Vietnam J Comput Sci (2016) 3:223–233
DOI 10.1007/s40595-016-0069-x

REGULAR PAPER

Mixture of hyperspheres for novelty detection

Duy Nguyen[1] · **Vinh Lai**[1] · **Khanh Nguyen**[1] · **Trung Le**[1]

Received: 30 November 2015 / Accepted: 6 May 2016 / Published online: 4 June 2016

Abstract In this paper, we present a mixture of support vector data descriptions (mSVDD) for one-class classification or novelty detection. A mixture of optimal hyperspheres is automatically discovered to characterize data. The model includes two parts: log likelihood to control the fit of data to model (i.e., empirical risk) and regularization quantizer to control the generalization ability of model (i.e., general risk). Expectation maximization (EM) principle is employed to train our proposed mSVDD. We demonstrate the advantage of the proposed model: if learning mSVDD in the input space, it simulates learning a single hypersphere in the feature space and the accuracy is thus comparable, but the training time is significantly shorter.

Keywords Mixture of experts · Mixture model · Kernel method · One-class classification

1 Introduction

Novelty detection is an interesting research topic in many data analytics and machine learning tasks ranging from video security surveillance, network abnormality detection, and detection of abnormal gene expression sequence to name a few. Different from the binary classification which focuses mainly on balance dataset, novelty detection aims to learn from imbalance dataset where a majority in the dataset is normal data, and abnormal data or outliers constitute a minor portion of dataset. The purpose of novelty detection is to find patterns in data that do not conform to expected behaviors [5]. To obtain this aim, a data description is constructed to capture all characteristics of normal data, and subsequently this description is used to detect abnormal data which cannot fit it well. These anomaly patterns are interesting because they reveal actionable information, the known unknowns and unknown unknowns. Notable real-world applications of novelty detection include intrusion detection [10], fraud detection (credit card fraud detection, mobile phone fraud detection, insurance claim fraud detection) [7], industrial damage detection [2], and sensor network data processing [13].

Probability density-based and neural network-based approaches have been proposed to address novelty detection problems. Another notable solution is the kernel-based approach. At its crux, data are mapped into the feature space with very high or infinite dimension via a transformation. In this space, a simple geometric shape is learned to discover the domain of novelty [16,20,21]. One-class support vector machine (OCSVM) [20] uses an optimal hyperplane, which separates the origin from data samples, to distinguish abnormality from normality. The domain of novelty in this case is certainly a positive region of the optimal hyperplane with maximal margin, the distance from the origin to the hyperplane. In another approach, support vector data description (SVDD) [21], the novelty domain of the normal data, is defined as an optimal hypersphere in the feature space, which becomes a set of contours tightly covering the normal data when mapped back to the input space.

Data in real-world applications are often collected from many different data sources. Such a heterogeneous dataset requires a mixture of individual data descriptions. A mixture of experts refers to the problem of using and combining many experts (e.g., classification or regression models) for classification or prediction purpose [12]. Two challenging obstacles that need to be addressed include: (1) how to automatically

✉ Trung Le
trunglm@hcmup.edu.vn

[1] Faculty of Information Technology, HCMc University of Pedagogy, Ho Chi Minh City, Vietnam

discover the appropriate number of experts in use or automatically do model selection; (2) how to train an individual expert with reference to others.

In this paper, we present a mixture of support vector data descriptions (mSVDD) for the novelty detection task. The idea of mSVDD is to use a set of hyperspheres as a data description for normal data. In mSVDD, we leverage the probabilistic framework with kernel-based method, where the model comprises two quantities: log likelihood to control the fit of data to the model (i.e., empirical error) and regularization quantizer to control the generalization capacity of the model (i.e., general error). The EM algorithm is used to train the model and the model is guaranteed to gradually converge to an appropriate set of optimal hyperspheres that well describes data. In the model, for each hypersphere (expert), a quantity, which expresses the total fit of data to this hypersphere, is utilized for model selection. We start with a maximal number of hyperspheres and step by step the redundant hyperspheres are eliminated. In the experimental section, we demonstrate the advantage of mSVDD: although learning in the input space, the set of hyperspheres offered by mSVDD is able to approximate a set of contours formed by a single optimal hypersphere in the feature space. In particular, our experiment on several benchmark datasets shows that mSVDD often yields a comparable classification accuracy while achieving a significant speed-up compared with the baselines.

To summarize, the contribution of the paper consists of the following points:

- We have viewed the problem of mixture of hyperspheres under the probabilistic perspective and proposed a mixture of support vector data description (mSVDD) for novelty detection. We have employed the EM algorithm to train mSVDD. The resultant model can automatically discover the appropriate number of experts in use and the weight of each expert.
- We have conducted the experiments on several benchmark datasets. The results have showed that mSVDD can learn mixture of hyperspheres in the input space which can approximately represent the set of contours generated by the traditional SVDD in the feature space. While the accuracy of mSVDD is comparable, the training time of mSVDD is faster than the kernelized baselines, since all computations are performed directly in the input space.
- Compared with the conference version [15], we have introduced two new strategies, i.e., approximate SVDD (amSVDD) and probabilistic mSVDD (pmSVDD), to improve the training time and accuracy compared with the general mSVDD. We have also conducted more experiments to evaluate these two new strategies and investigated the behaviors of the proposed algorithms.

2 Related work

We review the studies on a mixture of experts closely related to our work. These works can be divided into two branches: mixture of hyperplanes and mixture of hyperspheres. The works presented in [6,12,14] applied divide-and-conquer strategy to partition the input space into many disjoint regions, and in each region, a linear or nonlinear SVM can be employed to classify data. In [19], multiple hyperplanes were used for learning to rank. It also follows up the divide-and-conquer strategy and the final rank is aggregated by the ranks offered by the hyperplanes. Multiple hyperplane was brought into play in [1,24] for multiclass classification where each class was characterized by a set of hyperplanes. In [9], a mixing of linear SVMs was used to simulate a nonlinear SVM. This approach has both the efficiency of linear SVM and the power of nonlinear SVM for classifying data. Recently, a Dirichlet process mixture of large-margin kernel machine was proposed in [27] for multi-way classification. This work conjoined the advantages of Bayesian nonparametrics in automatically discovering the underlying mixture components and maximum entropy discrimination framework in integrating large-margin principle with Bayesian posterior inference.

Yet another approach to combine linear classifiers for nonlinear classification is ensemble methods including bagging [3], boosting [8], and random forests [4]. They implement a strong classifier by integrating many weak classifiers. However, since ensemble methods tend to use a great number of base classifiers and for stable model like SVM, the training is quite robust to data perturbation. Hence, the classifiers obtained at different stages are usually highly correlated and this would increase the number of iterations required for convergence and also bring the negative effect to the performance of combination scheme with such classifiers used as weak learners.

Mostly related to ours are the works of [17,18,25], which are shared with us the idea of using a set of hyperspheres as the domain of novelty. In [25], the procedure to discover a set of hyperspheres is ad hoc, heuristic and does not conform to any learning principle. The works of [17,18] are driven by the principle learning with minimum volume. However, the main drawback in those works is that the model selection cannot be performed automatically and the number of hyperspheres in use must be declared a priori.

3 Background

3.1 Expectation maximization principle

The main task in machine learning problems is to find the optimal parameter θ of the model given a training set D. In

probabilistic perspective, it is described as a maximization of log likelihood function. However, estimating parameters using maximum-likelihood principle is very hard if there are some missing data or latent variables (i.e., cannot be observed). The expectation maximization principle [11], for short EM, was proposed to overcome this obstacle. EM is an iterative method, in which each iteration consists of expectation step (E-step) followed by the maximization step (M-step). The basic idea of EM is described as follows.

Let $\{x_n\}_{n=1}^N$ be the observed data and $\{z_n\}_{n=1}^N$ the latent variables; the log likelihood function is given as follows:

$$l(\theta) = \sum_{n=1}^{N} \log\ p(x_n \mid \theta) = \sum_{n=1}^{N} \log \left[\sum_{z_n} p(x_n, z_n \mid \theta) \right].$$

As can be seen, the log cannot be pushed inside the sum; hence it is hard to find the maximum of $l(\theta)$. Instead of finding maximum likelihood, EM involves the complete data log likelihood as follows:

$$l_c(\theta) = \sum_{n=1}^{N} \log\ p(x_n, z_n \mid \theta).$$

In the E-step, the expected complete data log likelihood $Q\left(\theta, \theta^{t-1}\right)$ is computed by formulation as follows:

$$Q\left(\theta, \theta^{t-1}\right) = \mathbb{E}\left[l_c(\theta) \mid D, \theta^{t-1} \right].$$

Then, in the M-step, θ^t is found by optimizing the Q function w.r.t θ

$$\theta^t = \arg\max\ Q\left(\theta, \theta^{t-1}\right).$$

3.2 Weighted support vector data description

Given the training set $D = \{(x_n, y_n)\}_{n=1}^N$ where $x_n \in \mathbb{R}^d$ and $y_n \in \{-1; 1\}$, $n = 1, \ldots, N$ including normal (labeled by 1) and abnormal (labeled by -1) observations. To find the domain of novelty, weighted SVDD [26] learns an optimal hypersphere which encloses all normal observations with tolerances. Each observation x_n is associated with a weight $C\lambda_n$ and a smaller value for the weight implies that the correspondent observation is allowed to have a bigger error. The optimization problem of weighted SVDD is defined as follows:

$$\begin{aligned} &\min_{c,R,\xi} \left(R^2 + C \sum_{n=1}^{N} \lambda_n \xi_n \right) \\ &\text{s.t.}: \|x_n - c\|^2 \le R^2 + \xi_n,\ n = 1, \ldots, N;\ y_n = 1 \\ &\|x_n - c\|^2 \ge R^2 - \xi_n,\ n = 1, \ldots, N;\ y_n = -1 \\ &\xi_n \ge 0,\ n = 1, \ldots, n \end{aligned} \tag{1}$$

where R, c are the radius and center of the optimal hypersphere, respectively.

It follows that the error at the observation x_n is given as

$$\xi_n = \max\left\{0,\ y_n\left(\|x_n - c\|^2 - R^2\right)\right\}$$

and the optimization problem in Eq. (1) can be rewritten as follows:

$$\min_{c,R}\left(R^2 + C \sum_{n=1}^{N} \lambda_n \max\left\{0,\ y_n\left(\|x_n - c\|^2 - R^2\right)\right\}\right).$$

4 Mixture of support vector data descriptions

4.1 Idea of mixture of support vector data descriptions

Let us denote the latent variable which specifies the expert of x_n by $z_n \in \{0; 1\}^m$. The latent variable z_n, a bit pattern of size m where only its jth component is 1 and others are 0, means that the jth expert (i.e., the jth SVDD) is used to classify the observation x_n. The graphical model of mSVDD is shown in Fig. 1.

In mSVDD, we describe normal data using a set of m hyperspheres denoted by $\mathbb{S}_j\left(c_j, R_j\right)$, $j = 1, \ldots, m$. Given the jth hypersphere and the observation x_n, the conditional probability to classify x_n w.r.t the jth hypersphere is given as follows:

$$\begin{aligned} &p\left(\text{normal} \mid x_n, \mathbb{S}_j\right) = p\left(y_n = 1 \mid x_n, \mathbb{S}_j\right) \\ &\quad = p\left(y_n = 1 \mid x_n, z_n^j = 1, \theta\right) = s\left(R_j^2 - \left\|x_n - c_j\right\|^2\right) \\ &p\left(\text{abnormal} \mid x_n, \mathbb{S}_j\right) = p\left(y_n = -1 \mid x_n, \mathbb{S}_j\right) \\ &\quad = p\left(y_n = -1 \mid x_n, z_n^j = 1, \theta\right) = s\left(\left\|x_n - c_j\right\|^2 - R_j^2\right), \end{aligned} \tag{2}$$

where $s(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function and θ is the model parameter.

Intuitively, $p\left(\text{normal} \mid x_n, \mathbb{S}_j\right)$ is exactly $\frac{1}{2}$ if x_n locates at the boundary of the hypersphere $\mathbb{S}_j$ and increases to 1 when x_n resides inside the hypersphere or decreases from 0 when

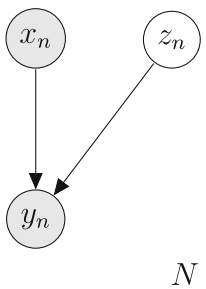

Fig. 1 Graphical model of a mixture of SVDD

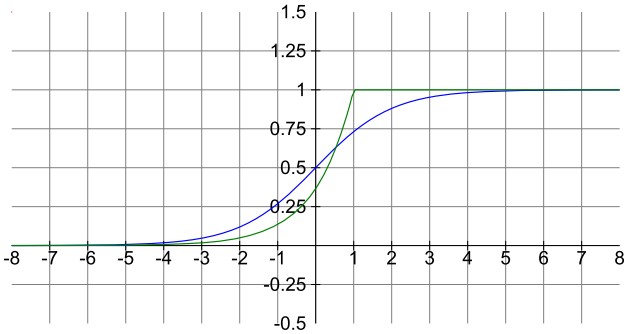

Fig. 2 Plots of two functions on the same coordinate when $\delta = 0$

x_n resides outside the hypersphere and moves apart from it. Because it is difficult to handle the log of $s(x) = \frac{1}{1+e^{-x}}$, we approximate $s(x)$ as follows:

$$s(x) = \frac{1}{1+e^{-x}} \approx e^{-\max\{0;\delta-x\}}. \tag{3}$$

To visually see how tight the approximation is, we plot the above two functions on the same coordinate when $\delta = 1$. As can be seen in Fig. 2, the blue line (presenting function $\frac{1}{1+e^{-x}}$) is close to the green line (presenting function $e^{-\max\{0;\delta-x\}}$). Therefore, the function $e^{-\max\{0;\delta-x\}}$ is a good approximation of the sigmoid function $s(x)$.

The marginal likelihood is given as: $p(Y|X,\theta) = \prod_{n=1}^{N} p(y_n \mid x_n, \theta)$, where $X = [x_n]_{n=1}^{N} \in \mathbb{R}^{d\times N}$, $Y = [y_n]_{n=1}^{N} \in \mathbb{R}^{N}$, and $\theta = (\theta_1, \theta_2, \ldots, \theta_m)$ encompass the parameters of all experts.

4.2 Optimization problem

To take into account both the general and empirical risks, the following optimization problem is proposed:

$$\max_{\theta} \left(-\sum_{j=1}^{m} R_j^2 + C \times \log p(Y \mid X, \theta) \right). \tag{4}$$

In the optimization problem as shown in Eq. (4), we minimize $\sum_{j=1}^{m} R_j^2$ to maximize the generalization capacity of the model. In the meanwhile, we maximize the log marginal likelihood $\log p(Y \mid X, \theta)$ to boost the fit of the model to the observed data X. The trade-off parameter C is used to control the proportion of the first and second quantities or to govern the balance between overfitting and underfitting.

To efficiently solve the optimization problem in Eq. (4), we make use of EM principle by iteratively performing two steps: E-step and M-step.

4.2.1 E-step

Given $z = (z_1, z_2, \ldots, z_N)$, we first compute the complete log likelihood as

$$l_c(\theta \mid D) = \log p(Y, \mathbf{z} \mid X, \theta) = \sum_{n=1}^{N} \log p(y_n, z_n | x_n, \theta). \tag{5}$$

It is clear that (cf. [15] for details)

$$\begin{aligned} &\log p(y_n, z_n \mid x_n, \theta) \\ &= \log \left(\prod_{j=1}^{m} \left(p\left(y_n \mid z_n^j = 1, x_n, \theta\right) p\left(z_n^j = 1 \mid x_n, \theta\right) \right)^{z_n^j} \right) \\ &= \sum_{j=1}^{m} z_n^j \left(-\xi_j(x_n) + \log \alpha_j \right), \end{aligned} \tag{6}$$

where $\alpha_j \triangleq p\left(z_n^j = 1 \mid \theta\right)$ is interpreted as the mixing proportion of the j^{th} expert and satisfies $\sum_{j=1}^{m} \alpha_j = 1$. Because it is difficult to compute the log of $p\left(y_n \mid x_n, z_n^j = 1, \theta\right)$, as mentioned before, we approximate $p\left(y_n \mid x_n, z_n^j = 1, \theta\right)$ as

$$\begin{aligned} &p\left(y_n \mid x_n, z_n^j = 1, \theta\right) \\ &= e^{-\xi_j(x_n)} = e^{-\max\left\{0;\delta - y_n\left(R_j^2 - \|x_n - c_j\|^2\right)\right\}}. \end{aligned}$$

Therefore, we have

$$\log p(y_n, z_n \mid x_n, \theta) = \sum_{j=1}^{m} z_n^j \left(-\xi_j(x_n) + \log \alpha_j \right).$$

Substituting Eq. (6) in Eq. (5), we gain the following:

$$\begin{aligned} l_c(\theta \mid D) &= \sum_{n=1}^{N} \sum_{j=1}^{m} z_n^j \left(-\xi_j(x_n) + \log \alpha_j \right) \\ &= \sum_{j=1}^{m} \sum_{n=1}^{N} z_n^j \left(-\xi_j(x_n) + \log \alpha_j \right). \end{aligned}$$

To fulfill the E-step, we compute the following conditional expectation when $\mathbf{z}$ is varied (cf. [15] for details):

$$\begin{aligned} \mathbb{E}(\theta) &= \left\langle -\sum_{j=1}^{m} R_j^2 + C l_c(\theta \mid D) \right\rangle_{z/D,\theta^{(t)}} \\ &= -\sum_{j=1}^{m} R_j^2 + C \sum_{j=1}^{m} \sum_{n=1}^{N} \left\langle z_n^j \right\rangle_{z/D,\theta^{(t)}} \left(-\xi_j(x_n) + \log \alpha_j \right) \end{aligned}$$

$$= -\sum_{j=1}^{m} R_j^2 + C\sum_{j=1}^{m}\sum_{n=1}^{N} \tau_{jn}^{(t)} \left(-\xi_j\left(x_n\right) + \log \alpha_j\right),$$

where $\theta^{(t)}$ is the value of parameter θ at t^{th} iteration and $\tau_{jn}^{(t)} \triangleq p\left(z_n^j = 1 \mid x_n, y_n = 1, \theta^{(t)}\right)$ can be computed by the Bayes formula as follows:

$$\begin{aligned}\tau_{jn}^{(t)} &\triangleq p\left(z_n^j = 1 \mid x_n, y_n, \theta^{(t)}\right) \\ &= \frac{p\left(y_n \mid z_n^j = 1, x_n, \theta^{(t)}\right)\alpha_j^{(t)}}{\sum_{k=1}^{m} p\left(y_n \mid z_n^k = 1, x_n, \theta^{(t)}\right)\alpha_k^{(t)}}.\end{aligned}$$

4.2.2 M-step

In this step, we need to find $\theta = (\theta_1, \theta_2, \ldots, \theta_m)$ where $\theta_j = \left(\alpha_j, c_j, R_j\right)$, which maximizes $\mathbb{E}(\theta)$. To this end, we rewrite this function as

$$\begin{aligned}\mathbb{E}(\theta) &= -\sum_{j=1}^{m}\left(R_j^2 + \sum_{n=1}^{N} C\tau_{jn}^{(t)}\xi_j\left(x_n\right)\right) + C\sum_{j=1}^{m}\tau_j^{(t)}\log\alpha_j \\ &= -E_1(\mathbf{R}, \mathbf{c}) + CE_2(\boldsymbol{\alpha}),\end{aligned}$$

where $\tau_j^{(t)} \triangleq \sum_{n=1}^{N}\tau_{jn}^{(t)}$, $\mathbf{R} = \left[R_j\right]_{j=1}^{m}$, $\mathbf{c} = \left[c_j\right]_{j=1}^{m}$, and $\boldsymbol{\alpha} = \left[\alpha_j\right]_{j=1}^{m}$.

It is obvious that the two following optimization problems need to be solved. The first becomes

$$\max_{\boldsymbol{\alpha}}\left(\sum_{j=1}^{m}\tau_j^{(t)}\log\alpha_j\right)$$
$$\text{s.t.}: \sum_{j=1}^{m}\alpha_j = 1,\ \alpha_j \geq 0,\ j = 1, \ldots, m.$$

The rule to update $\boldsymbol{\alpha}$ is given as: $\alpha_j^{(t+1)} = \frac{\tau_j^{(t)}}{\sum_{k=1}^{m}\tau_k^{(t)}}$.

The second is given as:

$$\min_{c_j, R_j}\left(R_j^2 + \sum_{n=1}^{N} C\tau_{jn}^{(t)}\max\left\{0, \delta - y_n\left(R_j^2 - \left\|x_n - c_j\right\|^2\right)\right\}\right)$$
where $j = 1, \ldots, m$.

If we choose $\delta = 0$, the second is indeed split into m independent weighted SVDD problems as

$$\min_{c_j, R_j}\left(R_j^2 + \sum_{n=1}^{N} C\tau_{jn}^{(t)}\max\left\{0,\ y_n\left(\left\|x_n - c_j\right\|^2 - R_j^2\right)\right\}\right)$$
where $j = 1, \ldots, m$. (7)

4.3 How to do model selection in mixture of SVDDs

We start with the number of hyperspheres $m = m_{\max}$ and make use of the criterion $\alpha_j^{(t+1)} < \lambda$, where $\lambda > 0$ is a threshold, for decision to eliminate the jth hypersphere. The small value for the mixing proportion $\alpha_j^{(t+1)}$ of the jth hypersphere implies that $\tau_j^{(t)}$ is too small as compared to others. The quantity $\tau_j^{(t)} = \sum_{n=1}^{N} p(z_n^j = 1 \mid x_n, y_n = 1, \theta^{(t)})$ is interpreted as the sum of the relevance levels of observations to the jth hypersphere and its small value implies that this hypersphere is redundant and has a too small radius.

4.4 The approaches to make a decision in mSVDD

In mSVDD, a natural way to classify a new observation x is to examine whether it belongs to at least one of m hyperspheres. The decision function is given as follows:

$$\begin{aligned}f(x) = \max\Big\{&\operatorname{sign}\left(R_1^2 - \|x - c_1\|^2\right), \ldots, \\ &\operatorname{sign}\left(R_m^2 - \|x - c_m\|^2\right)\Big\}\end{aligned}$$

where R_j, c_j are radius and center of the j^{th} hypersphere, respectively.

The above decision function can be regarded as a hard way to decide the normality or abnormality. To take advantage of the probabilistic perspective of mSVDD, we propose a probabilistic approach to make a decision. This approach decides an observation x as a normal data point if more than k experts (e.g., $k = 0.8m$) classify it as normal data with a confidence level over ρ, that is

$$f(x) = \begin{cases} 1 & \text{if } \exists j:\ p\left(\text{normal} \mid x, \mathbb{S}_j\right) = 1 \\ 1 & \text{if } \left|\left\{j:\ p\left(\text{normal} \mid x, \mathbb{S}_j\right) \geq \rho\right\}\right| \geq k \\ -1 & \text{otherwise}\end{cases},$$

where the second case means that x is classified as normal if there are more than k experts that predict it as normal with a confidence level over ρ.

Table 1 Statistics of experimental datasets

Datasets	#Train	#Dim	Domain
a9a	26, 049	123	Social survey
usps	5833	256	OCR images
Mushroom	6500	112	Biology
Shuttle	34, 800	9	Physical
Splice	800	60	Biology

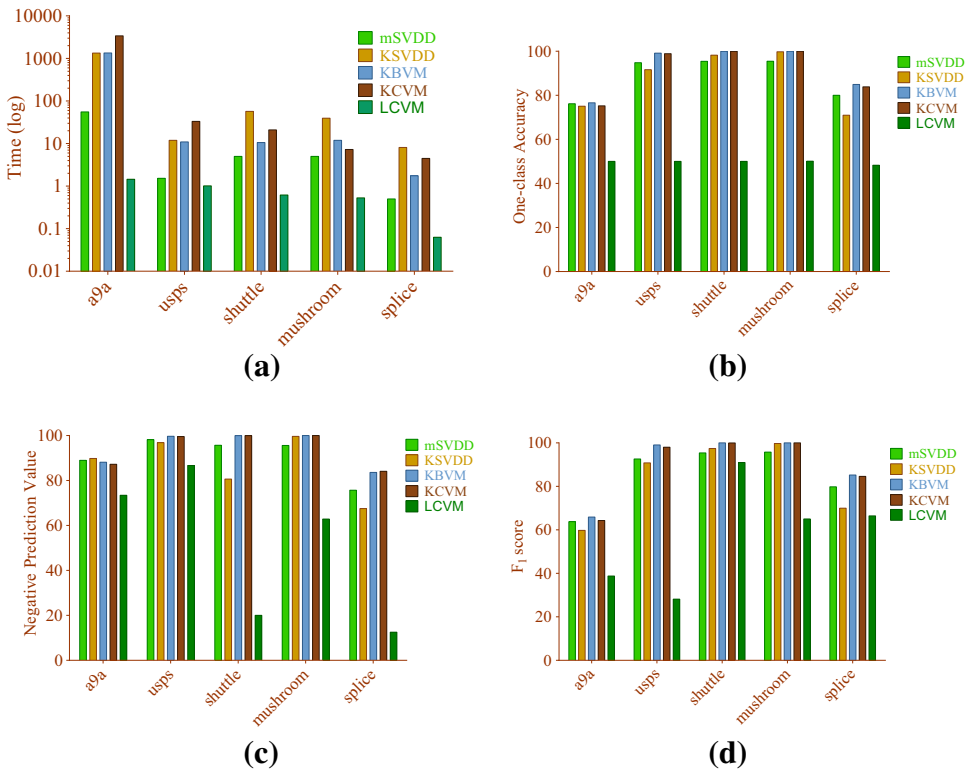

Fig. 3 Experimental results on the datasets. **a** Training times, **b** One-class accuracy, **c** negative prediction value, **d** F_1 score

4.5 Approximate mixture of SVDDs

According to Eq. (7), M-step requires the solutions of m weighted SVDD problems, each of which requires training the whole training set. It is also noteworthy that the variable $\tau_{jn}^{(t)}$ governs the influence of data point x_n to the jth hypersphere. A large value of $\tau_{jn}^{(t)}$ highly affects the determination of the jth hypersphere and vice versa, and a small value of $\tau_{jn}^{(t)}$ slightly affects the determination of the jth hypersphere. Therefore, the data point x_n with a small value of $\tau_{jn}^{(t)}$ can be eliminated when training the jth hypersphere without possibly changing the learning performance. In return, the training time of a mixture of SVDDs would be improved because training each SVDD now possibly involves a small subset of the training set. To realize this idea, we define the parameter $\mu \in (0, 1)$ as a threshold to eliminate data point x_n in the determination of the jth hypersphere (i.e., $\tau_{jn}^{(t)} < \mu$).

5 Experiment

5.1 Experimental settings

We establish the experiments over five datasets.[1]

[1] All datasets can be downloaded at the URL http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets.

The statistics of the experimental datasets is given in Table 1. To form the imbalanced datasets, we choose a class as the positive class and randomly select the negative data samples from the remaining classes such that the proportion of positive and negative data samples is 10 : 1. We make a comparison of our proposed mSVDD with ball vector machine (BVM) [22], core vector machine (CVM) [23] and support vector data description (SVDD) [21]. All codes are implemented in C/C++. All experiments are run on the computer with core I3 2.3GHz and 16GB in RAM.

5.2 How fast and accurate the proposed method compares with the baselines

We do experiments to investigate the performance of our proposed mSVDD in the input space compared with BVM, CVM and SVDD in the feature space with RBF kernel, named KBVM, KCVM, and KSVDD, respectively. To prove the necessity of mSVDD, we also compare mSVDD with CVM using the linear kernel (i.e., LCVM). We wish to validate that with mixture model dataset, a single hypersphere offered by LCVM cannot be sufficiently robust to classify it. We apply fivefold cross-validation to select the trade-off parameter C for KBVM, LCVM, KCVM, KSVDD, mSVDD and the kernel width parameter γ for KBVM, KCVM and KSVDD. The considered ranges are $C \in \{2^{-3}, 2^{-1}, \ldots, 2^{29}, 2^{31}\}$ and $\gamma \in \{2^{-3}, 2^{-1}, \ldots, 2^{29}, 2^{31}\}$. With mSVDD and LCVM, the linear kernel given by $K(x, x') = \langle x, x' \rangle$ is used, whereas

Table 2 One-class accuracy (in %) comparison

Dataset	One-class accuracy				
	mSVDD	SVDD	KBVM	KCVM	LCVM
a9a	76.18	75.06	76.62	75.24	50.00
usps	94.81	91.65	99.15	98.89	50.02
Shuttle	95.43	98.24	99.99	99.99	50.00
Mushroom	95.48	99.75	100	100	50.10
Splice	80.00	70.98	84.92	83.84	48.28

Table 3 Negative prediction value (in %) comparison

Dataset	Negative prediction value				
	mSVDD	KSVDD	KBVM	KCVM	LCVM
a9a	88.99	89.80	88.13	87.24	73.44
usps	98.20	96.84	99.67	99.58	86.67
Shuttle	95.68	80.63	99.99	99.97	20.00
Mushroom	95.61	99.64	100	100	62.86
Splice	75.70	67.52	83.60	84.11	12.50

Table 4 F_1 score (in %) comparison

Dataset	F_1 score				
	mSVDD	KSVDD	KBVM	KCVM	LCVM
a9a	63.76	59.75	65.91	64.30	38.76
usps	92.60	90.82	99.07	98.04	28.14
Shuttle	95.40	97.45	100	99.99	90.99
Mushroom	95.76	99.74	100	100	65.02
Splice	79.80	69.96	85.24	84.59	66.44

with kernel versions, the RBF kernel given by $K\left(x, x'\right) = e^{-\gamma\|x-x'\|^2}$ is employed. It is certainly true that the computation cost of the linear kernel is much lower than that of the RBF kernel. We fix $\delta = 0$ in all experiments; then in Sect. 5.4 we will investigate the behavior of mSVDD when δ is varied. We repeat each experiment five times and compute the means of the corresponding measures. We report training times (cf. Table 5 and Fig. 3a), the one-class accuracy (cf. Table 2 and Fig. 3b), the negative prediction values (NPVs) (cf. Table 3 and Fig. 3c) and F_1 score (cf. Table 4 and Fig. 3d) corresponding to the optimal values of C, γ.

In addition, the one-class accuracy is computed by $acc = \frac{\%TP+\%TN}{2} = \frac{acc^+ + acc^-}{2}$. This measure is appropriate for one-class classification, since it is required to achieve high values for both acc^+ and acc^- to produce a high one-class accuracy. The negative prediction values (NPVs) are computed by $NPV = \frac{TN}{TN+FN}$. This formulation indicates that a less number of false negative (FN) produces a higher value of NPV. F_1 score is the harmonic mean of precision and recall and is computed by $F_1 = \frac{2.precision.recall}{precision+recall}$ where $precision = \frac{TP}{TP+FP}$ and $recall = \frac{TP}{TP+FN}$. F_1 score can be rewritten as $F_1 = \frac{2.TP}{2.TP+FP+FN}$ which shows that a higher value of F_1 score is caused by a less number for both false negative and false positive.

Table 5 Training time (in s) comparison

Dataset	Training time				
	mSVDD	KSVDD	KBVM	KCVM	LCVM
a9a	55.48	1,334.75	1,342.05	3,396.91	1.45
usps	1.53	11.92	10.95	33.30	1.02
Shuttle	5.00	57.11	10.55	20.92	0.62
Mushroom	5.00	39.36	11.90	7.29	0.53
Splice	0.50	8.10	1.75	4.47	0.06

As observed from the experimental results, our proposed mSVDD is faster than others (except LCVM) on all datasets. This observation is reasonable since mSVDD learns m hyperspheres in the input space and, hence, is slower than learning only one hypersphere in the input space. However, mSVDD is faster than the kernel versions because of its lower kernel computation cost. Regarding the measures involving the learning performance, mSVDD is comparable or a little less than others especially KBVM and KCVM. The reason is that in mSVDD, each component hypersphere in the input space can simulate a contour generated by mapping back the optimal hypersphere of KSVDD, KBVM, and KCVM in the feature space into the input space. The slightly lower accuracy and NPVs of mSVDD compared to the kernel versions comes from the fact the hyperspheres of mSVDD may be less robust than its representative contours.

To visually support the above reason, we design experiments on 2-D datasets as displayed in Figs. 4 and 5. In Fig. 4, mSVDD recommends two hyperspheres in the input space to approximate two contours formed by a single hypersphere in the feature space. In Fig. 5, mSVDD offers three simple hyperspheres which can be visually seen to sufficiently simulate the set of contours formed by a single hypersphere in the feature space. The results of LCVM in all cases are worse, because it learns a single hypersphere in the input space which cannot sufficiently represent the contours formed by a single hypersphere in the feature space.

5.3 How approximate and probabilistic approaches improve mSVDD

We empirically compare mSVDD with its two variations which are approximate approach (amSVDD) and probability approach (pmSVDD). In amSVDD, we use the threshold $\mu = 0.8$ to eliminate data points for reducing training size

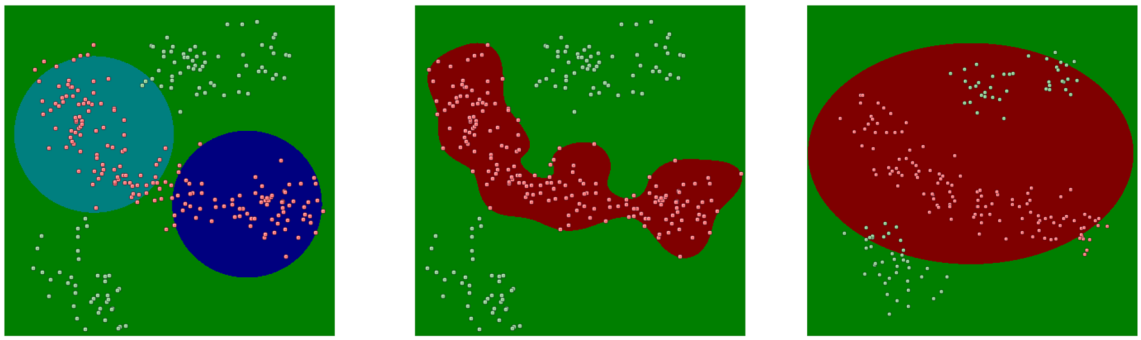

Fig. 4 Comparison of mSVDD with the linear kernel, SVDD with the RBF kernel, and CVM with the linear kernel

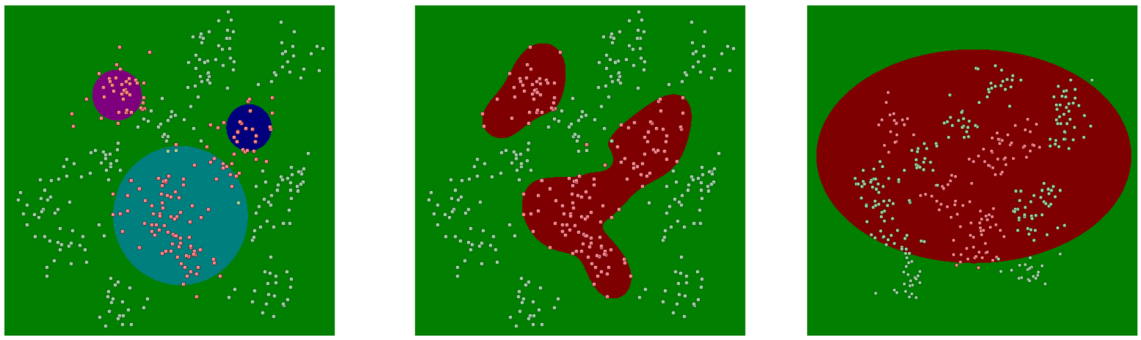

Fig. 5 Comparison of mSVDD with the linear kernel, SVDD with the RBF kernel, and CVM with the linear kernel

Table 6 One-class accuracy (in %) and training time (in s) comparison

Dataset	One-class accuracy			Training time		
	mSVDD	amSVDD	pmSVDD	mSVDD	amSVDD	pmSVDD
a9a	76.18	77.42	77.43	55.48	60.89	58.77
usps	94.81	93.29	95.20	1.53	1.41	1.09
Mushroom	95.48	92.74	92.35	5.00	4.74	5.39
Australian	81.39	81.06	82.57	0.06	0.02	0.06
Breast-cancer	96.44	94.51	96.75	0.02	0.01	0.02

Table 7 Negative prediction value (in %) and F_1 score (in %) comparison

Dataset	Negative prediction value			F_1 score		
	mSVDD	amSVDD	pmSVDD	mSVDD	amSVDD	pmSVDD
a9a	88.99	90.77	90.78	63.76	63.49	63.49
usps	98.20	97.77	98.36	92.60	89.85	92.68
Mushroom	95.61	92.14	91.42	95.76	92.25	91.79
Australian	83.02	84.69	84.12	79.23	79.90	80.66
Breast-cancer	92.37	86.59	94.19	97.13	95.14	97.54

in each SVDD problem. Our pmSVDD applies probability perspective and allows experts to vote when predicting new data. We set the parameter k to 2 and the parameter ρ to 0.8. Tables 6, 7 and Fig. 6 summarize the performance measures of mSVDD, amSVDD, and pmSVDD. The results show that pmSVDD achieves higher accuracies on all datasets, except mushroom, in comparison to mSVDD. This demonstrates that the probability approach really improves mSVDD by allowing a voting scheme and using probabilistic perspective. For amSVDD, its training time is usually less than others in

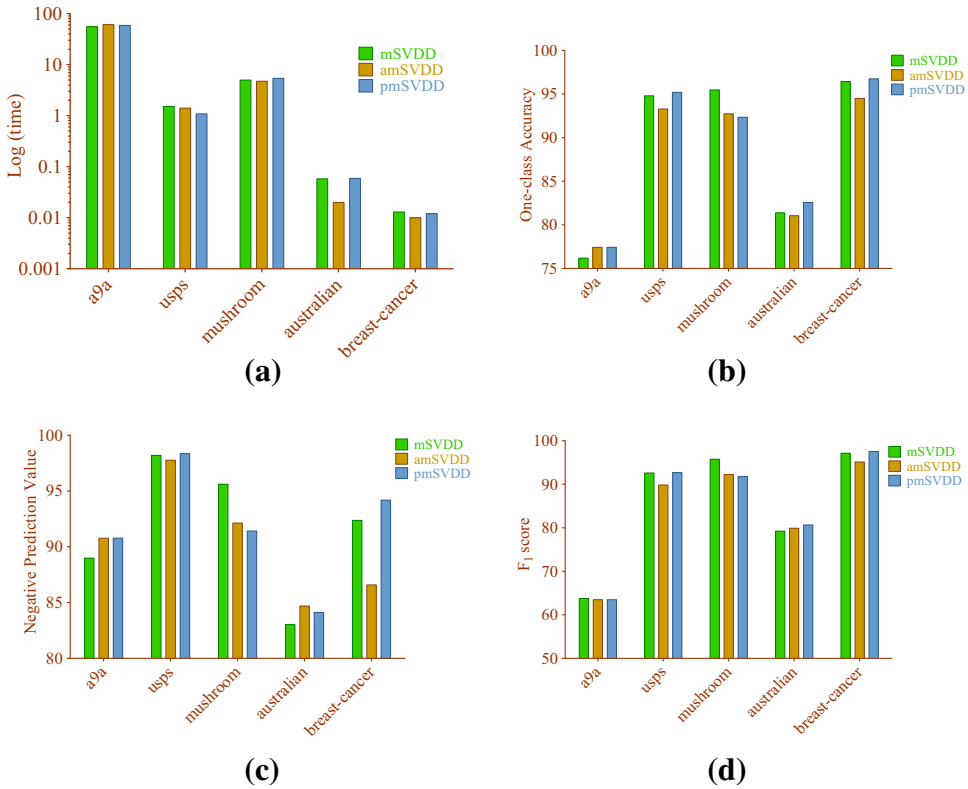

Fig. 6 Experimental results of the datasets on cross-validation. **a** Training times, **b** One-class accuracy, **c** negative prediction value, **d** F_1 score

Table 8 The one-class accuracy when parameter δ is varied

$\delta =$	−1	−0.75	−0.50	−0.25	0	0.25	0.50	0.75
a9a	74.93	76.08	75.54	74.78	74.32	75.69	75.07	75.41
usps	93.96	95.011	92.68	93.51	93.41	94.46	93.50	92.57
Mushroom	95.55	95.30	96.20	95.83	95.86	95.00	95.05	95.55
Australian	78.90	79.04	79.08	77.12	78.94	78.06	77.42	77.49
Diabetes	67.99	68.40	68.30	72.78	72.70	71.69	74.03	73.21

Table 9 The negative prediction value when parameter δ is varied

$\delta =$	−1	−0.75	−0.50	−0.25	0	0.25	0.50	0.75
a9a	87.86	88.83	88.56	87.69	87.49	88.39	88.00	88.14
usps	98.01	98.37	97.56	97.79	97.68	98.08	97.78	97.52
mushroom	95.61	96.12	96.19	95.24	96.29	94.45	95.02	94.82
australian	79.73	79.21	80.00	77.30	79.87	78.62	77.34	76.74
diabetes	49.28	50.73	51.88	51.78	52.99	52.42	54.21	52.10

all datasets, except a9a. Obviously, it comes from the reasonable reduction in training size of each SVDD problem at each step. This allows amSVDD to run faster than others while still preserving the accuracy.

5.4 How variation of the parameter δ influences accuracy

To analyze how parameter δ affects the one-class accuracy and negative prediction value, we conduct the experiment where δ is varied and other parameters are kept fixed. As observed from Tables 8 and 9, when δ is varied in ascending order, the accuracy at first increases to its peak and then gradually decreases. This fact may be partially explained as δ is altered and also changes the probability function of the observed data to classify them with respect to a particular hypersphere. When δ decreases, the component hypersphere tends to absorb more data outside it and, consequently, the hypersphere becomes bigger. Therefore, some abnormal data points can be misclassified. Reversely,

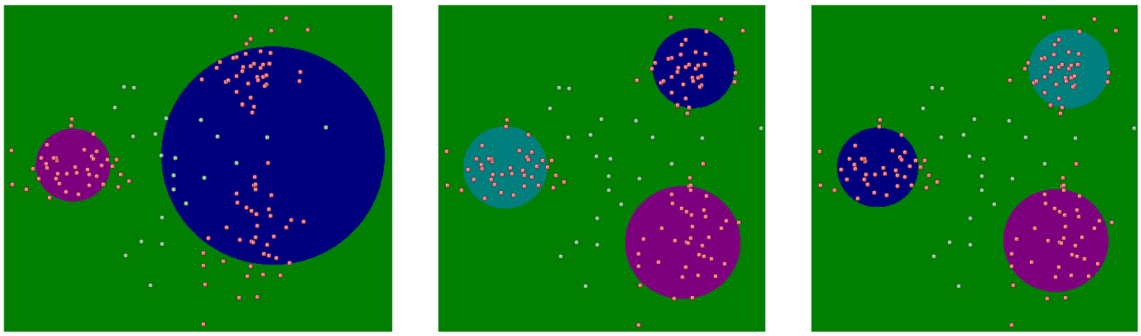

Fig. 7 Behavior of mSVDD when parameter δ is varied

the component hypersphere tends to be smaller and would not give a very high probability to data located inside the hypersphere and near the boundary. This prevents the model from misclassifying abnormal data points, but have the side effect of increasing the misclassification of normal data.

To visually manifest the above reason, we also provide simulation study on $2 - D$ datasets, as displayed in Fig. 7. When we set $\delta = -0.5$ and start with 3 hyperspheres, the hyperspheres tend to become bigger; as a result, two of them overlap each other. In the meanwhile, we increase δ to 1 and get a better solution which induces 90 % one-class accuracy. However, if we continue to increase δ to 12, the hypersphere is too small to describe normal data and, thus, the one-class accuracy declines to 88.75 %.

6 Conclusion

Leveraging on the expectation maximization principle, we propose a mixture of support vector data descriptions for one-class classification or novelty detection problem. Instead of learning the optimal hypersphere in the feature space involving costly RBF kernel computation, mSVDD learns a set of hypersphere(s) in the input space. Each component hypersphere is able to simulate a contour generated by mapping back the optimal hypersphere in the feature space into the input space. The experiments established on the benchmark datasets show that the mSVDD obtains shorter training time while achieving comparable one-class classification accuracies compared with other methods operated in the feature space.

References

1. Aiolli, F., Sperduti, A.: Multiclass classification with multi-prototype support vector machines. J. Mach. Learn. Res. **6**, 817–850 (2005)
2. Basu, S., Meckesheimer, M.: Automatic outlier detection for time series: an application to sensor data. Knowl. Inf. Syst. **11**(2), 137–154 (2007)
3. Breiman, L.: Bagging predictors. Mach. Learn. **24**(2), 123–140 (1996)
4. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
5. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM Comput. Surv. **41**(3), 1–58 (2009)
6. Collobert, R., Bengio, S., Bengio, Y.: A parallel mixture of svms for very large scale problems. Neural Comput. **14**(5), 1105–1114 (2002)
7. Fawcett, T., Provost, F.: Activity monitoring: Noticing interesting changes in behavior. In: Proceedings of the fifth ACM SIGKDD international conference on knowledge discovery and data mining, 53–62 (1999)
8. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci. **55**(1), 119–139 (1997)
9. Fu, Z., Robles-Kelly, A., Zhou, J.: Mixing linear SVMs for nonlinear classification. Neural Netw. IEEE Trans. **21**(12), 1963–1975 (2010)
10. Gwadera, R., Atallah, M.J., Szpankowski, W.: Reliable detection of episodes in event sequences. Knowl. Inf. Syst. **7**(4), 415–437 (2005)
11. Hartley, H.O.: Maximum likelihood estimation from incomplete data. Biometrics **14**(2), 174–194 (1958)
12. Jacobs, R.A., Jordan, M.I., Nowlan, S.J., Hinton, G.E.: Adaptive mixtures of local experts. Neural Comput. **3**(1), 79–87 (1991)
13. Janakiram, D., Reddy, V. A., Kumar, A. V. U. P.: Outlier detection in wireless sensor networks using bayesian belief networks. In: Proceedings of the 1st international conference on communication system software and middleware (Comsware), pp 1–6 (2006)
14. Krnger, S.E., Schafföner, M., Katz, M., Andelic, E., Wendemuth, A.: Mixture of support vector machines for hmm based speech recognition. ICPR **4**, 326–329 (2006)
15. Lai, V., Nguyen, D., Nguyen, K., Le, T.: Mixture of support vector data descriptions. In: Information and computer science (NICS), 2015 2nd national foundation for science and technology development conference, IEEE, pp 135–140 (2015)
16. Le, T., Tran, D., Ma, W., Sharma, D.: An optimal sphere and two large margins approach for novelty detection. Int Joint Conf Neural Netw IJCNN **2010**, 1–6 (2010)

17. Le, T., Tran, D., Ma, W., Sharma, D.: A theoretical framework for multi-sphere support vector data description. Lecture Notes in Computer Science, vol. 6444, pp. 132–142. Springer, Heidelberg (2010)
18. Le, T., Tran, D., Ma, W., Sharma, D.: Fuzzy multi-sphere support vector data description. In: FUZZ-IEEE, IEEE, pp. 1–5. (2012)
19. Qin, T., Zhang, X-D., Wang, D-S., Liu, T-Y., Lai, W., Li, H.: Ranking with multiple hyperplanes. In: Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval, SIGIR '07, 279–286. (2007)
20. Schölkopf, B., Platt, J.C., Shawe-Taylor, J.C., Smola, A.J., Williamson, R.C.: Estimating the support of a high-dimensional distribution. Neural Comput. **13**(7), 1443–1471 (2001)
21. Tax, D.M.J., Duin, R.P.W.: Support vector data description. J. Mach. Learn. Res. **54**(1), 45–66 (2004)
22. Tsang, I. W., Kocsor, A., Kwok, J. T.: Simpler core vector machines with enclosing balls. In: Proceedings of the 24th international conference on machine learning, ICML '07, pp. 911–918 (2007)
23. Tsang, I.W., Kwok, J.T., Cheung, P., Cristianini, N.: Core vector machines: fast SVM training on very large data sets. J Mach Learn Res **6**, 363–392 (2005)
24. Wang, Z., Djuric, N., Crammer, K., Vucetic, S.: Trading representability for scalability: Adaptive multi-hyperplane machine for nonlinear classification. In: Proceedings of the 17th ACM SIGKDD international conference on knowledge discovery and data mining, KDD '11, ACM, pp. 24–32. (2011)
25. Xiao, Y., Liu, B., Cao, L., Wu, X., Zhang, C., Hao, Z., Yang, F., Cao, J.: Multi-sphere support vector data description for outliers detection on multi-distribution data. In: ICDM Workshops, pp. 82–87. (2009)
26. Yan, J., Wang, Y., Cao, C., Zheng, H.: Example error weighted support vector data description. Comput. Eng. **2**, 009 (2005)
27. Zhu, J., Chen, N., Xing, E. P.: Infinite svm: a dirichlet process mixture of large-margin kernel machines. In: ICML, pp. 617–624. Omnipress, Madison (2011)