geometric structure is called *approximate convex polytope decision ensemble*. In these low-dimensional spaces, computing the convex hull and establishing whether points lie inside the geometric structure are both well known problems having very computationally efficient solutions [25,26]. In this work, the effect of projecting and constructing the ensemble using two-dimensional and one-dimensional random spaces is analyzed. As a result, a very efficient and powerful one-class classifier is obtained.
3. However, many real world problems are not well modeled using a convex polytope. Thus, an ensemble of convex polytopes is finally proposed in order to approximate the non-convex boundary of the one-class classification problem. The algorithm is based on a tiling strategy and each convex polytope is approximated by the *approximate convex polytope decision ensemble.*

All the proposed one-class methodologies are validated on a set of 5 toy problems with different cardinalities, 82 one-class problems derived from the UCI machine learning repository, 15 problems related to mobile user verification from walking patterns and 100 text categorization datasets. The paper is organized as follows. In Section 2, the proposed one-class classification method based on the convex hull is described in detail. Its extension for modeling non-convex boundaries is described in Section 3. In Section 4, the validation protocol is described and in Section 5, experimental results are presented. In Section 6 some methodological topics of interest are discussed. In particular, discussions on the number of random projections needed by the proposed methodology, the effect of the expansion parameter and the computational complexity in comparison to state of the art one-class classification methods are addressed. Finally Section 7 concludes the paper.

2. Approximate polytopes decision ensemble for one-class classification

One-class classification can be performed by modeling the boundary of the set of points defining the problem. If the boundary encloses a convex area, then the convex hull, defined as the minimal convex set containing all the training points, provides a good general tool for modeling the target class. The *convex hull* of a set $C \subseteq \mathbf{R}^n$, denoted **conv** C, is the smallest convex set that contains C and is defined as the set of all convex combinations of points in C:

$$\mathbf{conv}\ C = \left\{ \theta_1 x_1 + \cdots + \theta_m x_m | x_i \in C, \theta_i \geq 0, \forall i; \sum_i \theta_i = 1 \right\}.$$

In this scenario, the one-class classification task is reduced to the problem of knowing if test data lie inside or outside the hull. Although the convex hull provides a compact representation of the data, outliers may lead to shapes of the convex polytope not corresponding to the desired model and a decision using this structure is prone to over-fitting. In this sense, it is useful to define a parameterized set of convex polytopes associated with the original convex hull of the training data. This set of polytopes are shrunken/enlarged versions of the original convex hull governed by a parameter α. This expansion parameter allows to calculate the Receiver Operating Characteristic (ROC) curve and set the optimal operating point for the final one-class classifier. Given the set $C \subseteq \mathbf{R}^n$, the *extended convex polytope* is defined with respect to the center point c and expansion parameter α. Vertices of the convex polytope are defined as in

$$v_\alpha : \left\{ v + \alpha \frac{(v-c)}{\| v-c \|} | v \in \mathbf{conv}\ C \right\} \tag{1}$$

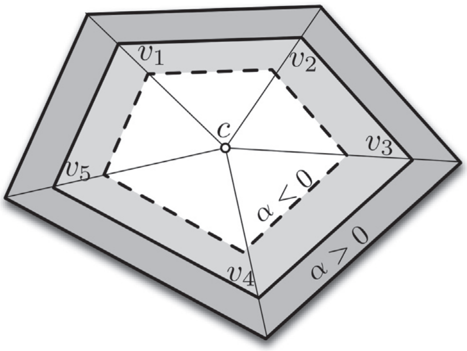

Fig. 1. Illustration of the expanded convex polytope in the 2D space. The central light gray convex polygon represents the original convex hull with vertices $\{v_i, i = 1 \ldots 5\}$. The outer dark gray polygon corresponds to the enlargement of the original convex hull using $\alpha > 0$. The inner white polygon corresponds to a shrunken version of the convex hull using $\alpha < 0$.

[1] The current paper improves and extends our seminal work [24]. The previous paper shares with the present work the basic idea of using random projections for reducing the complexity of building the convex hull in a high dimensional space. In the current paper the methodology is revised and new contents are introduced, i.e. the use of the expansion factor and a methodology for handling non-convex shapes as well as a large validation section.

The parameter α defines a constant shrinking $(-\|v-c\| \le \alpha \le 0)$ or enlargement $(\alpha \ge 0)$ of the convex structure with respect to the c. If $\alpha = 0$ then $v_0 = \textbf{conv}\ C$. Fig. 1 illustrates a shrunken and enlarged expanded convex polytopes. Two fundamental limitations exist in the suggested approach: Both the task of computing the *extended convex polytope* and testing if a point lies in its interior in high dimensional spaces is computationally unfeasible. In the following section, a solution to these problems is proposed.

2.1. Approximate convex polytope decision ensemble

The creation of a high-dimensional convex hull is computationally hard. The cost of computing a D-dimensional convex hull on N data examples is $\mathcal{O}(N^{\lfloor D/2 \rfloor + 1})$. This cost is prohibitive in time and memory and, for a classification task, only checking if points lie inside the multi-dimensional structure is needed. The *approximate convex polytope decision ensemble (APE)* consists in approximating the decision made by the *extended convex polytope* in the original D-dimensional space by aggregating a set of τ randomly projected decisions made on low-dimensional spaces. The approximation is based on the observation that the vertices defining a convex polytope in a low-dimensional projection of the data set correspond to a subset of the projected vertices belonging to the convex polytope in the original D-dimensional space. With this consideration in mind, the *decision rule* can be defined as follows: given a set of τ randomly projected replicas of the training set, a point does not belong to the modeled class if and only if there exists at least one case in which the point lies outside of the projected convex polytope.

In Fig. 2, the graphical explanation of the method is shown. A three-dimensional convex polytope and a test point lying outside the hull are presented. In the bottom, three random projection planes are displayed. It should be noted that a testing point outside the original structure might appear inside one or more projections.

2.2. The expansion factor in random spaces

The decision that one point belongs to the class is made by considering the *extended convex polytope*. This structure is a shrunken or expanded version of the convex hull governed by parameter α. The *approximate convex polytope decision strategy* relies in creating the expanded polytope in a low-dimensional space. Since the projection matrix is created at random, the norm of the original space is not preserved in the resulting space. Hence, a constant value of the parameter α in the original space corresponds to a set of values γ_i in the projected one. As a result, the low-dimensional approximation of the expanded polytope is defined by the set of vertices as

$$\overline{v}^{\alpha} : \left\{ \overline{v_i} + \gamma_i \frac{(\overline{v_i} - \overline{c})}{\| \overline{v_i} - \overline{c} \|} \right\}, \quad i = 1 \dots N \tag{2}$$

where $\overline{c} = Pc$ represents the projection of the center c given a random projection matrix P, $\overline{v_i}$ is the set of vertices belonging to the convex hull of the projected data and γ_i is defined as

$$\gamma_i = \frac{(v_i - c)^T P^T P (v_i - c)}{\| v_i - c \|} \alpha \tag{3}$$

where v_i is the ith vertex of the convex hull in the original space.

It is worthy of attention that there exist different expansion factors for each of the vertex $\overline{v_i}$ belonging to the projected convex hull. Fig. 3 illustrates the difference between the constant expansion of the original polytope and the different gamma parameters in the projected counterpart.

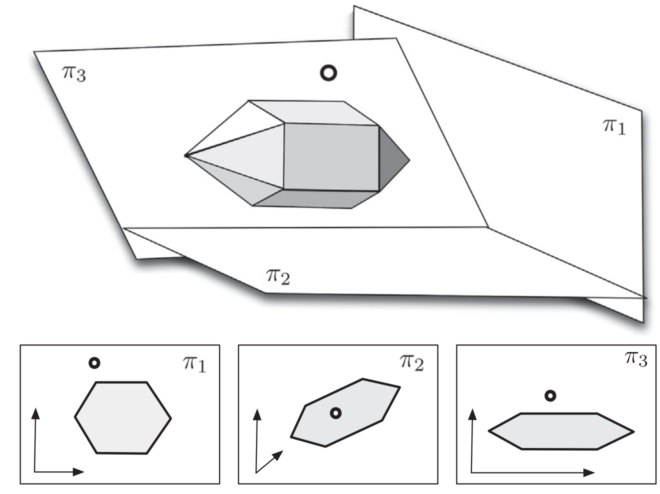

Fig. 2. Illustration of the approximate convex polytope decision strategy: a point does not belong to the modeled class if there exists at least one projection where the point is outside the projected convex polytope.

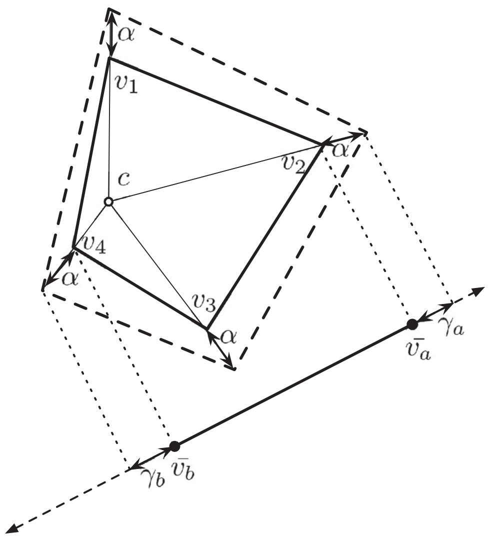

Fig. 3. Illustration of an expanded convex polytope in the original 2D space and the corresponding projected convex polytope in a 1D space. Observe that though the convex polytope is expanded by a constant value α in the original space, different expansion parameters γ_i arise in the projected space.

2.3. Approximate convex polytope decision ensemble learning algorithms

The steps needed for learning and testing the proposed approach are described in Algorithms 1 and 2, respectively. Both algorithms require defining the number of projections τ. In the learning phase, at each iteration, a random matrix is created. Then, the training set is projected into the space spanned by the random projection matrix. Finally, the vertices of the convex hull of the projected data set are found. $\mathbf{R}^2$ and $\mathbf{R}$ have been used as final destination spaces. Although this solution may appear to contradict the JL Lemma, it must be noted that APE does not directly rely on the pair-wise distance preservation property but on a relaxed version of it since only whether a point lies inside the polytope is needed. Building a convex hull in $\mathbf{R}^2$ and $\mathbf{R}$ and checking if a point lies inside or outside of it are well known problems in computational geometry with very efficient solutions.

Algorithm 1. Approximate convex polytope decision ensemble learning algorithm.

```
Input:  Training set $C=\{x_i\}\in\mathbf{R}^D$, $i=1...N$ , with $D$ the dimensionality of each data example $x_i$;
        Number of projections $\tau$
Output: The model $\mathcal{M}$ composed of $\tau$ projection matrices and their respective convex hull vertices.
$\mathcal{M}=\varnothing$;
$c=\frac{1}{N}\sum_i x_i,\ \forall x_i\in C$;
for $t=1..\tau$ do
    $P_t\sim\mathcal{N}(0,1)$ % Create a normal random projection matrix;
    $C_t:\{P_t x|x\in C\}$ % Project data onto the low dimensional random space;
    $\{v_i\}_t=\mathbf{conv}\ C_t$ % Find the convex hull and return the set of vertices;
    $\mathcal{M}=\mathcal{M}\cup(P_t,\{v_i\}_t)$ % Store the set of vertices associated to the convex hull in the projected space and the projection matrix;
end
```

Algorithm 2 describes the test procedure. At each iteration t of the algorithm, the test point is projected into the space spanned by the t-th random projection matrix. Then, each vertex of the t-th convex hull computed at the training step is expanded by its corresponding gamma value defined in Eq. (3). Given the set of vertices of the expanded convex polytope in the low-dimensional space as defined in Eq. (2), it is possible to check if the test point lies inside the projected polytope. A point approximately belongs to the model if it lies inside all the t projected polytopes. Many algorithms may be used to check if a data point is inside a 2-D convex polytope like, for instance, the ray casting algorithm [27]. In the 1-D case, the convex polytope is reduced to a single line segment. Hence, checking if a point lies on the segment requires at most two comparisons.

Algorithm 2. Approximate convex polytope decision ensemble testing algorithm.

```
Input: A test point $x\in\mathbf{R}^D$;
       The model $\mathcal{M}$; The parameter $\alpha$
Output: $Results\in\{INSIDE, OUTSIDE\}$
Results=INSIDE;
for $t=1..\tau$ do
    $\overline{x_t}=P_t x$ % Project data.;
    $v_t^\alpha:\left\{v_i+\gamma_i\frac{(v_i-c)}{\|v_i-c\|}\middle|v_i\in\{v\}_t\right\}$ % Find the expanded convex
    polytope in the low dimensional space;
    if $\overline{x_t}\notin\mathbf{conv}\ v_t^\alpha$ then
        Results = OUTSIDE
        Break
    end
end
```

3. Modeling non-convex distributions

The main drawback of APE is that the boundary of training data may not be well modeled by a convex polytope. Hence, an extension of the algorithm to cope with non-convex boundaries is also proposed. The underlying idea of this extension is to divide the non-convex boundary into a set of convex problems. Then, each of the convex problems is solved by means of the *approximate convex polytope decision ensemble*. The result of this process is another ensemble algorithm called *non-convex APE (NAPE)*. Algorithm 3 describes the pseudo-code for creating this decomposition. Starting with a random point c, the set of points inside a ball centered at c with radius r are considered in the first step. Fig. 4(a) illustrates this first step. The black point corresponds to the first center c. Around c a D-ball of radius r is laid and the convex hull of the set of points inside the ball is computed. This first D-dimensional convex hull is approximated using the APE technique. For each projection, the set of vertices conforming the convex hull in the reduced space is back projected[2] into the original space. A graphical explanation is provided in Fig. 4(a). These points are added to a list of new candidate centers. In the following iteration, the algorithm removes one of the possible candidate centers of the list and repeat the process as long as there are still training points not covered by any of the created convex hulls. Fig. 4(b) shows the second iteration of this process.

Algorithm 3. Non-convex approximate decomposition algorithm.

```
Input: Training set $C\in\mathcal{R}^D$, with $D$ the number of features;
    Number of Projections $\tau$
Output: Model $\mathcal{M}$ composed of several convex models defined
    by Algorithm 1
$L=\varnothing$;
Pick a random training point $c_{start}$;
$L=L\cup\{c_{start}\}$ % Initialize the list of possible centers with the
    first random element;
Set all data points $x\in C$ to the value not visited;
while $\exists$ x with value not visited do
    if $L=\varnothing$ then
        Pick a random a training point with attribute not visited,
         $p\in C$
        $L=L\cup\{p\}$
    end
    $p=$ first $(L)$ % Remove the first element of the list;
    $C_i:\{x\in C|\ \|x-p\|\le r\}$ % Find the set of points to be modeled
    with a convex polytope in this iteration;
    Set all data points $x\in C_i$ to the value visited;
    $\mathcal{M}_i=$ TrainAPE$(C_i,T)$ % Find the approximate model associated
    to the selected set using Algorithm 1;
    $\mathcal{M}=\mathcal{M}\cup\mathcal{M}_i$ % Add the new convex model to the final
    model set.;
    $L=L\cup\{v_i\in C|\overline{v_i}\in\mathcal{M}_i\}$ % Add the points of $C$ corresponding to
    vertices of the projected convex hulls of the current
    model $\mathcal{M}_i$;
end
```

[2] It should be noted that this step only needs to keep track of the points that are selected in the low-dimensional space and correspond to the vertices of the projected convex hull.

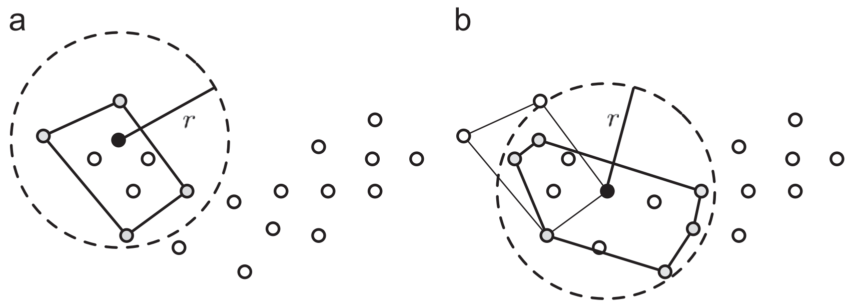

Fig. 4. Approximation of bi-dimensional banana-shaped dataset using NAPE.

In the same way the extended convex polytope is defined, an extended non-convex polytope is also defined. Observe that due to the definition of the extended convex polytope (Eq. (1)) and its approximation in Eq. (2), taking a global α value for all the convex polytopes suffices to expand the whole non-convex shape by a constant value α. In order to test if a point lies inside the non-convex model, one must check if it lies *inside of at least one* of the constituting approximate polytope ensemble. Otherwise, it is said to be outside.

4. Validation methodology

The APE and NAPE algorithms are compared with several state-of-the-art one-class pattern recognition methods. APE is created by projecting data down to both one-dimensional (APE-1) and two-dimensional (APE-2) spaces. The methods are validated on three different typology of problems using standard performance evaluation metrics.

Datasets. APE and NAPE are validated on three different typologies of problems:

- *Artificial datasets.* These datasets are used to evaluate the behavior of the methods with respect to non-convexity. Normal distribution, banana-shaped, S-shaped, toroidal and 3-shaped distributions are used. Non-target points are generated using the procedure described in [28]. The radius of the generating hypersphere is set to 2. For each problem, 10-fold cross validation is performed on ten randomly created sets and results averaged.
- *UCI datasets*. 82 real one-class problems derived from 23 UCI machine learning repository datasets are considered. For each dataset, one-class classification problems are obtained considering one of the classes as target and all the other classes as non-targets. The list of datasets used is shown in Table 1. Each problem is evaluated using 5-fold stratified cross-validation on 10 different permutations of the data for a total of 50 experiments per problem.The final result is obtained averaging all the results. Due to its high computational complexity in training, a reduced set of problems are used for comparison with SVDD. This subset D^* is composed by problems in Table 1 marked with a star. The choice of these datasets is based on the fact that all these datasets, except Tic-tac-toe, have cardinality less than 500 elements.
- *Real world classification problems*. Two one-class modeling tasks related to user verification from walking patterns and text categorization have been considered as real-world scenarios for validating the proposed methodology.
The user verification task must be viewed as a prototypical example of one-class classification task due to the difficulty to obtain all the possible counterexamples of unauthorized users. In this application, a commercial Android smart-phone is used to verify users *in the wild*, such as in rough terrains or in adversarial conditions, in crowded places or with obstacles using smartphone sensors data from 15 testers. Data associated to each user present high non-convexity. This is attributed to different user's walking speed, terrain conditions and obstacles. A 36-dimensional feature vector is computed from the triaxial acceleration data. Given a time frame, the extracted features are the average and standard deviation of the difference between pairs of consecutive peaks, difference between the maximum and the minimum values of consecutive peaks for acceleration and its derivative. The amount of data collected adds to 11 M examples.
Text categorization has been considered as the second classification problem. The Technion Repository of Text Categorization Datasets [29] provides a large number of diverse collections for use in text categorization research. The collection contains many different datasets automatically acquired. In this experiment, the TechTC-100 collection containing 100 datasets has been used. Each dataset consists of a pair of categories with 150–200 documents. The number of features vary depending on the dataset. The dimensionality of the problems is considerably high, ranging from $\sim 13,000$ to $\sim 30,000$. In order to have one-class classification problems, for each dataset the first class to be categorized has been considered as target class.

Table 1
List of datasets: for each class, a one-class classification problem is considered.

Dataset	Classes	Dataset	Classes
Balance-scale	3	*Ionosphere	2
Breast-cancer	2	*Iris	3
*Breast tissue	6	*Monks 1	2
*Bupa	2	*Monks 2	2
Car	4	*Monks 3	2
Cardiotogography	10	*Statlog heart	2
*CB-sonar	2	Statlog Seg	7
*CB-vowel	2	*Teaching	3
Contraceptive	3	*Tic-tac-toe	3
*Glass	3	Vowel	11
*Haberman	2	*Wine	3
*Hayes–Roth	3		

Methods. The proposed approaches are compared to different density and boundary methods for one class classification. Gaussian model (Gauss), Mixture of Gaussians (MoG), Parzen Density Estimation (PDE) have been chosen as density methods. The boundary methods selected are *k*-centers (kC), *k*-nearest neighbor (kNN), *k*-means (kM) [30], Minimum Spanning Trees (MST) [31] and SVDD.[3] These methods are the most commonly used in the literature for one-class classification problems.

[3] SVDD has been shown to behave better than one-class SVM classifiers over a wide range of problems [2].

Table 2
Comparative results for artificial datasets: mean AUC and standard deviation. In bold font, the method with best performance is highlighted.

Method	Normal	Banana	Esse	Tre	Toro
Gauss	0.963 ± 0.006	0.941 ± 0.010	0.951 ± 0.010	0.954 ± 0.008	0.912 ± 0.010
MoG	0.963 ± 0.006	0.967 ± 0.005	0.973 ± 0.006	$\mathbf{0.971} \pm 0.007$	0.941 ± 0.009
PDE	0.963 ± 0.006	0.967 ± 0.005	0.973 ± 0.006	0.970 ± 0.006	0.941 ± 0.008
kNN	0.949 ± 0.009	0.956 ± 0.008	0.966 ± 0.007	0.965 ± 0.006	0.930 ± 0.009
kM	0.958 ± 0.008	0.961 ± 0.006	0.969 ± 0.006	0.969 ± 0.007	0.934 ± 0.009
kC	0.961 ± 0.007	0.955 ± 0.008	0.961 ± 0.007	0.963 ± 0.008	0.924 ± 0.012
MST	0.942 ± 0.012	0.953 ± 0.009	0.964 ± 0.007	0.964 ± 0.006	0.927 ± 0.010
SVDD	0.959 ± 0.006	0.955 ± 0.010	0.954 ± 0.007	0.952 ± 0.007	0.911 ± 0.013
APE-1	0.921 ± 0.029	0.872 ± 0.055	0.931 ± 0.027	0.881 ± 0.056	0.911 ± 0.023
APE-2	0.960 ± 0.013	0.880 ± 0.064	0.962 ± 0.010	0.955 ± 0.015	0.831 ± 0.050
NAPE	$\mathbf{0.966} \pm 0.008$	$\mathbf{0.975} \pm 0.005$	$\mathbf{0.977} \pm 0.006$	0.960 ± 0.004	$\mathbf{0.957} \pm 0.012$

Table 3
Counts of wins, ties and losses obtained on D^*. In bold font, methods that are pairwise statistically different at 95% significance level.

Method	Gauss	MoG	PDE	kNN	kM	kC	MST	SVDD	APE-1	APE-2	NAPE
Gauss	0/0/0	21/4/17	22/3/17	19/3/20	23/2/17	23/0/19	19/1/22	19/3/20	25/0/17	17/0/25	17/0/25
MoG	17/4/21	0/0/0	22/3/17	14/3/25	**27/2/13**	23/0/19	18/1/23	19/3/20	22/0/20	13/0/29	12/0/30
PDE	17/3/22	17/3/22	0/0/0	11/3/28	**27/2/13**	25/0/17	13/1/28	14/3/25	22/0/20	13/0/29	12/0/30
kNN	20/3/19	25/3/14	**28/3/11**	0/0/0	**29/2/11**	**29/0/13**	12/1/29	16/4/22	23/0/19	16/0/26	14/0/28
kM	17/2/23	13/2/27	13/2/27	11/2/29	0/0/0	24/0/18	9/0/33	14/2/26	24/0/18	11/0/31	11/0/31
kC	19/0/23	19/0/23	17/0/25	13/0/29	18/0/24	0/0/0	15/0/27	15/0/27	**30/0/12**	16/0/26	13/0/29
MST	22/1/19	23/1/18	**28/1/13**	**29/1/12**	**33/0/9**	**27/0/15**	0/0/0	17/1/24	22/0/20	14/0/28	13/0/29
SVDD	20/3/19	20/3/19	25/3/14	22/4/16	**26/2/14**	**27/0/15**	24/1/17	0/0/0	25/0/17	16/0/26	15/0/27
APE-1	17/0/25	20/0/22	20/0/22	19/0/23	18/0/24	12/0/30	20/0/22	17/0/25	0/0/0	12/2/28	10/1/31
APE-2	25/0/17	**29/0/13**	**29/0/13**	26/0/16	**31/0/11**	26/0/16	**28/0/14**	26/0/16	**28/2/12**	0/0/0	0/30/12
NAPE	25/0/17	**30/0/12**	**30/0/12**	**28/0/14**	**31/0/11**	**29/0/13**	**29/0/13**	**27/0/15**	**31/1/10**	**12/30/0**	0/0/0

Evaluation metrics. For artificial and UCI datasets, the Area Under the ROC Curve (AUC) is used as the performance measure for comparing different classification methods [32]. For real world classification problems, precision, recall and F-measure have been used to quantify the classification performances and explicitly show values resulting from parameter selection. For each run of cross-validation, the performance measures have been computed on the testing set once the operating point on the ROC curve has been chosen on a validation set.

Parameters setting. In order to compute the Area Under the Curve, each ROC curve is evaluated on 50 points varying the appropriate threshold of the classification method or the α parameter in APE-1, APE-2 and NAPE. In the case where additional parameters must be set, and for choosing the threshold values in the rest of the performance measures, i.e. precision, recall and F-measure, 2-fold cross-validation on the remaining training set for each iteration of the global stratified cross-validation process is used. The k value in k-NN, k-means, k-centers is chosen in the range from 1 to 10. The regularization parameter of SVDD is set using an outlier rejection rate of 0.1. For APE-1, APE-2 and NAPE, the number of projections has been set arbitrarily to 1000. Proper discussion regarding the number of projections needed is reported in Section 6.

5. Experimental results

In this section, results obtained on artificial and UCI datasets are described in Sections 5.1 and 5.2, respectively. In Section 5.3, results related to user verification and text categorization are reported.

5.1. Results on artificial datasets

Experiments on artificial datasets have been performed on datasets with 500, 750 and 1000 data-points. Due to its high computational complexity in training, SVDD has been evaluated only on a dataset with 500 data-points. In Table 2, mean AUC and standard deviations are reported. APE-2 generally achieves better results than APE-1 for the same number of projections. NAPE always performs better than APE with significant improvement in the performances when the data set can be modeled using highly non-convex models, i.e. banana and toroidal datasets. Comparing with the rest of the methods, NAPE performs better than all the other methods on all the artificial problems taken into account, closely followed by MoG.

5.2. Results on UCI datasets

Results obtained on D^* and D are reported in Tables 3 and 4. Each element of the table represents the number of times the methods reported on the rows wins, ties and loses with respect to the method reported on the column. In order to evaluate the statistical significance at 95% significance level of the results, a z-test [33] is applied. The results show that NAPE is statistically better than almost all the methods on D^*. On a bigger number of problems, NAPE is in most cases significantly better at 95% significance level and always significantly better at 90% significance level. This fact is shown in Table 4. It should be noted that, there exists a wide range of problems where APE-2 and NAPE tie. This fact derives directly from the definition of the NAPE methodology that behaves like APE-2 when the problem is correctly modeled with just one convex model. APE-1 does not provide significant results when compared with the other methods.

5.3. Results on user verification

Average AUC and standard deviation on the user verification problem are shown in Fig. 5. Both APE-2 and NAPE achieve the best results. At the same time, NAPE displays very small standard deviation with respect to the rest of the classifiers and the highest

Table 4
Counts of wins, ties and losses obtained on *D*. In bold font, methods that are pairwise statistically different at 95% significance level.

Method	Gauss	MoG	PDE	kNN	kM	kC	MST	APE-1	APE-2	NAPE
Gauss	0/0/0	25/21/36	28/18/36	26/18/38	**49/13/20**	**50/4/28**	28/9/45	**50/0/32**	31/0/51	30/0/52
MoG	36/21/25	0/0/0	34/18/30	27/18/37	**54/13/15**	**49/4/29**	34/9/39	51/0/31	39/0/43	36/0/46
PDE	36/18/28	30/18/34	0/0/0	20/22/40	**54/13/15**	**50/4/28**	25/13/44	**50/0/32**	38/0/44	35/0/47
kNN	38/18/26	37/18/27	40/22/20	0/0/0	**54/13/15**	**54/4/24**	23/16/43	**51/0/31**	41/0/41	37/0/45
kM	20/13/49	15/13/54	15/13/54	15/13/54	0/0/0	43/3/36	16/4/62	39/0/43	24/0/58	20/0/62
kC	28/4/50	29/4/49	28/4/50	24/4/54	36/3/43	0/0/0	27/4/51	48/0/34	32/0/50	27/0/55
MST	45/9/28	39/9/34	**44/13/25**	**43/16/23**	**62/4/16**	**51/4/27**	0/0/0	47/0/35	36/0/46	32/0/50
APE-1	32/0/50	31/0/51	32/0/50	31/0/51	43/0/39	34/0/48	35/0/47	0/0/0	13/20/49	10/5/67
APE-2	**51/0/31**	43/0/39	44/0/38	41/0/41	**58/0/24**	**50/0/32**	46/0/36	**49/20/13**	0/0/0	0/51/31
NAPE	**52/0/30**	46/0/36	47/0/35	45/0/37	**62/0/20**	**55/0/27**	**50/0/32**	**67/5/10**	**31/51/0**	0/0/0

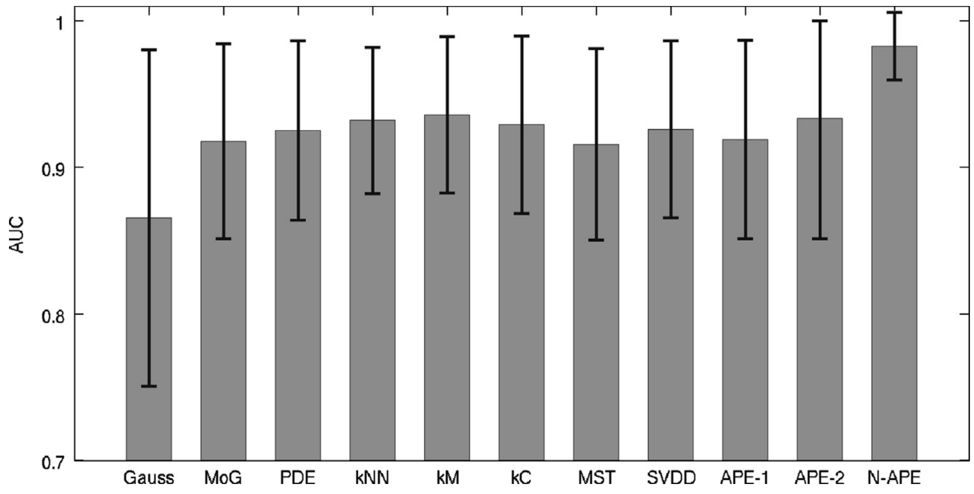

Fig. 5. User verification problem: AUC obtained on different one-class classifiers.

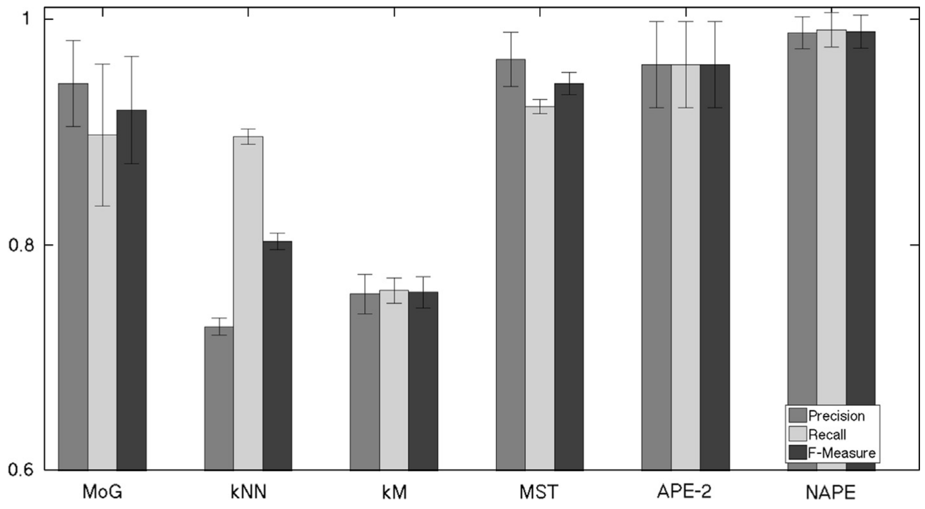

Fig. 6. User verification problem: precision, recall and F-measure obtained on different one-class classifiers.

classification performance. *k*-means is the method that, after NAPE and APE-2, provide the best AUC. APE-1 does not provide significant good results when compared with other one-class classification methods. Precision, recall and F-measure obtained on the user verification problem are shown in Fig. 6. NAPE provides the best performance for all the considered metrics. APE-2 achieves the second best general performance, though it has precision lower than MST because it can only model convex shapes. A pairwise comparison of MoG, MST, *k*-means, and *k*-NN against APE-2 and NAPE is provided in Figs. 5 and 6. Figs. 8 and 9 display scatter plots of the F-measure of one method compared with another. Each problem is depicted as a point in the plot. Each point above the separation line indicates that the F-measure performance on that problem is better by using the method reported on the *Y*-axis than using the method on the *X*-axis. Few problems have been better modeled by MoG and MST when compared with APE2. On the other hand, NAPE is able to model all the user verification problems better than the rest of the methods.

5.4. Results on text categorization

Error bars for average precision, recall and F-measure for the text categorization problems are reported in Fig. 7. The experimental results obtained on these very high dimensional datasets with scarce data availability show that NAPE converges to APE-2. This is expected since the correct modeling of a discrete set of points by means of convex models is a matter of resolution and data point density. Although APE2-NAPE have lower recall when compared to Gauss, *k*-means and MST, the method always shows higher precision and F-measure. Scatter plots for the F-measure comparing APE2-NAPE with the rest of the most significant methods are shown in Fig. 10. Observe that the number of problems where APE2-NAPE outperform the other technique is significantly high.

6. Discussions

In this section, the number of random projections needed to approximate the original multi-dimensional convex hull, the role of the expansion parameter α in the classification process and the computational complexity of the methods are discussed. These parameters define the operating regime of the proposed methodologies.

6.1. Number of projections

The number of projections in APE-1, APE-2 and NAPE has been arbitrarily set to 1000. Experiments show that using 1000 projections, the AUC obtained generally converges to a maximum level of performances. However, using a lower number of projections, high level of performance can still be achieved. In Table 5, the minimum, mean and maximum number of projections needed to reach 90%,

Table 5
Number of projections: minimum value, mean value and maximum value.

Method	90%	95%	99%
APE-1	1/31/69	2/63/255	14/266/715
APE-2	1/15/40	1/51/175	1/167/522
NAPE	1/11/33	1/38/131	1/141/463

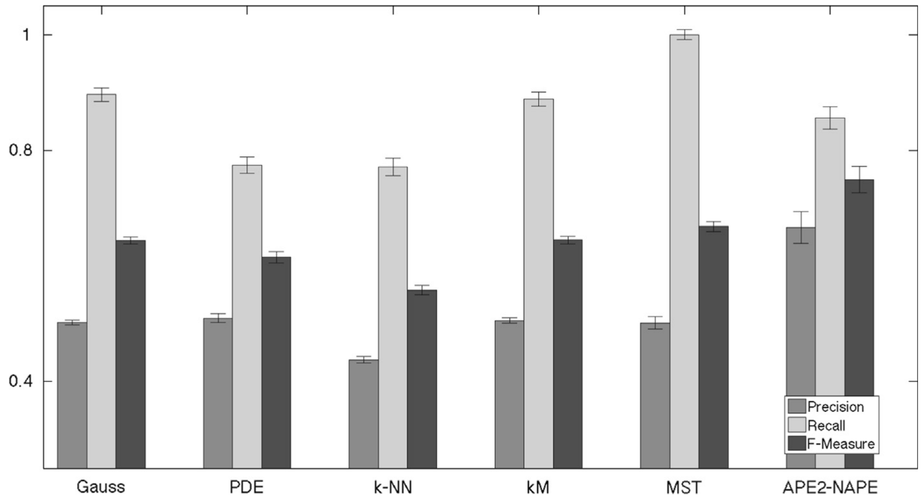

Fig. 7. Text categorization problem: precision, recall and F-measure obtained on different one-class classifiers.

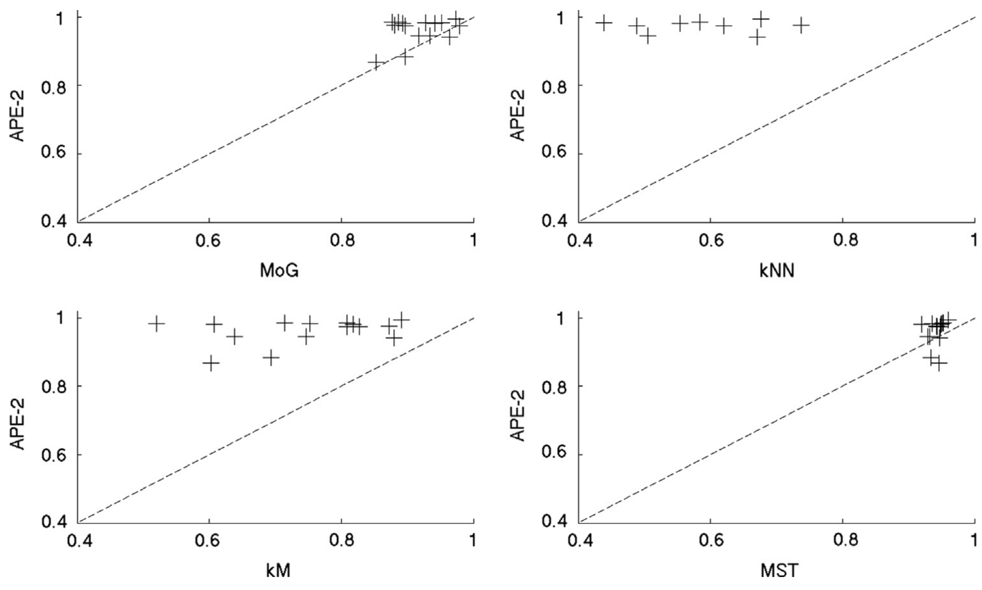

Fig. 8. User verification problem: F-measure obtained by APE2 and other one-class classifiers.

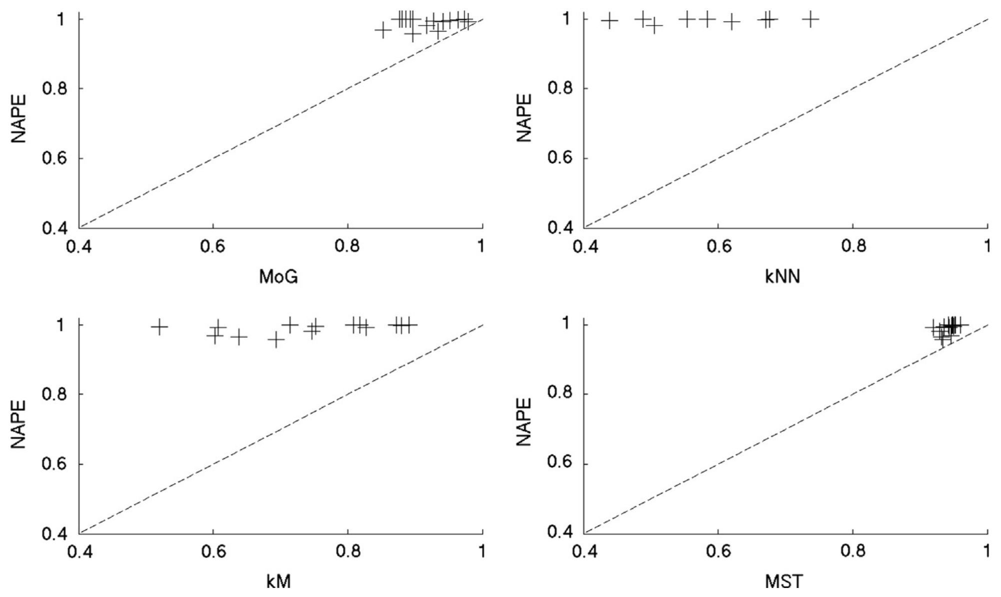

Fig. 9. User verification problem: F-measure obtained by NAPE and other one-class classifiers.

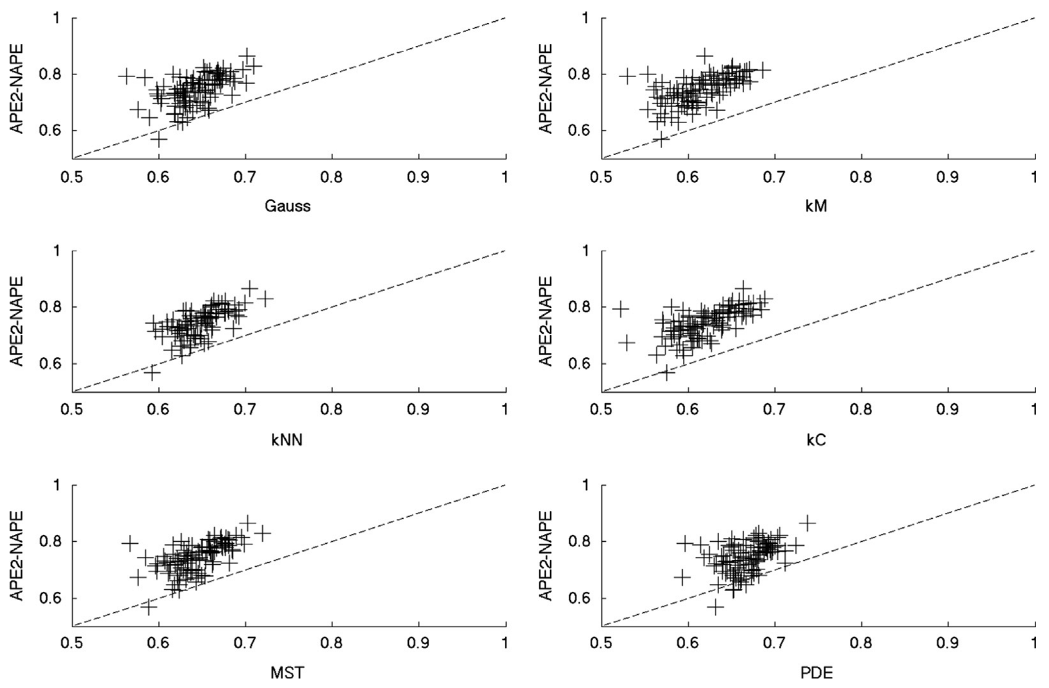

Fig. 10. Text categorization problem: comparative F-measure between NAPE-APE2 and other one-class classifiers.

95% and 99% of the highest performance level computed on the one-class problems derived from UCI datasets are reported.

APE-2 and NAPE need a lower number of projections with respect to APE-1. Moreover, APE-2 and NAPE approximately used the same number of projections for reaching the maximum performance that is much lower than 1000, the number of projections used in the experiments with the UCI datasets. In order to reach 90% of the maximum value, APE-2 and NAPE need a very small number of projections. It is worth to note that, though the number of projections for reaching 90% and 95% of the performance is very small, a considerable number of projections is needed for reaching 99% of the maximum performance. Nevertheless, a mean value of 170 projections is needed for reaching the maximum performance with APE-2 and NAPE.

6.2. *Expansion parameter*

When non-target data are close to the boundary of the convex hull, the number of projections needed for checking if those points are inside the polytope might be very high. However, there is a synergistic effect that allows to mitigate this drawback. The number of projections needed for checking if a point lies inside of the polytope depends on the relative distance of the point to be checked d and the size of the polytope. If a ball of radius R inscribing the polytope is considered, the number of projections depends on $\propto e^{-d/R}$. It should be noted that a negative value of the expansion factor α shrinks the polytope. This fact has the double effect of increasing the distance of the point to the polytope by α and reducing the size of the polytope. As a result, the relative

Table 6
Training and testing computational complexity for each analyzed method.

Method	APE1	APE2	NAPE	SVDD	k-NN	k-means	MoG	MST
Training	$\tau(N(d+1))$	$\tau(Nd+N\log N)$	$\tau K(Nd+N\log N)$	N^3d	–	τKNd	Nd^3K	$N^2\log N$
Test	$\tau(d+2)$	$\underset{M<N}{\tau(d+M)}$	$\underset{M<N}{\tau K(d+M)}$	$\underset{M=\lvert SV\rvert<N}{Md}$	Nd	Kd	Kd^2	$>Nd$

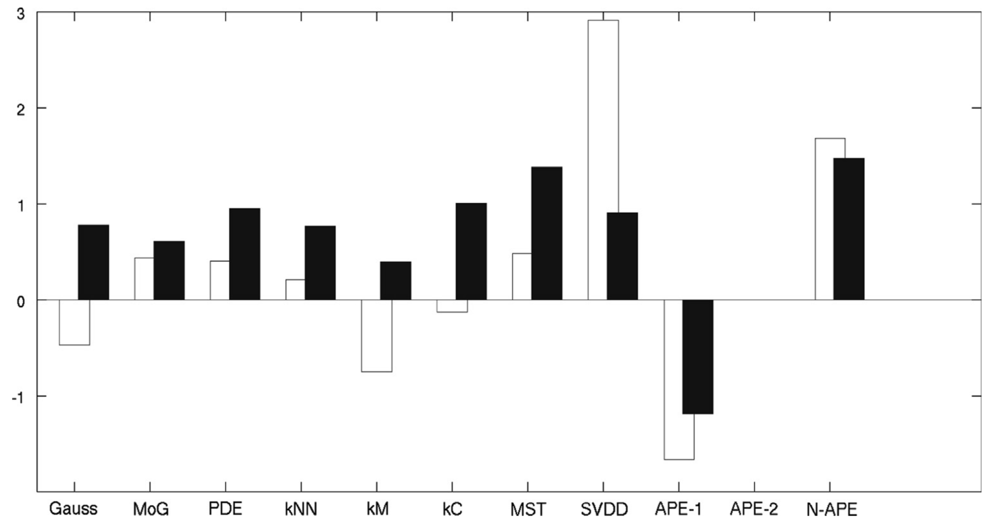

Fig. 11. Comparison of training (white) and testing (black) time relative to APE-2 times. Values on Y-axis are reported in the logarithmic scale.

distance becomes $(d+\alpha)/(R-\alpha)$. Thus, reducing the value of the α parameter reduces drastically the number of projections needed at the cost of possibly changing the operating point.

6.3. Computational complexity

The proposed strategy has great advantages from both computational and memory storage points of view when compared with computing the multi-dimensional convex hull. In the following analysis, the number of examples is denoted by N; space dimension, d; iterations in k-means and number of projections in APE variants, τ; components in k-NN, k-means, MoG and NAPE, k; the number of support vectors and the number of vertices in APE variants, M, where $M \ll N$. Table 6 shows the different training and testing complexities for each of the methods.[4] APE1 and APE2 have two different parts: First, the random projection the from original space to a one- or two-dimensional spaces and, second, the creation of the convex hull in low dimensionality. The complexity for building the convex hull in 1D is $\mathcal{O}(N)$ and in 2D is $\mathcal{O}(N\log N)$. The projection of N data samples is $\mathcal{O}(Nd)$. Thus the complete computational complexity is given by $\mathcal{O}(\tau(N(d+1)))$ for APE1 and $\mathcal{O}(\tau(Nd+N\log N))$ for APE2. In terms of memory storage, it requires storing the set of vertices and the projection matrices. This value is upper bounded by τKd (note that many vertices can be shared among projected convex hulls). In test time, APE1 projects data into τ random subspaces of dimension 1, $\mathcal{O}(\tau d)$, and then checks if the data point lies inside each interval $\mathcal{O}(\tau)$. APE2 test complexity just adds the complexity of checking if a point lies inside the polygon, $\mathcal{O}(M)$. Thus, the final test computational complexities for APE1 and APE2 are $\mathcal{O}(\tau(d+2))$ and $\mathcal{O}(\tau(d+M))$, respectively. NAPE adds k components to the method. This results in a k times increment of the computational complexities of APE1 or APE2. When comparing the theoretical training computational complexities for reasonable values, we may observe that APE and k-means variants have presumably the smallest computational complexities. These methods are closely followed by MoG. MST and NAPE come afterwards. Finally, the highest training complexity corresponds to SVDD. In test time, the presumably smallest complexity is given by k-means and APE variants, followed by MoG and SVDD. The most complex methods are k-NN, NAPE and MST. An estimation of training and testing time for all the methods is reported in Fig. 11. Tests have been performed in Matlab R2009a on 4-core Intel i5-2300@2.80 GHz desktop computer with 8 GiB RAM. Training time is represented by white bars and testing time with black ones. Positive bars are representative of slower times, negative bars represent faster times. Measures are reported with respect to APE-2 computational performance. For APE and NAPE, training and testing have been performed on 300 projections. Observe that the theoretical expected results are consistent with the bars shown in the figure.[5] APE-1 is the fastest algorithm in both training and testing, followed by APE-2. SVDD is the slowest algorithm in training, followed by NAPE, though NAPE is 15 times faster than SVDD. NAPE is the slowest method in testing, followed by MST by a slight difference. NAPE is built using many APE-2 classifiers and the size of its ensemble depends by the radius parameter. If the problem is highly non-convex, many polytopes are needed to properly approximate the geometry of the original structure.

7. Conclusions

In this work, the approximate polytope ensemble (APE) and non-convex approximate polytope ensemble (NAPE) are presented.

[4] Some computational complexities can differ according to the algorithm used. We have chosen the most fair or known algorithms complexities.

[5] The training time of k-NN shown in the figure corresponds to the time for setting the distance threshold value for a certain outlier rejection rate.

APE is based and extends the geometric concept of convex hull to model one-class classification problems. Expansion and contraction of the original polytope governed by a parameter α allows to avoid over-fitting and to choose the optimal operating point in the ROC curve. The high computational complexity for building the convex hull in high dimensional spaces is handled by projecting data down to one or two-dimensional spaces. In those low-dimensional spaces, building the convex hull and check if a point lies inside the polygon are well known problems with very efficient solutions. NAPE extends this approach using a tiling strategy of convex patches able to model non-convex structures. APE and NAPE have been compared on three different typologies of problems with widely used one-class classifiers. When highly non-convex distributions are present, the differences in performance between NAPE and APE-2 is significant, as highlighted by the results obtained on artificial datasets. On those datasets, MoG is competitive with NAPE. On 82 one-class problems derived from UCI multi-class datasets, NAPE consistently outperforms the rest of the methods. A specific one-class problem related to mobile-phone user verification from walking patterns has been also used for evaluating the new methodology. On such problem, NAPE provides the best solution due to its ability to manage strong non-convex distributions. Finally, the proposed methods are validated in 100 text categorization datasets with very high dimensionality and scarce data availability. In those datasets NAPE converges to APE-2 and both significantly outperform the rest of the methods. Experimental results show that the number of projections needed by the method is not critical. For NAPE, there exists an optimal radius able to provide the best approximation at the expense of the time complexity. Further effort in improving the computational cost for building NAPE is needed. Additionally, the study of the role of the number of projections as a regularizer for controlling the model complexity and theoretical bounds on this number should be further investigated.

Conflict of interest

None declared.

Acknowledgments

This work has been partially supported by projects CONSOLIDER (CSD2007-00018), the EU FP7 Marie Curie Network iCareNet under Grant no. 264738 and RecerCaixa 2011 Remedi.

References

[1] N. Japkowicz, Concept-learning in the absence of counter-examples: an autoassociation-based approach to classification, in: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-95), 1995, pp. 518–523.

[2] D.M.J. Tax, One-Class Classification, Ph.D. Thesis, 2001.

[3] R. Camci, R.B. Chinnam, General support vector representation machine for one-class classification of non-stationary classes, Pattern Recognition 41 (10) (2008) 3021–3034.

[4] A. Ypma, D.M.J. Tax, R.P.W. Duin, Robust machine fault detection with independent component analysis and support vector data description, in: Proceedings of the 1999 IEEE Signal Processing Society Workshop on Neural Networks for Signal Processing IX, 1999, pp. 67–76.

[5] M. Bicego, M.A. Figueiredo, Soft clustering using weighted one-class support vector machines, Pattern Recognition 42 (1) (2009) 27–32.

[6] C.H.M. Girolami, G. RossW, Employing optimized combinations of one-class classifiers for automated currency validation, Pattern Recognition 37 (6) (2004) 1085–1096.

[7] Q. Tao, G. wei Wu, J. Wang, A new maximum margin algorithm for one-class problems and its boosting implementation, Pattern Recognition 38 (7) (2005) 1071–1077.

[8] J.W.B.Y.W. Zhang, B. Qin, Convex hull-based support vector machine rule extraction, in: 2012 9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), 2012, pp. 689–692.

[9] B.Z.M. Yang, Y. Liu, Z. Li, Classification by nearness in complementary subspaces, Pattern Analysis and Applications (2012) 1–14.

[10] A.R.G. Madureira, M. Ruano, On-line operation of an intelligent seismic detector, Soft Computing Applications, vol. 195, Springer, Berlin, Heidelberg, 2013, pp. 531–542.

[11] J. Hong, E.S. Kim, H.-J. Lee, Rotation-invariant hand posture classification with a convexity defect histogram, in: International Symposium on Circuits and Systems, IEEE, 2012, pp. 774–777.

[12] K.P. Bennett, E.J. Bredensteiner, Duality and geometry in SVM classifiers, in: Proceedings of the 17th International Conference on Machine Learning, ICML00, Morgan Kaufmann Publishers Inc., 2000, pp. 57–64.

[13] J. Bi, K.P. Bennett, Duality, geometry, and support vector regression, in: Proceedings of the Advances in Neural Information Processing Systems 14 (NIPS 2001), 2001, pp. 593–600.

[14] M. Tang, J. Zhao, R. Tong, D. Manocha, Gpu accelerated convex hull computation, Computers and Graphics 5 (36) (2012) 498–506.

[15] F. Aurenhammer, B. JÜttler, On computing the convex hull of (piecewise) curved objects, Mathematics in Computer Science 6 (3) (2012) 261–266.

[16] S. Vempala, The Random Projection Method, American Mathematical Society, 2004.

[17] L. Liu, P. Fieguth, D. Clausi, G. Kuang, Sorted random projections for robust rotation-invariant texture classification, Pattern Recognition 45 (6) (2012) 2405–2418.

[18] X. Zhang, Y. Jia, A linear discriminant analysis framework based on random subspace for face recognition, Pattern Recognition 40 (9) (2007) 2585–2591.

[19] G. Yu, G. Zhang, C. Domeniconi, Z. Yu, J. You, Semi-supervised classification based on random subspace dimensionality reduction, Pattern Recognition 45 (3) (2012) 1119–1135.

[20] W. Johnson, J. Lindenstauss, Extensions of lipschitz mapping into hilbert space, in: Conference in Modern Analysis and Probability, vol. 26 of Contemporary Mathematics, American Mathematical Society, 1984, pp. 189–206.

[21] R.I. Arriaga, S. Vempala, An algorithmic theory of learning: robust concepts and random projection, Machine Learning 63 (2006) 161–182.

[22] A. Rahimi, B. Recht, Weighted sums of random kitchen sinks: replacing minimization with randomization in learning, in: Proceedings of the Advances in Neural Information Processing Systems (NIPS 2008), MIT Press, 2008, pp. 1313–1320.

[23] A. Blum, Random projection, margins, kernels, and feature-selection, in: Lecture Notes on Computer Science, vol. 3940, 2005, pp. 52–68.

[24] P. Casale, O. Pujol, P. Radeva, Approximate convex hulls family for one-class classification, in: Proceedings of the 10th international conference on Multiple classifier systems (MCS'11), 2011, pp. 106–115.

[25] S.K. Ghosh, R.K. Shyamasundar, A linear time algorithm for obtaining the convex hull of a simple polygon, Pattern Recognition 16 (6) (1983) 587–592.

[26] C.-L. Chen, Computing the convex hull of a simple polygon, Pattern Recognition 22 (5) (1989) 561–565.

[27] F.P. Preparata, M.I. Shamos, Computational Geometry: An Introduction, Springer-Verlag, New York, Inc., 1985.

[28] D. Tax, R. Duin, Uniform object generation for optimizing one-class classifiers, Journal of Machine Learning Research 2 (2001) 155–173.

[29] D. Davidov, E. Gabrilovich, S. Markovitch, Parameterized generation of labeled datasets for text categorization based on a hierarchical directory, in: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '04), ACM, 2004, pp. 250–257.

[30] C. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, USA, 1995.

[31] P. Juszczak, D.M.J. Tax, R.P.W. Duin, Minimum spanning tree based one-class classifier, Neurocomputing 72 (2009) 1859–1869.

[32] T. Fawcett, An introduction to ROC analysis, Pattern Recognition Letters 27 (8) (2006) 861–874.

[33] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal Machine Learning Research 7 (2006) 1–30.

Pierluigi Casale is a post-doctoral researcher in the Signal Processing Group, Faculty of Electrical Engineering at Eindhoven University of Technology, The Netherlands. He holds a Ph.D. in Applied Mathematics from the University of Barcelona, with a thesis on approximated machine learning methods for physical activity recognition applications. He is currently a Marie Curie fellows and he takes part in Healthcare and Ambient Assisted Living projects sponsored by European and National founding schemes. His main research interests are in Continuous and Distributed Machine Learning applications, Ubiquitous Sensors Mining and Wireless Sensor Networks with special interest in continuous monitoring of patients in hospital and home settings. His professional interests are in Data Science and Big Data.