Sparsity-aware Support Vector Data Description Reinforced by Expectation Maximization

Mahdie Eghdami[1], Hadi Sadoghi Yazdi[2*], Neshat Salehi [3]

Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran

[1]eghdami.mahdie@mail.um.ac.ir
[2]h-sadoghi@ um.ac.ir
[3]salehi.63@mail.um.ac.ir
[*] Corresponding author

Conflicts of Interest Statement
The authors certify that they have NO affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter discussed in this manuscript.

Abstract
Support Vector Data Description (SVDD) characterizes a dataset by a spherically shaped boundary around it. Since the complexity of SVDD training is $O(N^3)$, its performance decreases for large-scale datasets. In this paper, we propose an improved SVDD algorithm, called EM-SVDD, which combines the Expectation Maximization (EM) algorithm and SVDD to reduce the complexity and accelerate the training phase, while the accuracy of the classifier remains unchanged. First, the dataset is clustered to obtain smaller subsets, and then the boundary of each subset is identified by SVDD. After that, to construct the dataset boundary and get the optimal weighted combination of SVDDs, the EM algorithm is utilized to estimate the parameters and weights of SVDDs. The time complexity of the proposed method is *N/i* times lower than SVDD, where *i* is the number of EM iterations. In addition to EM-SVDD, Sparse EM-SVDD is proposed to guarantee the sparsity of the iteratively estimated parameters. EM-SVDD is well compared with several similar methods. Simulation results indicate higher speed and performance of the proposed method in the training and testing phases. Furthermore, the capability of the proposed method is tested on a large image dataset acquired from social networks and our method identifies in-class and outlier images with 0.71 accuracy rate.

Keywords
Support Vector Data Description, Large Dataset, Mixed Classification, EM Technique.

1. Introduction

The one class classification (OCC) problem determines the boundary of a target set of samples (D. M. J. Tax, 2001). A one-class classifier differs from multiclass classifiers in two main aspects: 1) only the samples of one class are used to train the classifier and 2) the trained classifier recognizes in-class samples and outliers among the testing samples. Until now, many efforts have been made to address the problem of one class classification. These works can be divided into three major categories (Sanchez-Hernandez et al., 2007): reconstruction methods (Pizzi et al., 2001)(Yang et al., 1998), density methods (Fumera et al., 2000)(Dit-Yan Yeung & Chow, 2002), and boundary methods (Schölkopf et al., 2001)(David M J Tax & Duin, 1999). Among these categories, the boundary methods concentrate on the boundary that encloses the target class. The main superiority of the boundary methods is that they do not require extensive knowledge and a large amount of information about the dataset (D. M. J. Tax, 2001). Support vector data description (SVDD) (H. Jiang et al., 2019)(J. Wang et al., 2019) and one-class support vector machine (OCSVM) (H.-J. Xing & Li, 2020)(H.-J. Xing & Liu, 2020) are two commonly used methods among boundary-based classifiers. OCSVM describes the target class by a hyperplane in the feature space such that every sample located below the hyperplane is considered as an outlier. On the other hand, the objective function of SVDD tries to construct a minimum spherical boundary around the samples of the target class. However, under certain conditions, OCSVM and SVDD are proved to be equivalent (Schölkopf et al., 2001)(David M.J. Tax & Duin, 2004). SVDD is used in many research fields, such as outlier detection and clustering (Jae Hyuk Shin et al., 2011)(S. Wang et al., 2013)(Prakash & Singh, 2015), fault prognostic (Benkedjouh et al., 2012), process monitoring (Ge et al., 2011)(Q. Jiang et al., 2014), image reconstruction (Hwang et al., 2014) and feature selection (Nekkaa & Boughaci, 2015).

SVDD was introduced by Tax and Duin (David M.J. Tax & Duin, 2004) based on the SVM method (Vapnik, 1999). Following their approach, many methods attempted to improve the efficiency and the performance of classification, as explained in Section 2. In order to construct the decision boundary, a constrained convex quadratic programming (QP) problem is solved in the training phase of SVDD. Although all samples are considered in the training phase, the boundary of the classifier can be described by minority samples called support vectors (Y. Li, 2011)(Manevitz & Yousef, 2001).

Therefore, an efficient training set can be generated by selecting support vector samples. A test sample is classified as in-class when it is inside the decision boundary, while outliers are defined as samples located outside the boundary. Since the spherical boundary is not applicable to all kinds of classes, kernel functions are utilized. The kernel function transforms the data to a high-dimensional feature space and makes the classifier more flexible.

SVDD is impractical for large-scale datasets due to the high time and space complexity of the training phase. The space complexity of SVDD is $O(N^2)$ and its time complexity is $O(N^3)$, where N is the number of targets. Therefore, the major challenges of SVDD for large-scale datasets are dealing with the large amount of memory and enormous training time. In this paper, we propose EM-SVDD to reduce the computational complexity of the SVDD training. The idea is to construct the dataset boundary from a mixture of a finite number of weighted SVDD distributions with unknown parameters. To achieve this goal, the dataset is partitioned into smaller subsets and the SVDD classifier is used to build the boundary of each subset. After that, the support vectors that are not included in other SVDDs are selected. Then, to build the dataset boundary from the optimal weighted combination of SVDDs, the EM algorithm is used to estimate the weights of SVDDs and the coefficients of the selected support vectors.

The EM algorithm (Dempster et al., 1977)(Krishnan & McLachlan, 2008) is an iterative approach for parameter estimation. It is frequently used in the maximum likelihood estimate problem in the presence of missing data. This algorithm is appropriate for many statistical models especially Gaussian mixture models (GMMs) which is the focus of this paper. The major advantages of the EM algorithm over alternates are reliable global convergence, low computational cost per iteration, low storage requirement and easy programming. In addition, the sequence generated by the EM iterations increases the likelihood and often converges quickly to a maximum point.

Consider X as the set of observed data and Y as the set of complete data. In addition, let $L(\theta; X, Y) = p(X, Y \mid \theta)$ be the likelihood function generated by the statistical model, where θ is the vector of the parameters of the statistical model. The goal of the maximum likelihood estimate problem is to find the maximum likelihood estimate of θ, formulated in (1). However, solving this problem is not always simple. Therefore, the EM algorithm attempts to find the solution in two main steps. In the beginning of the algorithm, an initial guess about θ is made. In the first step, called the E-step, the conditional expected log-likelihood, termed the Q-function, is formed. The Q-function formulates the expected value of the log-likelihood function with regard to the conditional distribution of Y given X and the current value of θ. The Q-function is demonstrated in (2) where θ^g is the current estimate of θ. In the M-step, the new estimate of θ that maximizes the Q-function is constructed. These two steps iterate until convergence is achieved.

$$\hat{\theta}_{MLE} = \arg\max_{\theta} \log\left(L(\theta; X)\right) \tag{1}$$

$$Q\left(\theta \mid \theta^g\right) = E_{Y|X,\theta^g}\left[\log L(\theta; X)\right] = E_{Y|X,\theta^g}\left[\log L(\theta; X) \mid X, \theta^g\right] = E_{Y|X,\theta^g}\left[l(\theta; X)\right] \tag{2}$$

The rest of this paper is organized as follows. Related work is reviewed in Section 2. Motivations and the proposed method are explained in Section 3. Finally in Section 4, the experiments that demonstrate our method's advantages are presented.

2. Related Work

In this section, we review the methods which aim to solve the high complexity problem of SVM-based classifiers, with an emphasis on SVDD classifiers. Generally, these methods are divided into four categories:

I. *Dataset Level*: In these methods, a pre-processing is performed to remove a number or a group of non-support vector samples from the dataset. Therefore, the complexity of the QP optimization and SVDD training is reduced. Literature (Liang et al., 2009), proposed to extract boundary targets based on the boundary description determined by the support vectors. In Fast SVDD (FSVDD) (Luo et al., 2010), the dataset was decomposed and then the decomposed regions were combined to derive a global solution. Literature (Xiao et al., 2010), proposed a two-step method. First, the k-farthest neighbour method was used to identify the samples around the boundary, while other data samples were removed from the training set in the second step. Literature (Chaudhuri et al., 2016), developed a sampling-based method which computed the target class description by combining the iteratively computed SVDDs on independent random samples obtained from the training dataset. Incremental SVDD (Inc-SVDD) (Hua & Ding, 2011) and Incremental Fast SVDD (IFSVDD)(H. Jiang et al., 2019) analyzed the possible change of support vectors set after new samples were added to the training set.

II. *Feature Level*: Weston proved that the standard SVM can suffer from the presence of irrelevant features (Weston et al., 2000). Thus, feature level methods were proposed to select a smaller feature subset to improve the efficiency and the accuracy of the classifier. Literature (Vergara & Estévez, 2014), divided the feature selection methods into three groups: filter (Lorena et al., 2015), wrapper (Kohavi & John, 1997) and embedded methods (Lal et al., 2006). In filter methods, a metric independent of the classifier is used to evaluate the selected features. However, selected features are evaluated using the classifier in wrapper methods. Embedded methods attempt to take advantage of both filter and wrapper strategies. Fung et al. (Fung & Mangasarian, 2004) proposed a reformulation for SVM training to enforce feature sparsity in the solution. Literature (Peng et al., 2015), developed an outlier detection method based on SVDD in which sparse feature selection was modelled using integer programming and the complex problem was solved by

an iterative method. In (Lorena et al., 2015), a feature selection method was introduced for one class classification. First, a number of adapted feature importance measures were proposed. Then, a ranked list of features was created for each measure. Finally, the rankings were combined using rank aggregation methods.

III. *Classification Level*: These methods change the objective function or the conditions of the problem to reduce the complexity of the QP optimization problem. SMO (Platt, 1999) and SVMlight (Joachims, 1998) broke the large QP problem of the SVM training into a series of smallest QP problems. LIBSVM (Chang & Lin, 2011) was an online integrated software whose LIBSVM-SVDD component used SMO to accelerate SVDD training. KM-SVDD (D. M. J. Tax, 2001) used K-means clustering to divide the training set into smaller subsets. After that, SVDD was used to describe the boundary of each subset. The training time of SVDD was decreased in this algorithm due to the smaller size of the training set. After that, many efforts have been done to improve KM-SVDD, such as (C.-D. Wang & Lai, 2013)(D. Wang & Tan, 2013)(Xu et al., 2011). One of the most recent methods was SA-SVDD (Wu et al., 2016) which used Affinity Propagation clustering algorithm (Frey & Dueck, 2007) to cluster the input data. Then, SVDD was used to obtain the boundary of each cluster. The parameters of SVDD were acquired by the global prediction-based adaptive mutation particle swarm optimization algorithm (GPAM–PSO) (Q. Li et al., 2014). Recently, some coordinate descent methods for linear OCSVM and SVDD classifiers have also been developed to accelerate the convergence (Chou et al., 2020).In addition, a dynamic hyper-sphere SVDD without describing boundary was proposed to classify complicated datasets. In this method, first, important support vectors of the training dataset were extracted to describe the static hyper-sphere. Then, the dynamic hyper-sphere was determined using new important support vectors of the training dataset and testing sample. Finally, the testing sample was classified as an outlier if a significant change of hyper-sphere structure was observed.

IV. *Combining classification Level*: To refine the accuracy of one class classification and to obtain a more compact boundary for the target class as well, Tax and Duin (David M J Tax & Duin, 2001) proposed the ensemble of OCCs. In this approach, a number of base classifiers are combined together to benefit from their advantages (Alimoglu & Alpaydin, 1997)(Hatami & Ebrahimpour, 2007). An important issue in these methods is selecting the combination rules so that the base classifiers cover the weaknesses of each other. Zhang et al. (J. Zhang et al., 2011) utilized a number of SVDDs as the base classifiers, and combined their outputs using different rules. Hamdi and Bennani (Hamdi & Bennani, 2011) developed an ensemble of OCCs by utilizing the orthogonal projection operator and the bootstrap idea. The clustering-based ensemble of one-class classifiers was proposed by Krawczyk et al. (Krawczyk et al., 2014) where the target class was clustered into several sub-regions and then a single OCC was trained on each sub-region. Finally, the outputs of all the OCCs were combined together. Due to the computational overhead of constructing several base classifiers and their combinations, Zhou et al. (Zhou et al., 2002) proposed to select part of base classifiers to participate in the ensemble. Therefore, selective ensemble algorithms were appeared (N. Li & Zhou, 2009)(L. Zhang & Zhou, 2011)(Yan et al., 2017). As an instance, literature (H. Xing & Wang, 2017) proposed SESVDD, a selective ensemble strategy, where the Renyi entropy based diversity measure was used to get the optimal combination weights of base classifiers.

EM-SVDD belongs to the last category which proposes an ensemble of weighted SVDDs to reduce the complexity of SVDD training. The details of the proposed method are described in Section 3.

3. Proposed Method

3.1. Motivation

As previously declared, the performance of SVDD classifier decreases when the number of training samples increases. Therefore, to deal with large-scale datasets, this paper proposes an algorithm to speed up the SVDD training phase, without reducing its performance. Consequently, the proposed classifier can be used in the real-world applications which usually cope with thousands of training samples. To this aim, we break the main problem into a set of smaller sub-problems with lower complexities. After solving sub-problems, the solution of the main complex problem is determined by a mixture of sub-solutions. Fig. 1 summarizes the procedure of the proposed method. Considering the training dataset given in Fig. 1-a, first, the training data is partitioned into several subsets and a boundary is constructed for each subset (Fig. 1-b). After that, the boundary of the whole dataset is built by combining the subsets' boundaries (Fig. 1-c).

3.2. SVDD via Risk Minimization

Consider a dataset which contains N samples $\{x_i | x_i \in R^d, i = 1...., N\}$. SVDD aims to construct the hyper-sphere with minimal volume that encloses most of the dataset samples. For a hyper-sphere described by the centre a and the radius $R>0$, the cost function of the SVDD classifier is stated in (3), where λ is a known constant coefficient and the loss function, i.e. $l(e)$, is formulated as $l(e)= max (0, ||x-a||^2-R^2)$. Thus, $l(e)$ equals zero for samples located inside or on the hyper-sphere. On the other hand, $l(e)=||x-a||^2-R^2$ for samples that are outside the hyper-sphere.

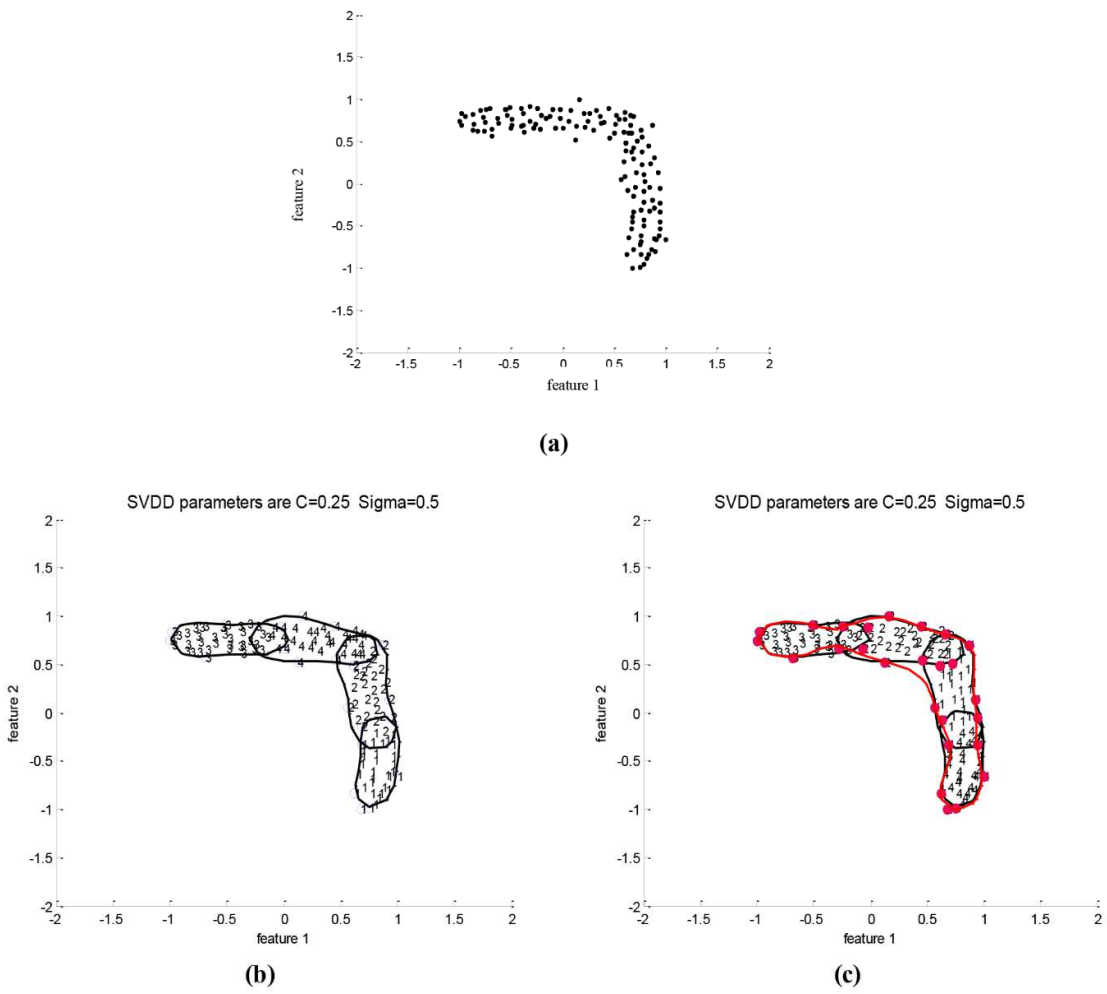


Figure 1: a) Training samples. b) Clustering the training samples and using SVDD classifier, the numbers present the cluster number of each sample. c) Selecting non-overlapped support vectors and estimating decision boundary.

$$J(R,a) = E\{l(e)\} + \lambda\|R\|^2 \tag{3}$$

On letting $E\{l(e)\} = \int l(e) f(e) de$, the cost function can be expressed as:

$$J(R,a) = \sum_{i=1}^{N} l(e_i) f(e_i) + \lambda\|R\|^2 \tag{4}$$

Using a non-parametric method, $f(e_i)$ can be written as:

$$f(e_i) = \frac{1}{N\sqrt{2\pi}h} \sum_{i=1}^{N} \exp\left(\frac{-(e-e_i)^2}{2h^2}\right) \tag{5}$$

Thus, in the case of replacing ξ_i by $l(e_i)$ in (4), the SVDD model can be defined by the following optimization problem:

$$\begin{aligned} &Minimize\ J(R,a) = \sum_{i=1}^{N} \xi_i f(e_i) + \lambda\|R\|^2 \\ &s.t.\quad \xi_i \geq \|x_i - a\| - R^2, \quad \xi_i \geq 0, \forall i \end{aligned} \tag{6}$$

By factoring out λ and setting $C=1/\lambda$ and $f(e_i)$=1, the standard SVDD optimization can be formulated as follows:

$$\begin{aligned} &Minimize\ J(R,a)=C\sum_{i=1}^{N}\xi_i+R^2 \\ &s.t.\quad \|x_i-a\|^2\le R^2+\xi_i,\quad \xi_i\ge 0,\forall i \end{aligned} \tag{7}$$

ξ_i indicates the misclassification penalty and the regularization parameter C indicates the trade-off between the volume of the sphere and the number of outliers. The parameter C should be selected before the SVDD training phase.
By using Lagrange multipliers $\alpha_i \ge 0$ and $\gamma_i \ge 0$ for the constraints and replacing them in (7), the following Lagrange function is obtained:

$$L=R^2+C\sum_{i=1}^{N}\xi_i-\sum_i\alpha_i\left\{R^2+\xi_i-\left(\|x_i-a\|^2\right)\right\}-\sum_{i=1}^{N}\gamma_i\xi_i \tag{8}$$

Setting the partial derivatives of L with respect to R, a and ξ_i to zeros, respectively, we have:

$$\sum_{i=1}^{N}\alpha_i=1,\quad a=\sum_{i=1}^{N}\alpha_i x_i,\quad 0\le\alpha_i\le C \tag{9}$$

Using the obtained results, we get the dual programming in (10), where $<x_i,x_j>$ indicates the inner product of the vectors. The samples with $0<\alpha_i<C$ are on the boundary of the hyper-sphere and are called support vectors. The samples with $\alpha_i=0$ are inside the hyper-sphere and are termed in-class samples. In addition, outliers which are outside the hyper-sphere have $\alpha_i=C$. The center a and the radius R can be obtained by figuring out the optimal solution of the dual problem in (10). As a result, a given test sample z is accepted if its distance to the hyper-sphere center is less than or equal to R; otherwise, it is rejected. The acceptance condition is shown in inequality (11), where each x_i is a support vector.

$$\begin{aligned} &Maximize\quad \sum_i \boldsymbol{\alpha_i}\langle \boldsymbol{x_i},\boldsymbol{x_i}\rangle-\sum_{i,j}\boldsymbol{\alpha_i}\boldsymbol{\alpha_j}\langle \boldsymbol{x_i},\boldsymbol{x_j}\rangle \\ &\boldsymbol{s.t.}\quad \sum_i \boldsymbol{\alpha_i}=\boldsymbol{1},\quad \boldsymbol{0}\le\boldsymbol{\alpha_i}\le\boldsymbol{C} \end{aligned} \tag{10}$$

$$\|z-a\|^2\le R^2\Rightarrow\langle z,z\rangle-2\sum_i\alpha_i\langle z,x_i\rangle+\sum_{i,j}\alpha_i\alpha_j\langle x_i,x_j\rangle\le R^2 \tag{11}$$

Similarly, we can get the dual programming in (12) for the kernel case, where $\phi(x_i,x_i)$ is the kernel function. For any support vector x_k, the radius of the hyper-sphere can be computed using (13). In addition, the acceptance condition is expressed in (14), in which each x_i is a support vector.

$$\begin{aligned} &Maximize\quad \sum_i \boldsymbol{\alpha_i}\boldsymbol{\varphi}(\boldsymbol{x_i},\boldsymbol{x_i})-\sum_{i,j}\boldsymbol{\alpha_i}\boldsymbol{\alpha_j}\boldsymbol{\varphi}(\boldsymbol{x_i},\boldsymbol{x_j}) \\ &\boldsymbol{s.t.}\quad \sum_i \boldsymbol{\alpha_i}=\boldsymbol{1},\quad \boldsymbol{0}\le\boldsymbol{\alpha_i}\le\boldsymbol{C} \end{aligned} \tag{12}$$

$$R^2(x_k)=\|x_k-a\|^2=\phi(x_k,x_k)-2\sum_i\alpha_i\,\phi(x_k,x_i)+\sum_{i,j}\alpha_i\alpha_j\,\phi(x_i,x_i) \tag{13}$$

$$\phi(z,z)-2\sum_i\alpha_i\,\phi(z,x_i)+\sum_{i,j}\alpha_i\alpha_j\,\phi(x_i,x_j)\le R^2 \tag{14}$$

Gaussian kernel function, demonstrated in (15), is usually used for transforming the samples into a higher-feature space. In this case, the bandwidth σ of the kernel function must be selected beforehand.

$$\phi(x_i,x_j)=\exp\left(\frac{-\|x_i-x_j\|^2}{2\sigma^2}\right) \tag{15}$$

3.3. EM-SVDD Algorithm

EM-SVDD's main idea has been taken from the GMM, in which the boundary of the dataset is assumed to be generated

from a mixture of a finite number of weighted SVDD distributions with unknown parameters:

$$SVDD_F(x;\theta) = \sum_{m=1}^{M} \omega_m \; SVDD_m(x;\theta_m) \tag{16}$$

$SVDD(x;\theta_m)$ is the probability density function of the m[th] partition of data which is described by an SVDD with parameter θ_m and ω_m is the weight assigned to the m[th] partition's SVDD. The other novelty of the paper is estimating the weights and the parameters of SVDDs via the EM technique. To achieve this goal, first, the training data is divided into M partitions. Second, partial SVDDs are constructed by applying the SVDD classifier to each partition. Third, a probability density function is constructed for each partition. Fourth, partial SVDDs are combined to construct the final boundary, where the EM algorithm is used to estimate the weights and the parameters of partial SVDDs. In the rest of this section, the steps of the EM-SVDD training phase are described in detail. After that, the testing phase of EM-SVDD is stated. Then, the computational complexity and the convergence of EM-SVDD are discussed. Finally, SEM-SVDD is proposed to refine the performance of EM-SVDD.

Step 1 Data partitioning

The training dataset is broken down into M data partitions $\{D_m\}_{m=1}^{M}$ where the m[th] partition consists of N_m samples. We suppose $N_m=k$, where k is a constant number and, as a result, the number of partitions is obtained by $M= N/k$, in which N is the number of training samples. The value of k should be chosen in such a way that the following statements hold:

- Since traditional SVDD will be applied to each partition, k is chosen so that SVDD achieves acceptable accuracies at reasonable times on partitions.
- M is large enough to correctly describe the complex datasets.

Step 2 SVDD applying

In this step, an SVDD classifier is applied to each data partition. Therefore, the SVDD's dual problem in (12) is solved for all partitions. Let the solution for the m[th] partition be β_{mi}, $i=1,\ldots,N_m$, the Lagrange coefficient of the i[th] support vector, and $J_m \subset \{1,\ldots,N_m\}$ be the set of the indices of the nonzero β_{mi}s. As a result, the trained Gaussian kernel support function for the m[th] data partition can be written as follows in (17) for each test sample z (Lee & Lee, 2007).

$$f_m(z) = 1 - 2\sum_{mi \in J_m} \beta_{mi} \; \phi(z, x_{mi}) + \sum_{mi,mj \in J_m} \beta_{mi}\beta_{mj} \; \phi(x_{mi}, x_{mj}) \tag{17}$$

Step 3 Constructing a prior embed for each partial SVDD

Utilizing the pseudo-density function expressed in (18) for each data partition, literature (Lee & Lee, 2007) proposed the pseudo-posterior probability distribution function demonstrated in (19) for the m[th] SVDD, where $f_m(x)$ and $r_m=R^2(x_k)$ can be computed using (17) and (13), respectively.

$$\hat{p}(x \mid m) = \frac{1}{2}(r_m - f_m(x)) \tag{18}$$

$$\delta(m \mid x) = const. \times \hat{p}(m \mid x) = \hat{p}(m).\hat{p}(x \mid m) = \frac{N_m}{N}(r_m - f_m(x)) \tag{19}$$

Using the distribution function in (19) makes it impossible to estimate the parameters, i.e., the Lagrange coefficients of support vectors, via the EM algorithm, due to the complexity of the Q-function derivative and the parameters' dependence. Furthermore, using (19) as the pseudo-probability distribution function in the gradient EM technique (Lange, 1995) is aborted, due to the limited domain of the logarithm function. In order to overcome this problem, we propose to modify the probability distribution function in (19) in such a way that parameters can be estimated in the EM algorithm. Thus, a new probability density function is defined using the exponential function, owing to its monotonicity and non-negative output. The new probability density function for SVDD is formulated as follows:

$$\delta'(m \mid x) = \exp\left(\frac{N_m}{N}(r_m - f_m(x))\right) \tag{20}$$

Step 4 Estimating the parameters of the target class's boundary using EM

Due to the similarity of our central idea to the GMM problem (Sundberg, 1972), we choose the EM technique to solve it. Since the likelihood function of the GMM problem is formulated as presented in (21) (Dempster et al., 1977), the log-likelihood function of the proposed method can be formulated as expressed in (22) (Dempster et al., 1977), where $p_m(x_n;\theta_m)$ and $SVDD_m(x_n;\theta_m)$ are probability density functions.

$$L(X;\theta)=p(X,m\mid\theta)=\prod_{n=1}^{N}\omega_m\, p_m(x_n;\theta_m) \tag{21}$$

$$\log(L(\theta;X))=l(\theta;X)=\sum_{n=1}^{N}\log(\omega_m SVDD_m(x_n;\theta_m))=\sum_{n=1}^{N}\log\omega_m+\sum_{n=1}^{N}\log SVDD_m(x_n;\theta_m) \tag{22}$$

Lemma: the conditional distribution $p(m|x_n,\theta^g)$ can be formulated as $p(m|x_n,\theta^g)=\frac{\omega_m^g p_m(x_n|\theta_m^g)}{\sum_{k=1}^{M}\omega_k^g p_k(x_n|\theta_k^g)}$. In our specific problem, $p_m(x_n|\theta_m^g)$ can be computed by the density function introduced in (20).

Proof: $p(m|x_n,\theta^g)$ represents the probability of occurrence of the mth SVDD with regard to the data sample x_n and the parameter θ^g. Utilizing the conditional probability definition, $p(m|x_n,\theta^g)$ can be rewritten as follows:

$$p\left(m\mid x_n,\theta^g\right)=\frac{p\left(x_n,m,\theta^g\right)}{p\left(x_n,\theta^g\right)}=\frac{p\left(x_n\mid m,\theta^g\right)p\left(m,\theta^g\right)}{p\left(x_n,\theta^g\right)}=\frac{p\left(x_n\mid m,\theta^g\right)p\left(m\mid\theta^g\right)p\left(\theta^g\right)}{p\left(x_n\mid\theta^g\right)p\left(\theta^g\right)}=\frac{p\left(x_n\mid m,\theta^g\right)p\left(m\mid\theta^g\right)}{p\left(x_n\mid\theta^g\right)} \tag{23}$$

$p(m|\theta^g)$ is the probability of occurrence of the mth SVDD with parameter θ^g, which is the same as the weight of the mth SVDD in the gth iteration, i.e., $p(m|\theta^g)=\omega_m{}^g$. In addition, $p(x_n|m,\theta^g)$ can be compressed as $p_m(x_n|\theta_m{}^g)$. Therefore, we obtain:

$$p\left(m\mid x_n,\theta^g\right)=\frac{p\left(m\mid\theta^g\right)p\left(x_n\mid m,\theta^g\right)}{p\left(x_n\mid\theta^g\right)}=\frac{\omega_m^g\; p_m\left(x_n\mid\theta_m^g\right)}{p\left(x_n\mid\theta^g\right)} \tag{24}$$

Using the conditional probability definition and the marginal distribution, i.e., $p(A)=\sum_B p(A,B)$, we have:

$$p\left(m\mid x_n,\theta^g\right)=\frac{\omega_m^g\; p_m\left(x_n\mid\theta_m^g\right)}{p\left(x_n\mid\theta^g\right)}=\frac{\omega_m^g\; p_m\left(x_n\mid\theta_m^g\right)}{\sum_k p\left(x_n,k\mid\theta^g\right)}=\frac{\omega_m^g\; p_m\left(x_n\mid\theta_m^g\right)}{\sum_k p\left(x_n\mid k,\theta^g\right)p\left(k\mid\theta^g\right)}=\frac{\omega_m^g\; p_m\left(x_n\mid\theta_m^g\right)}{\sum_{k=1}^{M}\omega_k^g\; p_k\left(x_n\mid\theta_k^g\right)} \tag{25}$$

■

In order to estimate the unknown parameters using the EM algorithm, first, the non-overlapped support vectors, i.e., support vectors that are not included in other SVDDs, are selected from all SVDDs. Then, the EM algorithm iterates over the E-step and the M-step to find the optimal weights of SVDDs and the Lagrange coefficients of selected support vectors. These two steps are described below:

- **E-step:** In this step, first, the probability of each sample associated with each SVDD is estimated using (25) and then, the Q-function is constructed by (26) (Dempster et al., 1977), in which θ and θ^g are the new and old parameters, respectively.

$$\begin{aligned}Q\left(\theta\mid\theta^g\right)&=E\left(l(\theta;X)\mid X,\theta^g\right)=\sum_{m=1}^{M}l(\theta;X)\,p\left(m\mid X,\theta^g\right)=\\&\sum_{m=1}^{M}\sum_{n=1}^{N}\log(\omega_m)\,p\left(m\mid x_n,\theta^g\right)+\sum_{m=1}^{M}\sum_{n=1}^{N}\log(SVDD_m(x_n;\theta_m))\,p\left(m\mid x_n,\theta^g\right)\end{aligned} \tag{26}$$

- **M-step**: In this step, $\theta^*=arg\,max\,Q(\theta|\theta^g)$ should be computed. To this aim, first, the constraint $\sum_{m=1}^{M}\omega_m=1$ is incorporated into (26) with the help of a Lagrange multiplier. Second, the partial derivative of $Q(\theta|\theta^g)$ with respect to ω_m is set to zero and, consequently, the new weights of the SVDDs are estimated:

$$\frac{\partial}{\partial\omega_m}\left[\sum_{m=1}^{M}\sum_{n=1}^{N}\log(\omega_m)\,p\left(m\mid x_n,\theta^g\right)+\lambda\left(\sum_{m=1}^{M}\omega_m-1\right)\right]=0 \tag{27}$$

$$\Rightarrow\quad\lambda=-N\quad and\quad\omega_m^*=\frac{1}{N}\sum_{n=1}^{N}p\left(m\mid x_n,\theta^g\right) \tag{28}$$

Third, the partial derivative of $Q(\theta|\theta^g)$ with respect to θ_m should be set to zero to find the new estimation of θ_m. Since in our problem, θ_m represents the Lagrange coefficients of the SVDDs, the partial derivative of $Q(\theta|\theta^g)$ with respect to β_{mi}, i.e., the coefficient of the ith selected support vector of the mth SVDD, is computed. Then, the density

function introduced in (20) is incorporated into the Q-function derivative and as a result, a new estimation of β_{mi} is obtained by setting the derivative to zero:

$$\frac{\partial Q\left(\theta \mid \theta^{g}\right)}{\partial \beta_{mi}} = \frac{\partial\left(\sum_{m=1}^{M}\sum_{n=1}^{N} \log\left[SVDD_m(x_n;\theta_m)\right] p\left(m \mid x_n, \theta^{g}\right)\right)}{\partial \beta_{mi}}$$

$$= \frac{\partial\left(\sum_{m=1}^{M}\sum_{n=1}^{N} \log\left[\exp\left(\frac{N_m}{N}\left(r_m - f_m(x_n)\right)\right)\right] p\left(m \mid x_n, \theta^{g}\right)\right)}{\partial \beta_{mi}}$$

$$= \frac{\partial\left(\sum_{m=1}^{M}\sum_{n=1}^{N} \log\left[\exp\left(\frac{N_m}{N}\left(r_m - 1 + 2\sum_{mi\in J_m}\beta_{mi}\,\phi(x_n, x_{mi}) - \sum_{mi,mj\in J_m}\beta_{mi}\beta_{mj}\,\phi\left(x_{mi}, x_{mj}\right)\right)\right)\right] p\left(m \mid x_n, \theta^{g}\right)\right)}{\partial \beta_{mi}} = 0 \quad (29)$$

$$\Rightarrow \quad \beta_{mi}^{*} = \frac{\sum_{n=1}^{N}\left(\left(e^{-\left\|x_n - x_i\right\|^2 / 2\sigma^2}\right) . p\left(m \middle| x_n, \theta^{g}\right)\right)}{N.\sum_{n=1}^{N}\left(p\left(m \middle| x_n, \theta^{g}\right)\right)} - \frac{1}{2}\sum_{j\in J_m, i\neq j}\beta_{mj} e^{-\left\|x_i - x_j\right\|^2 / 2\sigma^2} \quad (30)$$

Generally, EM-SVDD generates the weight together with the Lagrange coefficients of the selected support vectors of each SVDD. The procedure of the training algorithm is presented in Fig. 2 and illustrated in Fig. 3. Consider a dataset that contains a number of training samples given in Fig. 3-a. First, the training data is partitioned into several clusters (Fig. 3-b). After that, the SVDD classifier is applied to each cluster (Fig. 3-c). Hence, we will have some SVDD classifiers, each of which is an expert in a small area of the whole dataset region. After selecting non-overlapped support vectors, the Lagrange coefficients of support vectors are estimated using the EM algorithm to form the final classifier (Fig. 3-d).

Testing phase

After the training phase, in order to use the selected support vectors of all SVDDs in the final decision function, the coefficient of each support vector, γi, is defined by the product of its Lagrange coefficient and the corresponding SVDD's weight. To evaluate a test sample z, the decision function is formulated by:

$$f(z) = 1 - 2\sum_{i\in SV}\gamma_i\,\phi(z - x_i) + \sum_{i\in SV}\sum_{j\in SV}\gamma_i\,\gamma_j\,\phi\left(x_i, x_j\right) \quad (31)$$

3.4. Computational Complexity Analysis

In this section, the runtime complexity of the proposed method is calculated. First, the dataset of N samples with d features is clustered into M partitions using any clustering method, where $M=N/k$ and k is a constant number. In the case of using the fuzzy c-means (FCM) clustering method (Dunn, 1973), a clustering method in which each data sample can belong to one or more clusters, the runtime complexity of the first step is $O(NM^2dr)$, in which r is the number of FCM repetitions. By constant consideration of r, the complexity of the clustering step will be $O(NM^2d)$. Second, SVDD classifier is applied to each of the M clusters. Since the complexity of the SVDD training on a set of k samples is $O(k^3d^3)$, the complexity of performing M SVDDs is $O(Mk^3d^3)$. Third, the non-overlapped support vectors are selected which is performed in $O(Mkd)$. Fourth, the EM algorithm is utilized to estimate the parameters of final classifier. Since the complexity of the E-step and the M-step is $O(NM+N)$ and $O(2NM)$, respectively, the complexity of the last step will become $O(iNM)$ where i is the number of EM iterations. Thus, in the case of $i<<N$, the overall runtime complexity of the proposed method is obtained as follows:

$$RC_{EM-SVDD} = O\left(N.M^2.d + M.k^3.d^3 + M.k.d + i.N.M\right) = O\left(N.M^2.d\right) \quad (32)$$

By replacing N/k *by* M and bearing in mind that k is a constant number, the runtime complexity of the proposed method becomes:

$$RC_{EM-SVDD} = O\left(N^3.d + N.d^3 + N.d + i.N^2\right) = O\left(N^3.d\right) \quad (33)$$

The above relation shows that the runtime complexity of EM-SVDD is better than the traditional SVDD's runtime complexity which is $O(N^3d^3)$. In addition, it can be concluded that the runtime complexity of the proposed method depends on the runtime complexity of the FCM clustering. By considering the clustering phase as a preprocess step, the EM-SVDD's complexity is N/i times lower than traditional SVDD's.

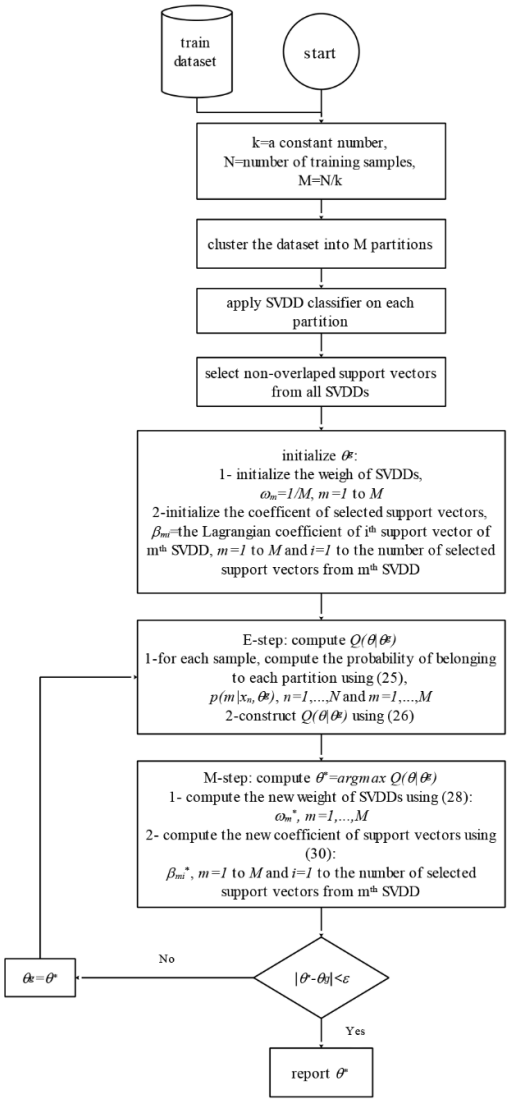


Figure 2: Flowchart of EM-SVDD

3.5. Convergence Analysis

As declared previously, θ^* is an estimation of θ which maximizes $Q(\theta|\theta^g)$. In addition, by performing the EM algorithm with the current estimation of θ, i.e. θ^g, $Q(\theta^g|\theta^g)=0$ will be resulted. In the gth iteration of the maximization step, θ^* is selected using (34).

$$\theta^* = \arg\max_{\theta} Q\left(\theta \mid \theta^g\right) \quad (34)$$

Thus, θ^* is selected in such a way that the inequality in (35) is established.

$$Q\left(\theta^{*} \mid \theta^{g}\right) \geq Q\left(\theta^{g} \mid \theta^{g}\right) \tag{35}$$

Therefore, according to (34), (35) and (2), the following analysis can be derived:

$$\begin{aligned} l\left(X \mid \theta^{*}\right) &\geq Q\left(\theta^{*} \mid \theta^{g}\right)+R\left(\theta^{g} \mid \theta^{g}\right) \\ &\geq Q\left(\theta^{g} \mid \theta^{g}\right)+R\left(\theta^{g} \mid \theta^{g}\right) \\ &= l\left(X \mid \theta^{g}\right) \end{aligned} \tag{36}$$

(36) satisfies the sufficient condition of convergence, which means that the logarithm of the likelihood increases in each iteration and will tend to a local maximum. Consequently, in the case of choosing a proper initial estimation of θ, it will move to a global maximum. The initial estimation of the proposed method is suitably chosen as the Lagrange coefficients of SVDDs.

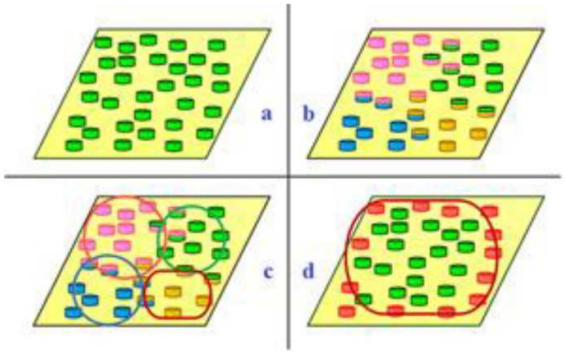


Figure 3: a) Training samples. b) Clustering the training samples. c) Using SVDD classifier. d) Selecting non-overlapped support vectors and estimating their coefficient.

3.6. Sparse EM-SVDD

In order to guarantee the sparsity of the parameters estimated in the EM iterations, Sparsity-aware EM-SVDD approach is introduced. Besides generating sparse parameters, SEM-SVDD will help to reduce the noise effect.
As discussed previously, the new estimation of parameter θ is computed as follows:

$$\theta^{new}=\arg\max_{\theta} p(\theta \mid X)=\arg\max_{\theta} \log p(\theta \mid X) \tag{37}$$

In addition, we know that the Laplacian prior for parameter θ is:

$$p(\theta)=\frac{\lambda}{r}\exp\left(-\lambda|\theta|\right) \tag{38}$$

Thus, $log\ p(\theta|X)$ which is termed $J(\theta)$ will be:

$$J(\theta)=\log\ p(\theta \mid X)=\log\left(p(X \mid \theta)p(\theta)\right)=\log\ p(X \mid \theta)+\log p(\theta)=\log\ p(X \mid \theta)+\log\frac{\lambda}{r}-\lambda|\theta| \tag{39}$$

Consequently, the new estimation of the parameter θ is obtained by maximizing $J(\theta)$. On letting $J_1(\theta)=log\ p(X|\theta)$ and $J_2(\theta)=log\ \lambda/r-\lambda|\theta|$, we have:

$$\begin{aligned} \theta^{new} &= \theta^{old}-\mu\frac{\partial J(\theta)}{\partial\theta}=\theta^{old}-\mu_1\frac{\partial J_1}{\partial\theta}-\mu 2\frac{\partial J 2}{\partial\theta} \\ &= \theta^{old}-\mu_1\frac{\partial J_1}{\partial\theta}-\mu_2(-\lambda\,\mathrm{sgn}(\theta)) \end{aligned} \tag{40}$$

, where θ^{old} equals the θ^{*} computed in each iteration of the M-step of EM-SVDD. Therefore, for each part of θ from (28) and (30) we have:

$$\omega_m^{new} = \omega_m^{*} + \mu_2 \lambda \operatorname{sgn}\left(\omega_m^{*}\right) \tag{41}$$

$$\beta_{mi}^{new} = \beta_{mi}^{*} + \mu_2 \lambda \operatorname{sgn}\left(\beta_{mi}^{*}\right) \tag{42}$$

To clarify the procedure of SEM-SVDD, its main steps are presented in Fig. 4.

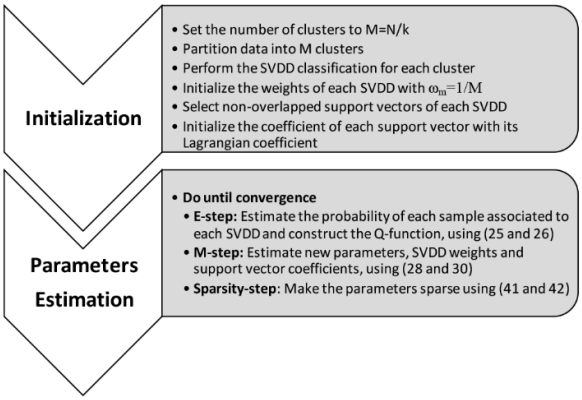


Figure 4: An overview of SEM-SVDD main steps.

4. Experimental Results

To evaluate EM-SVDD, 10 datasets taken from the UCI Machine Learning Dataset Repository (http://archive.ics.uci.edu/ml/) and 3 artificial datasets are used. In addition, an Intrusion Detection System (IDS) dataset is utilized to investigate the performance of the proposed method on large-scale datasets. All the experiments are performed on a PC with Intel Quad- Core 2.5 GHZ CPU and 4GB Memory, and Gaussian kernel is used. Furthermore, the FCM method is used to partition the dataset in all experiments. Besides the traditional canonical and online training methods, including standard SVDD (David M.J. Tax & Duin, 2004), FSVDD (Luo et al., 2010) , Inc-SVDD (Hua & Ding, 2011) and LibSVM with SVDD toolbox (Chang & Lin, 2011), EM-SVDD is compared with six ensemble-based classifiers which are ensemble of SVDDs by bagging (Breiman, 1996) , ensemble of SVDDs by AdaBoost (Freund et al., 1999), random subspace method based ensemble of SVDDs (RSMESVDDs) (Cheplygina & Tax, 2011), clustering based ensemble of SVDDs (CESVDDs) (Krawczyk et al., 2014), selective ensemble of SVDDs (SESVDDs) (H. Xing & Wang, 2017), and robust AdaBoost based ensemble of OCSVMs (H.-J. Xing & Liu, 2020). Furthermore, as a case study, EM-SVDD is applied to a large scale dataset of images obtained from social networks to identify in-class and outlier images. In this case study, the impact of the number of clusters on the performance of the proposed method has been investigated.

4.1. Evaluation Metrics

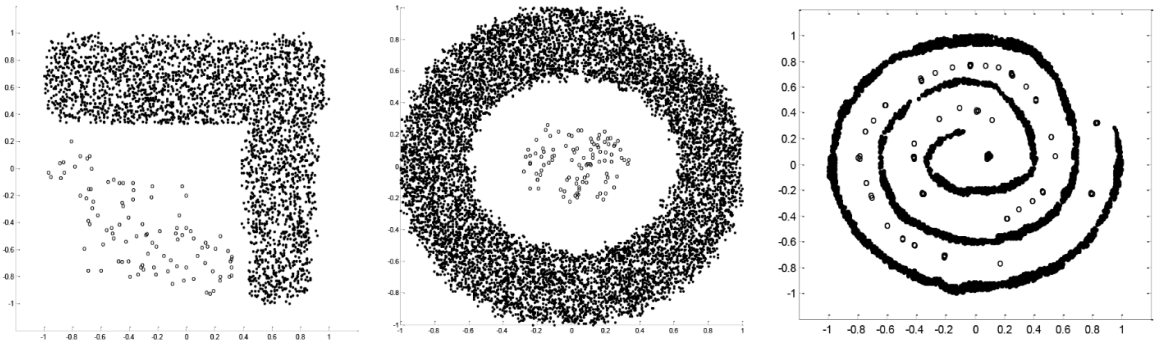
To compare the training speed of the proposed method with the other methods, the training time of the methods is measured. In addition, to evaluate the accuracy of the different methods, we consider the accuracy rate(ACC) and the geometric mean (g-mean) which can be calculated using (43) and (44), respectively. TP is the number of in-class test samples that are classified correctly while FN is the number of in-class test samples that are classified incorrectly. Similarly, TN is the number of outlier test samples that are classified correctly and FP is the number of outlier test samples that are classified incorrectly. Furthermore, to compare the testing time of the proposed method with others, the percentage of the number of support vectors out of the total number of training samples, i.e. SV%, is considered. It is noteworthy to declare that the decision function of SVDD-based classifiers is a function of support vector samples (see (31)). Thus, the number of support vectors affects the testing time and the required storage space.

$$ACC = \frac{TN + TP}{FP + FN + TP + TN} \tag{43}$$

$$g - mean = \sqrt{\frac{TP}{TP + FN} \times \frac{TN}{TN + FP}} \tag{44}$$

4.2. Parameter Tuning

Tuning SVDD parameters, i.e., the regularization parameter C and the bandwidth σ of the kernel function, is very uphill for some datasets. Fig. 5 shows some artificial datasets in which tuning SVDD parameters is hard, or even impossible in some cases, because of their complex nature. Therefore, to compare EM-SVDD with rival methods, we use the grid search approach to obtain the best parameters of each method for each dataset. In other words, to achieve the best performance of methods for each dataset, C and σ are exhaustively searched. The domains of C and σ depend on the complexity of the training data. Furthermore, the variable k which controls the number of clusters in the proposed method is also exhaustively searched in such a way that the number of clusters is big enough to correctly describe the complex datasets. However, with a large number of clusters, a proper value, i.e., a larger value, should be selected for σ to prevent overfitting.



(a) Crossover (b) Ring (c) Spiral

Figure 5: Artificial datasets - The black points: target samples. The hollow points: outlier samples.

4.3. Comparison of EM-SVDD with Other Methods on Artificial Datasets

In this experiment, a 10-fold cross validation is performed to compare EM-SVDD with the other methods on CrossOver, Spiral and Ring artificial datasets. Some details about these datasets are provided in Table 1. Table 2 shows the results of comparisons according to the average values of evaluation metrics on artificial datasets. The results illustrate that the proposed method outperforms other compared methods on dense datasets. The worst result of EM-SVDD belongs to the Spiral dataset which is sparse. The reason is that tuning of kernel parameter σ is hard for sparse datasets and so, leads to a performance reduction. One of the weaknesses of LIBSVM in comparison with EM-SVDD is that the number of final support vectors is about the half of the training samples. In addition, LIBSVM has the least training time among compared methods but not considering the distribution of data leads to a low accuracy. FSVDD is comparable with EM-SVDD from the point of support vector samples percentage. On the other hand, it is a weak method from the point of the training time and the accuracy rate of SVDD. Fig. 6 and Fig. 7 illustrate the comparison of accuracy rates and support vectors percentage of the methods.

TABLE 1
The number of in-class and outlier samples in artificial datasets

	CrossOver	**Spiral**	**Ring**
In-Class samples #	3000	5000	10000
Outliers #	100	100	100

TABLE 2
The performance of EM-SVDD in comparison with other methods on artificial datasets

Datasets	**Measurements**	**Methods**			
		SVDD[1]	**FSVDD**	**LIBSVM**	**proposed method**
CrossOver	**SV(%)**	3.33	11.44	31.12	0.89
	Training Time (s)	1.07	27.10	0.41	1.53
	ACC	100	80.20	80.82	95.38
Spiral	**SV(%)**	-	4.53	40.09	6.42
	Training Time (s)	-	23.59	1.12	18.48
	ACC	-	81.24	77.99	84.14
Ring	**SV(%)**	-	3.47	50.02	3.65
	Training Time (s)	-	45.37	5.50	11.43
	ACC	-	66.73	62.99	88.12

[1] SVDD method is unable to be performed on datasets with more than 3000 samples

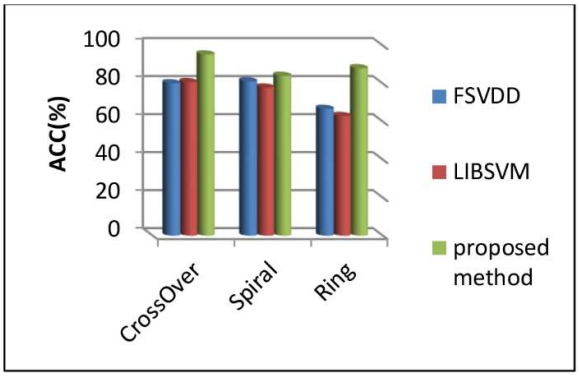


Figure 6: Comparison of the accuracy rate of the proposed method with online and canonical methods.

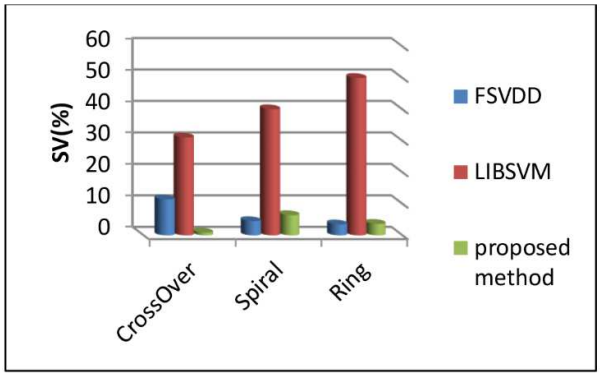


Figure 7: Comparison of the percentage of support vectors of the proposed method with online and canonical methods.

4.4. Comparison of EM-SVDD with Rival Methods on UCI Datasets

In this section, first, EM-SVDD performance is compared with standard SVDD, FSVDD and LibSVM with SVDD toolbox on 10 UCI datasets. After that, the accuracy of EM-SVDD is compared with its six related methods, including ensemble of SVDDs by bagging, ensemble of SVDDs by AdaBoost, random subspace method based ensemble of SVDDs (RSMESVDDs), clustering based ensemble of SVDDs (CESVDDs), selective ensemble of SVDDs (SESVDDs), and robust AdaBoost based ensemble of OCSVMs (H.-J. Xing & Liu, 2020).

4.4.1. Comparison of EM-SVDD with SVDD, FSVDD and LIBSVM

In this experiment, a 10-fold cross validation is performed to compare EM-SVDD with traditional SVDD, FSVDD and LIBSVM on the 10 well-known datasets from UCI Machine Learning Dataset Repository. Table 3 provides some details about selected datasets. Furthermore, the percentage of SVs, training times and accuracy rates are presented in Table 4 to compare EM-SVDD with the three mentioned methods on small and medium datasets. Both classes of each dataset are considered as the target class, respectively. From this table, it can be concluded that EM-SVDD is competitive with existing classifiers. In comparison with other methods, the proposed method achieved good time and accuracy results in small and medium datasets. It is obvious that only LIBSVM is superior to EM-SVDD in terms of training time, but its percentage of the support vectors is high. Thus, it is not a good achievement for LIBSVM. Fig. 8 illustrates the accuracy rate comparison of the methods. The results indicate that EM-SVDD has almost better accuracy rate than FSVDD and LIBSVM. Fig. 9 clearly shows that the percentage of the support vectors of EM-SVDD is generally less than all other methods.

TABLE 3
UCI Datasets information

Size	Dataset	Class#	Sample #	Feature #
Small	Sonar	2	208	60
	Liver	2	345	6
	Ionosphere	2	351	34
	Pima	2	768	8
Medium	German	2	1000	24
	Splice	2	1000	60
	Cloud	2	2048	10
Large	Football	2	4288	13
	Spambase	2	4601	58
	Mushroom	2	8124	112

The results on large datasets are summarized in the Table 5 and indicate a performance reduction. The EM-SVDD has a much better accuracy than LIBSVM and FSVDD, except in Mushrooms dataset which is not likely dense. The training time of LIBSVM is better than EM-SVDD while providing a large number of support vectors and low accuracy. The main reason that leads to the increase of training time is the time complexity of the FCM. Also, the performance can be affected by noisy and outlier data. Despite comparable accuracy rates, the training time of EM-SVDD is lower than FSVDD. The results of accuracy rates and support vector percentages are summarized in Fig. 10 and Fig. 11.

TABLE 4
The performance of EM-SVDD in comparison with the other methods on both classes of small and medium UCI Datasets

Datasets		Measurements	Methods							
			SVDD		FSVDD		LIBSVM		proposed method	
			Class 1	Class 2	Class 1	Class 2	Class 1	Class 2	Class 1	Class 2
Small	Sonar	SV(%)	55.35	66.9	83.92	74.07	20.31	42.02	18.46	14.09
		Training Time(s)	0.02	0.02	2.91	0.33	0	0	0.02	0.01
		Training ACC(%)	49.24	58.18	47	29.31	49.55	48.6	54.15	48.6
		Testing ACC(%)	80.21	75.49	79.01	51.21	75.32	50	64.16	58.03
	Liver	SV(%)	41.61	40.32	39.06	18.03	50.78	50.82	33.05	68.15
		Training Time(s)	0.03	0.04	0.11	0.49	0	0	0.03	0.1
		Training ACC(%)	100	100	60.96	55.56	67.24	70.82	100	100
		Testing ACC(%)	100	100	58.82	88.24	62.94	51.18	100	100
	Ionosphere	SV(%)	4.46	47.79	24.75	77.88	0.48	53.1	16.34	70.27
		Training Time(s)	0.02	0.03	0.45	0.78	0.01	0	0.01	0.03
		Training ACC(%)	100	100	54.83	31.67	70.84	51.33	88.01	90.51
		Testing ACC(%)	100	100	30.43	23.08	47.83	38.45	100	100
	Pima	SV(%)	52.4	28.07	32.37	31.42	50.83	40.33	49.79	30.87
		Training Time(s)	0.09	0.14	1.78	3.91	0.01	0.02	0.05	0.31
		Training ACC(%)	100	100	85.06	75.91	80	69.93	68.77	78.94
		Testing ACC(%)	100	100	74.07	75.52	45.21	54.53	62.07	79.25
Medium	German	SV(%)	28.81	-	63.59	72.32	99.84	90	48.81	46.86
		Training Time(s)	0.45	-	63.04	5.52	0.1	0.03	0.15	0.05
		Training ACC(%)	100	-	98.37	44.43	69.6	52.77	100	88.75
		Testing ACC(%)	100	-	98.59	34.48	53.52	65.52	100	89.66
	Splice	SV(%)	42.65	43.72	37.96	32.24	34.27	33.77	20.62	15.69
		Training Time(s)	0.24	0.36	3.45	3.15	0.03	0.02	0.15	0.15
		Training ACC(%)	100	100	43.21	44.43	70.47	50.84	86.02	51.46
		Testing ACC(%)	100	100	43.49	48.61	62.46	56.41	93.44	61.54
	Cloud	SV(%)	1.73	1.96	4.21	3.47	20.05	10.27	8.71	37.73
		Training Time(s)	0.51	0.27	1.38	0.91	0.07	0.07	0.2	1.39
		Training ACC(%)	100	100	90.06	91.98	70.27	80.05	98.6	92.6
		Testing ACC(%)	100	100	90.16	91.75	83.47	82.43	99.02	64.73

TABLE 5
The performance of EM-SVDD in comparison with the other methods on both classes of large UCI datasets

Datasets		Methods	Datasets					
			FSVDD		LIBSVM		proposed method	
			Class 1	Class 2	Class 1	Class 2	Class 1	Class 2
Large	Football	SV(%)	11.67	10.12	10.17	23.21	31.7	20.29
		Training Time(s)	19.1	40.25	0.18	0.47	0.55	1.38
		Training ACC(%)	78.69	92.33	50.17	50.04	94.03	83.58
		Testing ACC(%)	80.38	93.73	35.5	32.84	93.04	84.54
	Spambase	SV(%)	27.89	28.94	40.21	33.16	46.09	16.86
		Training time(s)	169.64	113.04	0.34	0.6	2.91	2.9
		Training ACC(%)	52.55	55.73	42.55	49.96	78.97	65.29
		Testing ACC(%)	50.72	54.4	43.96	48.96	78.03	58.74
	Mushrooms	SV(%)	26.41	14.19	31.15	42.28	18.48	21.34
		Training Time(s)	269.08	46.36	1.15	1.03	0.28	0.33
		Training ACC(%)	69.6	96.94	36.97	74.04	65.2	62.15
		Testing ACC(%)	69.6	96.94	38	75.77	65.56	64.8

4.4.2. Comparison of EM-SVDD with Six Related Methods

In this section, the accuracy of the proposed classifier is compared with five ensemble-based SVDD classifiers, i.e., ensemble of SVDDs by bagging, ensemble of SVDDs by AdaBoost, random subspace method based ensemble of SVDDs (RSMESVDDs), clustering based ensemble of SVDDs (CESVDDs) and selective ensemble of SVDDs (SESVDDs), and one ensemble-based OCSVM classifier, entitled robust AdaBoost based ensemble of OCSVMs (RAEOOCSVMs). To this aim, we apply all methods on four UCI datasets. In each dataset, the samples of one class are considered as in-class data and so the samples of the other class are outliers. Furthermore, for each dataset, the training set is constructed by randomly choosing 70% samples from the in-class data, while the testing set consists of the rest 30% samples of the in-class data and the whole outliers. Table 6 represents some details about the number of in-class and outlier samples in each selected dataset. Table 7 provides the average g-mean together with their corresponding standard deviations of all the methods after tuning their parameters on the four datasets (see (H. Xing & Wang, 2017) and (H.-J. Xing & Liu, 2020)). Usually, the goal of the ensemble methods is to improve the accuracy of the classifier. Thus, their base classifiers utilize almost all training samples. However, the goal of EM-SVDD is to speed up the classification on large datasets. Therefore, every base classifier is trained on a partition of the training dataset and then the results of them are aggregated. Therefore, the accuracy of EM-

SVDD can be lower than the other ensemble methods. However, the results indicate that the proposed method outperforms the other ensemble-based methods on Liver and Pima datasets. In addition, EM-SVDD shows higher accuracies on German dataset, compared to Bagging, AdaBoost and RSMESVDD. It is worth to mention that EM-SVDD outperforms OCSVM classifier on all datasets. The P-values obtained from paired T-tests which are reported in Table 7 confirm that the accuracy improvements of EM-SVDD are significant. Moreover, the reported standard deviation values show that the proposed method is more stable than the other five SVDD classifiers on all the four datasets.

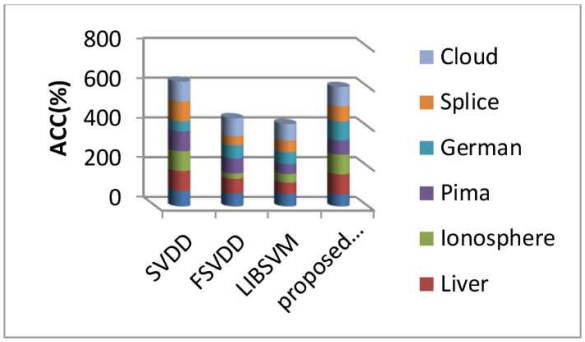


Figure 8: Comparison of the accuracy rate of the proposed method with online and canonical methods on small and medium datasets.

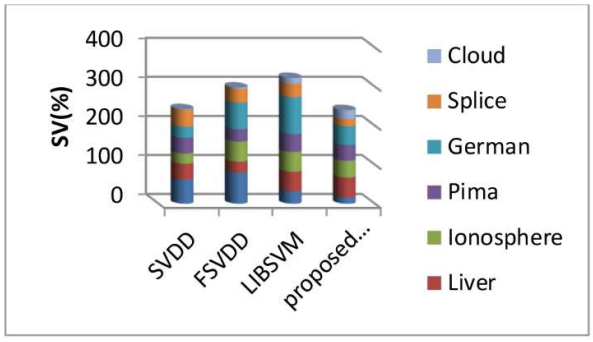


Figure 9: Comparison of support vectors percentage of the proposed method with online and canonical methods on small and medium datasets.

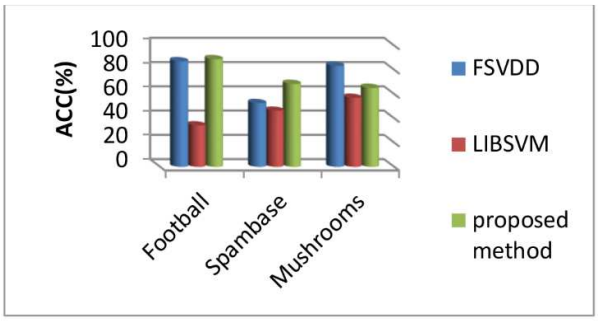


Figure 10: Comparison of the accuracy rate of the proposed method with online and canonical methods on large datasets.

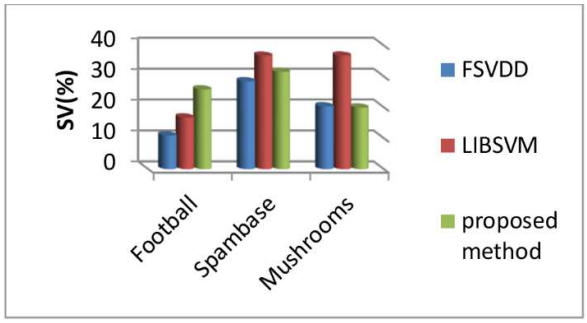


Figure 11: Comparison of support vectors percentage of the proposed method with online and canonical methods on large datasets.

TABLE 6
The number of in-class an outlier samples selected from each UCI dataset in the experiment

Datasets	In-class samples#	Outlier samples #
German	300	700
Liver	145	200
Pima	268	500
Sonar	111	97

TABLE 7
The average g-mean and standard deviation of ensemble methods on the four selected UCI datasets, together with the P-values of paired T-tests

Datasets	Methods						
	Bagging	**AdaBoost**	**RSMESVDDs**	**CESVDDs**	**SESVDDs**	**RAEOCSVMs**	**Proposed Method**
German	78.65±1.31 P=2.3105e-07	78.17±1.48 P=6.8831e-09	73.58±2.28 P=2.1966e-18	79.19±1.14 P=1.7481e-04	82.12±2.09 -	76.41±0.08 -	80.21±0.02 -
Liver	83.49±0.94 P=9.3572e-35	80.44±0.96 P=5.3355e-39	71.41±2.13 P=8.2404e-38	79.06±0.86 P=1.1050e-41	84.57±1.25 P=1.3608e-33	82.53±0.055 -	93.47±0.03 -
Pima	79.92±1.94 P=6.4676e-36	77.27±1.95 P=2.1406e-35	70.50±4.52 P=6.6404e-24	78.61±1.97 P=2.3317e-38	80.81±1.99 P=3.4493e-30	84.73±0.005 -	99.78±0.003 -
Sonar	91.56±1.60	89.36±2.14	79.30±3.45 P=9.7525e-05	90.48±3.16	92.12±1.61	78.46±0.067 -	85.13±0.04 -

4.5 Comparison of EM-SVDD with Rival Methods on an IDS Dataset

To compare the accuracy of the proposed method with Inc-SVDD and LIBSVM on large-scale datasets, we apply them on NSL-KDD dataset (http://nsl.cs.unb.ca/NSL-KDD/). The samples of this dataset are divided into five different classes, including Normal, Denial-of-Service (DoS), Probing (Probe), Remote-To-Local (R2L) and User-to-Root (U2R) which all have 41 features. The training and testing dataset descriptions and the obtained results are presented in Table 8 and Table 9, respectively. Furthermore, the samples of Normal, Dos and Probe classes are used to train the methods, respectively. As an instance, in the first experiment, the first row of Table 9, the classifiers are trained using the Normal samples of the training dataset and then they are evaluated by Normal samples of the testing dataset together with a set of randomly selected samples from all other classes. The results indicate that EM-SVDD provides comparable accuracies with the other methods ones.

TABLE 8
The number of samples of each class in NSL-KDD training and testing datasets

Dataset	Classes				
	Normal	DoS	Probe	R2L	U2R
NSL-KDD Train+	67343	45927	11656	995	52
NSL-KDD Test+	9710	7458	2422	2887	67

TABLE 9
The accuracy rate comparison of EM-SVDD in comparison with the other methods on NSL-KDD dataset

Training class	Methods		
	Inc-SVDD	LIBSVM	Proposed Method
Normal	90.44	89.63	92.25
DoS	82.74	81.19	80.41
Probe	87.58	80.32	82.12

4.6 Case Study

500px community has offered a job to research on machine learning models which can power spam detection in photos ("Machine Learning Engineer Summer Intern"). In order to illustrate the capability of the proposed method, we apply EM-SVDD classifier to "Carrots" dataset, a dataset obtained from 500px containing 1855 images. All images in this dataset are tagged as "Carrot"; however, some of them are irrelevant. The goal of the case study is to recognize irrelevant images. Therefore, each image is described by 4096 features using convolutional neural network, and then SEM-SVDD is used to classify the images. Consequently, all images that are accepted using the decision function of the SEM-SVDD are detected as in-class images. Similarly, the rejected images are detected as outliers. By tuning the SVDD parameter, C, and kernel parameter, σ, the number of outlier images can be controlled. Therefore, the best values for C and σ are obtained by a brute force search using different parameters. The results show that SEM-SVDD scores with g-mean equals 0.71 (C=0.5 and σ=0.3). Moreover, Fig. 12 is provided to illustrate the effectiveness of the parameters C and σ in the performance of the proposed method. Fig. 12-a shows some photos detected as outliers using SEM-SVDD. Improper tuning of C and σ leads to incorrect detection of outlier images. Fig. 12-b shows some outlier photos detected by improper tuning of C and σ.

5 Conclusions

In this paper, we proposed EM-SVDD and SEM-SVDD methods based on the EM technique to enhance the performance and speed up the SVDD classifier. It is well known that SVDD has a low performance when it is trained using a large population of training samples. Therefore, the proposed algorithms enhance the SVDD in order to be applicable to large-scale datasets, decrease the training time, and maintain the classification accuracy. The main contributions can be summarized as:

I. Using a combination of weighted SVDDs
II. Utilizing the EM algorithm to estimate the SVDD parameters
III. Considering the size of dataset in partitioning
IV. Considering data distribution in classification
V. Proposing a new probability density function for SVDD

Considering the convergence of the proposed method, EM-SVDD ensures the optimality of SVDD weights and support vectors coefficients, but it suffers from the high time complexity and noise sensitivity of the FCM method in the partitioning step. Thus, in the future work, to refine the performance of the proposed methods, we can replace it by a robust clustering method which dictates a lower overhead.

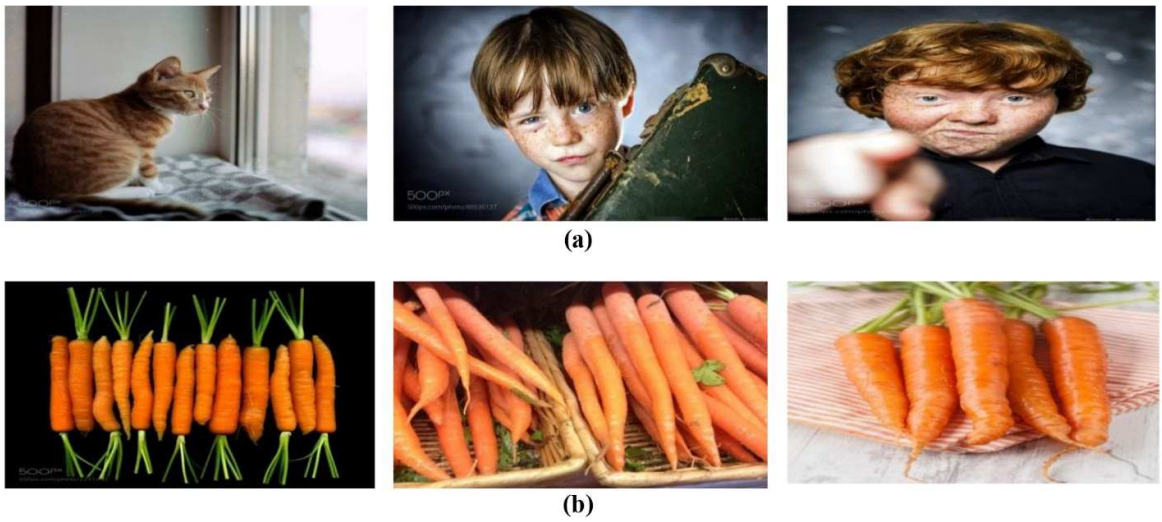


Figure 12: Spam images detected by EM-SVDD. (a) Proper tuning of C and σ. (b) Improper tuning of C and σ.

References

Alimoglu, F., & Alpaydin, E. (1997). Combining multiple representations and classifiers for pen-based handwritten digit recognition. *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, *2*, 637–640. https://doi.org/10.1109/ICDAR.1997.620583

Benkedjouh, T., Medjaher, K., Zerhouni, N., & Rechak, S. (2012). Fault prognostic of bearings by using support vector data description. *2012 IEEE Conference on Prognostics and Health Management*, 1–7. https://doi.org/10.1109/ICPHM.2012.6299511

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140. https://doi.org/10.1007/BF00058655

Chang, C.-C., & Lin, C.-J. (2011). LIBSVM. *ACM Transactions on Intelligent Systems and Technology*, *2*(3), 1–27. https://doi.org/10.1145/1961189.1961199

Chaudhuri, A., Kakde, D., Jahja, M., Xiao, W., Jiang, H., Kong, S., & Peredriy, S. (2016). Sampling method for fast training of support vector data description. *ArXiv Preprint ArXiv:1606.05382*.

Cheplygina, V., & Tax, D. M. J. (2011). Pruned Random Subspace Method for One-Class Classifiers. In *International Workshop on Multiple Classifier Systems* (pp. 96–105). Springer. https://doi.org/10.1007/978-3-642-21557-5_12

Chou, H.-Y., Lin, P.-Y., & Lin, C.-J. (2020). Dual Coordinate-Descent Methods for Linear One-Class SVM and SVDD. In *Proceedings of the 2020 SIAM International Conference on Data Mining* (pp. 181–189). Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9781611976236.21

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, *39*(1), 1–38.

Dit-Yan Yeung, & Chow, C. (2002). Parzen-window network intrusion detectors. *Object Recognition Supported by User Interaction for Service Robots*, *4*, 385–388. https://doi.org/10.1109/ICPR.2002.1047476

Dunn, J. C. (1973). A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. *Journal of Cybernetics*, *3*(3), 32–57. https://doi.org/10.1080/01969727308546046

Freund, Y., Schapire, R., & Abe, N. (1999). A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, *14*(5), 771–780.

Frey, B. J., & Dueck, D. (2007). Clustering by Passing Messages Between Data Points. *Science*, *315*(5814), 972–976. https://doi.org/10.1126/science.1136800

Fumera, G., Roli, F., & Giacinto, G. (2000). Reject option with multiple thresholds. *Pattern Recognit.*, *33*(12), 2099–2101.

Fung, G. M., & Mangasarian, O. L. (2004). A Feature Selection Newton Method for Support Vector Machine Classification. *Computational Optimization and Applications*, *28*(2), 185–202. https://doi.org/10.1023/B:COAP.0000026884.66338.df

Ge, Z., Gao, F., & Song, Z. (2011). Batch process monitoring based on support vector data description method. *Journal of Process Control*, *21*(6), 949–959. https://doi.org/10.1016/j.jprocont.2011.02.004

Hamdi, F., & Bennani, Y. (2011). Learning random subspace novelty detection filters. *The 2011 International Joint Conference on Neural Networks*, 2273–2280. https://doi.org/10.1109/IJCNN.2011.6033512

Hatami, N., & Ebrahimpour, R. (2007). Combining multiple classifiers: diversify with boosting and combining by stacking.

International Journal of Computer Science and Network Security, *7*(1), 127–131.
Hua, X., & Ding, S. (2011). Incremental learning algorithm for support vector data description. *JSW*, *6*(7), 1166–1173.
Hwang, B.-W., Kwon, S.-J., & Lee, S.-W. (2014). Facial image reconstruction from a corrupted image by support vector data description. *Computing and Informatics*, *32*(6), 1212–1228.
Jae Hyuk Shin, Boreom Lee, & Kwang Suk Park. (2011). Detection of Abnormal Living Patterns for Elderly Living Alone Using Support Vector Data Description. *IEEE Transactions on Information Technology in Biomedicine*, *15*(3), 438–448. https://doi.org/10.1109/TITB.2011.2113352
Jiang, H., Wang, H., Hu, W., Kakde, D., & Chaudhuri, A. (2019). Fast Incremental SVDD Learning Algorithm with the Gaussian Kernel. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*, 3991–3998. https://doi.org/10.1609/aaai.v33i01.33013991
Jiang, Q., Yan, X., Lv, Z., & Guo, M. (2014). Independent component analysis-based non-Gaussian process monitoring with preselecting optimal components and support vector data description. *International Journal of Production Research*, *52*(11), 3273–3286. https://doi.org/10.1080/00207543.2013.870362
Joachims, T. (1998). Making large scale SVM learning practical. *Advances in Kernel Methods - Support Vector Learning*, 169–184.
Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, *97*(1–2), 273–324. https://doi.org/10.1016/S0004-3702(97)00043-X
Krawczyk, B., Woźniak, M., & Cyganek, B. (2014). Clustering-based ensembles for one-class classification. *Information Sciences*, *264*, 182–195. https://doi.org/10.1016/j.ins.2013.12.019
Krishnan, T., & McLachlan, G. (2008). The EM algorithm and extensions. In *Wiley* (second edi).
Lal, T. N., Chapelle, O., Weston, J., & Elisseeff, A. (2006). Embedded Methods. In *Feature Extraction* (pp. 137–165). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-35488-8_6
Lange, K. (1995). A gradient algorithm locally equivalent to the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, *57*(2), 425–437. https://doi.org/10.2307/2345971
Lee, D., & Lee, J. (2007). Domain described support vector classifier for multi-classification problems. *Pattern Recognition*, *40*(1), 41–51. https://doi.org/10.1016/j.patcog.2006.06.008
Li, N., & Zhou, Z.-H. (2009). Selective Ensemble under Regularization Framework. In *International Workshop on Multiple Classifier Systems* (pp. 293–303). Springer. https://doi.org/10.1007/978-3-642-02326-2_30
Li, Q., Li, G., Han, X., Zhang, J., Liang, Y., Wang, B., Li, H., Yang, J., & Wu, C. (2014). Global prediction-based adaptive mutation particle swarm optimization. *2014 10th International Conference on Natural Computation (ICNC)*, 268–273. https://doi.org/10.1109/ICNC.2014.6975846
Li, Y. (2011). Selecting training points for one-class support vector machines. *Pattern Recognition Letters*, *32*(11), 1517–1522. https://doi.org/10.1016/j.patrec.2011.04.013
Liang, J., Liu, S., & Wu, D. (2009). Fast training of SVDD by extracting boundary targets. *Iranian Journal of Electrical and Computer Engineering*, *8*(2), 133–137.
Lorena, L. H. N., Carvalho, A. C. P. L. F., & Lorena, A. C. (2015). Filter Feature Selection for One-Class Classification. *Journal of Intelligent & Robotic Systems*, *80*(S1), 227–243. https://doi.org/10.1007/s10846-014-0101-2
Luo, J., Li, B., Wu, C., & Pan, Y. (2010). A fast SVDD algorithm based on decomposition and combination for fault detection. *IEEE ICCA 2010*, 1924–1928. https://doi.org/10.1109/ICCA.2010.5524160
Machine Learning Engineer Summer Intern. (n.d.). Retrieved September 10, 2017, from https://betalist.com/jobs/68986-machine-learning-engineer-summer-intern-at-500px
Manevitz, L. M., & Yousef, M. (2001). One-class SVMs for document classification. *Journal of Machine Learning Research*, *2*(Dec), 139–154.
Nekkaa, M., & Boughaci, D. (2015). A memetic algorithm with support vector machine for feature selection and classification. *Memetic Computing*, *7*(1), 59–73. https://doi.org/10.1007/s12293-015-0153-2
Peng, Z., Gurram, P., Kwon, H., & Yin, W. (2015). Sparse kernel learning-based feature selection for anomaly detection. *IEEE Transactions on Aerospace and Electronic Systems*, *51*(3), 1698–1716. https://doi.org/10.1109/TAES.2015.130730
Pizzi, N. J., Vivanco, R. A., & Somorjai, R. L. (2001). EvIdent™: a functional magnetic resonance image analysis system. *Artificial Intelligence in Medicine*, *21*(1–3), 263–269. https://doi.org/10.1016/S0933-3657(00)00095-6
Platt, J. C. (1999). Fast training of support vector machines using sequential minimal optimization. *Advances in Kernel Methods*, 185–208.
Prakash, J., & Singh, P. K. (2015). An effective multiobjective approach for hard partitional clustering. *Memetic Computing*, *7*(2), 93–104. https://doi.org/10.1007/s12293-014-0147-5
Sanchez-Hernandez, C., Boyd, D. S., & Foody, G. M. (2007). One-Class Classification for Mapping a Specific Land-Cover Class: SVDD Classification of Fenland. *IEEE Transactions on Geoscience and Remote Sensing*, *45*(4), 1061–1073. https://doi.org/10.1109/TGRS.2006.890414
Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., & Williamson, R. C. (2001). Estimating the Support of a High-Dimensional Distribution. *Neural Computation*, *13*(7), 1443–1471. https://doi.org/10.1162/089976601750264965
Sundberg, R. (1972). *Maximum likelihood theory and applications for distributions generated when observing a function*

of an exponential variable. Dissertation, Stockholm University.
Tax, D. M. J. (2001). *One-class classification: Concept-learning in the absence of counter-examples*. Dissertation, Delft University of Technology.
Tax, David M.J., & Duin, R. P. W. (2004). Support Vector Data Description. *Machine Learning*, *54*(1), 45–66. https://doi.org/10.1023/B:MACH.0000008084.60811.49
Tax, David M J, & Duin, R. P. W. (2001). Combining One-Class Classifiers. In *International Workshop on Multiple Classifier Systems* (pp. 299–308). Springer. https://doi.org/10.1007/3-540-48219-9_30
Tax, David M J, & Duin, R. P. W. (1999). Data domain description using support vectors. *ESANN*, *99*, 251–256.
Vapnik, V. N. (1999). An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, *10*(5), 988–999. https://doi.org/10.1109/72.788640
Vergara, J. R., & Estévez, P. A. (2014). A review of feature selection methods based on mutual information. *Neural Computing and Applications*, *24*(1), 175–186. https://doi.org/10.1007/s00521-013-1368-0
Wang, C.-D., & Lai, J. (2013). Position regularized Support Vector Domain Description. *Pattern Recognition*, *46*(3), 875–884. https://doi.org/10.1016/j.patcog.2012.09.018
Wang, D., & Tan, X. (2013). Centering SVDD for Unsupervised Feature Representation in Object Classification. In *International Conference on Neural Information Processing* (pp. 376–383). Springer. https://doi.org/10.1007/978-3-642-42051-1_47
Wang, J., Liu, W., Qiu, K., Xiong, H., & Zhao, L. (2019). Dynamic hypersphere SVDD without describing boundary for one-class classification. *Neural Computing and Applications*, *31*(8), 3295–3305. https://doi.org/10.1007/s00521-017-3277-0
Wang, S., Yu, J., Lapira, E., & Lee, J. (2013). A modified support vector data description based novelty detection approach for machinery components. *Applied Soft Computing*, *13*(2), 1193–1205. https://doi.org/10.1016/j.asoc.2012.11.005
Weston, J., Mukherjee, S., Chapelle, O., Pontil, M., Poggio, T., & Vapnik, V. (2000). Feature selection for SVMs. *Proceedings of the 13th International Conference on Neural Information Processing Systems*, 647–653.
Wu, T., Liang, Y., Varela, R., Wu, C., Zhao, G., & Han, X. (2016). Self-adaptive SVDD integrated with AP clustering for one-class classification. *Pattern Recognition Letters*, *84*, 232–238. https://doi.org/10.1016/j.patrec.2016.10.009
Xiao, Y., Liu, B., & Cao, L. (2010). K-farthest-neighbors-based concept boundary determination for support vector data description. *Proceedings of the 19th ACM International Conference on Information and Knowledge Management - CIKM '10*, 1701. https://doi.org/10.1145/1871437.1871708
Xing, H.-J., & Li, L.-F. (2020). Robust least squares one-class support vector machine. *Pattern Recognition Letters*, *138*, 571–578. https://doi.org/10.1016/j.patrec.2020.09.005
Xing, H.-J., & Liu, W.-T. (2020). Robust AdaBoost based ensemble of one-class support vector machines. *Information Fusion*, *55*, 45–58. https://doi.org/10.1016/j.inffus.2019.08.002
Xing, H., & Wang, X. (2017). Selective ensemble of SVDDs with Renyi entropy based diversity measure. *Pattern Recognition*, *61*, 185–196. https://doi.org/10.1016/j.patcog.2016.07.038
Xu, J., Yao, J., & Ni, L. (2011). Fault detection based on SVDD and cluster algorithm. *2011 International Conference on Electronics, Communications and Control (ICECC)*, 2050–2052. https://doi.org/10.1109/ICECC.2011.6067662
Yan, Y.-T., Zhang, Y.-P., Zhang, Y.-W., & Du, X.-Q. (2017). A selective neural network ensemble classification for incomplete data. *International Journal of Machine Learning and Cybernetics*, *8*(5), 1513–1524. https://doi.org/10.1007/s13042-016-0524-0
Yang, Y., Pierce, T., & Carbonell, J. (1998). A study of retrospective and on-line event detection. *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '98*, 28–36. https://doi.org/10.1145/290941.290953
Zhang, J., Lu, J., & Zhang, G. (2011). Combining one class classification models for avian influenza outbreaks. *2011 IEEE Symposium on Computational Intelligence in Multicriteria Decision-Making (MDCM)*, 190–196. https://doi.org/10.1109/SMDCM.2011.5949278
Zhang, L., & Zhou, W.-D. (2011). Sparse ensembles using weighted combination methods based on linear programming. *Pattern Recognition*, *44*(1), 97–106. https://doi.org/10.1016/j.patcog.2010.07.021
Zhou, Z.-H., Wu, J., & Tang, W. (2002). Ensembling neural networks: Many could be better than all. *Artificial Intelligence*, *137*(1–2), 239–263. https://doi.org/10.1016/S0004-3702(02)00190-X