Fast Incremental SVDD Learning Algorithm with the Gaussian Kernel

Hansi Jiang, Haoyu Wang, Wenhao Hu, Deovrat Kakde, Arin Chaudhuri
SAS Institute Inc.
100 SAS Campus Drive
Cary, North Carolina 27513
{Hansi.Jiang; Haoyu.Wang; Wenhao.Hu; Dev.Kakde; Arin.Chaudhuri}@sas.com

Abstract

Support vector data description (SVDD) is a machine learning technique that is used for single-class classification and outlier detection. The idea of SVDD is to find a set of support vectors that defines a boundary around data. When dealing with online or large data, existing batch SVDD methods have to be rerun in each iteration. We propose an incremental learning algorithm for SVDD that uses the Gaussian kernel. This algorithm builds on the observation that all support vectors on the boundary have the same distance to the center of sphere in a higher-dimensional feature space as mapped by the Gaussian kernel function. Each iteration involves only the existing support vectors and the new data point. Moreover, the algorithm is based solely on matrix manipulations; the support vectors and their corresponding Lagrange multiplier α_i's are automatically selected and determined in each iteration. It can be seen that the complexity of our algorithm in each iteration is only $O(k^2)$, where k is the number of support vectors. Experimental results on some real data sets indicate that FISVDD demonstrates significant gains in efficiency with almost no loss in either outlier detection accuracy or objective function value.

1 Introduction

Much effort has been made to detect faults and state shifts in industrial machines through monitoring data sensors. Successful fault diagnosis reduces cost of maintenance and improves both worker and machine efficiency. In machine learning, fault diagnosis can be viewed as an outlier detection problem. Support vector data description (SVDD), a machine learning technique that is used for single-class classification and outlier detection, is similar to support vector machine (SVM). SVDD was first introduced in Tax and Duin (2004), although the concept of using SVM to detect novelty was introduced in Schölkopf et al. (2000). SVDD is used in domains where the majority of data belongs to a single class, or when one of the classes is significantly undersampled. The SVDD algorithm builds a flexible boundary around the target class data; this data boundary is characterized by observations that are designated as support vectors. Having the advantage that no assumptions about the distribution of outliers need to be made, SVDD can describe the shape of the target class without prior knowledge of the specific data distribution and can flag observations that fall outside the data boundary as potential outliers. In the case of machine monitoring, data on the normal working conditions of a machine are in abundance, whereas information from outlier system failures are few. By using SVDD on the well-sampled target class, one can obtain a boundary around the distribution of normal working data, and subsequently capture the outlier points where the machine is faulty.

Traditional batch methods of SVDD typically pursue a global optimal solution of the SVDD problem; they suffer from low efficiency by considering all available data points. Moreover, these methods are usually ineffective when handling streaming data because the entire algorithm must be rerun with each incoming data point. In contrast, incremental methods deal with large or streaming data efficiently by focusing on smaller portions of the original optimization problem, as in Syed et al. (1999). Online variants of SVDD concentrate only on the current support vector set with incoming data.

Cauwenberghs and Poggio (2001) give an incremental and decremental training algorithm for SVM. Their method, also called the C&P algorithm, provides an exact solution for training data and one new data point. Tax and Laskov (2003) use a numerical method to solve incremental SVM, and they describe the relationship between incremental SVM and online SVDD. Their research was extended in Laskov et al. (2006), which provides complete learning algorithms for incremental SVM and SVDD.

The algorithm given in Laskov et al. (2006) updates weights of each support vector based on the fact that Karush-Kuhn-Tucker (KKT) conditions must be satisfied before and after a new data point comes in. Consequently, all data points must be kept to pursue an objective value closer to the global optimal value. Furthermore, a kernel matrix must be calculated every update, which can be memory-consuming and slow for large data.

These issues are handled by the algorithm that we propose: fast incremental support vector data description (FISVDD). One of the most important properties of support vectors is that in the most simplified form of SVDD they all have the same distance to the center of a sphere. A similar property remains even when the problem is generalized to flexible boundaries. This property is at the core of FISVDD.

Unlike the method in Laskov et al. (2006), FISVDD uses only matrix manipulations to find interior points and support vectors, and it is highly efficient in detecting outliers. It can be used either as a batch method or as an online method. It can be seen that the complexity of key parts of FISVDD is $O(k^2)$, where k is the number of support vectors. By Kakde et al. (2017), the number of support vectors should be much less than the number of observations in order to avoid overfitting.

The rest of the paper is organized as follows. In Section 2, we introduce the SVDD problem in Tax and Duin (2004). In Section 3, we state some theoretical support for FISVDD. In Section 4, the FISVDD algorithm is introduced and explained. In Section 5, we discuss several important issues in implementing FISVDD. In Section 6, FISVDD is applied to some data sets and compared with other methods. Finally, in Section 7, we give our conclusions.

In this paper we follow traditional linear algebra notation. Bold capital letters stand for matrices, and bold small letters stand for vectors. Specifically, matrix $\mathbf{A}$ is used as a Gaussian kernel matrix, and $\mathbf{A}_k$ is the Gaussian kernel matrix in the kth iteration. The vector $\mathbf{x} > \mathbf{0}$ stands for a positive vector, and $\mathbf{x} \geq \mathbf{0}$ stands for a nonnegative vector.

2 The SVDD Problem

The SVDD problem is first discussed by Tax and Duin (2004). The idea of SVDD is to find support vectors and use them to define a boundary around data. If a testing data point lies outside the boundary, it is classified as an outlier; otherwise, it is classified as normal data. The simplest form of a boundary is a sphere. For a set of data points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$, the mathematical formulation of the problem is to find a nonnegative vector $\boldsymbol{\alpha}$ that contains Lagrange multipliers for all data points, $\|\boldsymbol{\alpha}\|_1 = 1$, such that the following is maximized:

$$L = \sum_{i=1}^{n} \alpha_i \langle \mathbf{x}_i, \mathbf{x}_i \rangle - \sum_{i,j} \alpha_i \alpha_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle. \quad (2.1)$$

Here $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ is the inner product of $\mathbf{x}_i$ and $\mathbf{x}_j$. According to Tax and Duin (2004), there are three possibilities for each data point. The $\mathbf{x}_i$'s that have zero α_i's are *interior points*. The $\mathbf{x}_i$'s for which $0 < \alpha_i < C$ for a preselected $0 < C \leq 1$ lie on the boundary and are called *support vectors*. The $\mathbf{x}_i$'s for which $\alpha_i = C$ are outliers (also called *bounded support vectors*, or bsv, in Ben-Hur et al. (2001)). In this paper, we assume there are no outliers in the training phase, so we set $C = 1$. One example of where our algorithm would be useful is when there is a known period during which the incoming data are normal, such as streaming sensor data from machines or vehicles operating under normal conditions. Then the model can be used to detect abnormal states. To determine whether a new data point $\mathbf{z}$ lies inside the boundary, first the distance between $\mathbf{z}$ and the center of the sphere, $\mathbf{a}$, is calculated:

$$d^2(\mathbf{z}) = \|\mathbf{z} - \mathbf{a}\|^2 = \langle \mathbf{z}, \mathbf{z} \rangle - 2 \sum_i \alpha_i \langle \mathbf{z}, \mathbf{x}_i \rangle + \sum_{i,j} \alpha_i \alpha_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle. \quad (2.2)$$

This distance is then compared to the radius of the sphere for any support vector $\mathbf{x}_k$:

$$R^2 = \langle \mathbf{x}_k, \mathbf{x}_k \rangle - 2 \sum_i \alpha_i \langle \mathbf{x}_k, \mathbf{x}_i \rangle + \sum_{i,j} \alpha_i \alpha_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle. \quad (2.3)$$

A test data point $\mathbf{z}$ is accepted if $d^2 \leq R^2$, and it is classified as an outlier if $d^2 > R^2$. This check is also called *scoring*. It is easy to derive the conclusion that scoring is equivalent to checking whether the new data point violates the current KKT conditions.

A kernel function is needed to draw a more flexible boundary around data in order to avoid underfitting. By Tax and Duin (2004), using a kernel function is equivalent to implicitly mapping data points to a higher feature space. Usually the Gaussian kernel,

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}), \quad (2.4)$$

is preferred (Ben-Hur et al. 2001; Laskov et al. 2006; Gu et al. 2015), and the Gaussian kernel bandwidth σ must be selected beforehand. There are some papers that discuss how to choose a proper Gaussian kernel bandwidth (Evangelista, Embrechts, and Szymanski 2007; Xiao et al. 2014; Kakde et al. 2017). Throughout this paper, it is assumed that the Gaussian similarity is used and that a proper Gaussian kernel bandwidth σ has been chosen such that the number of support vectors is much less than the number of observations. As stated in Section 5, FISVDD has protections even if a bad bandwidth is provided. With the Gaussian kernel function, the objective function Eq. 2.1 can be simplified to minimizing

$$L = \sum_{i,j} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j), \quad (2.5)$$

because $K(\mathbf{x}_i, \mathbf{x}_i) = 1$, $\|\boldsymbol{\alpha}\|_1 = 1$, and $\boldsymbol{\alpha}$ is nonnegative.

Eq. 2.5 can also be expressed in matrix form:

$$L = \boldsymbol{\alpha}^T \mathbf{A} \boldsymbol{\alpha}, \quad (2.6)$$

where $\mathbf{A}$ is a Gaussian similarity matrix for all support vectors and $\boldsymbol{\alpha} > \mathbf{0}$. Formulas Eq. 2.2 and Eq. 2.3 then become as follows, respectively:

$$d^2(\mathbf{z}) = 1 - 2 \sum_i \alpha_i K(\mathbf{z}, \mathbf{x}_i) + \sum_{i,j} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j), \quad (2.7)$$

$$R^2 = 1 - 2 \sum_i \alpha_i K(\mathbf{x}_k, \mathbf{x}_i) + \sum_{i,j} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j). \quad (2.8)$$

Note that to determine whether a test data point $\mathbf{z}$ should be accepted, one can compute only

$$Q(\mathbf{z}) = (d^2(\mathbf{z}) - R^2)/2 = \sum_i \alpha_i K(\mathbf{x}_k, \mathbf{x}_i) - \sum_i \alpha_i K(\mathbf{z}, \mathbf{x}_i). \quad (2.9)$$

$Q(\mathbf{z}) \leq 0$ means that $\mathbf{z}$ is an interior point. It is worth mentioning that all support vectors satisfy $d^2 = R^2$, although they might have different α_i's.

3 Theoretical Foundations

Here we state and prove several theorems necessary for later discussion. First, we state a lemma in Smola and Schölkopf (1998) that a Gaussian similarity matrix has full rank. A direct conclusion of the lemma is that a Gaussian similarity matrix is symmetric positive definite (spd).

Lemma 1. *Suppose $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$ are distinct points and $\sigma \neq 0$. Then their Gaussian similarity matrix $\mathbf{A}$ formed with Eq. 2.4 has full rank.*

Lemma 1 implies that $\mathbf{A}$ is spd and its inverse exists. Next, we state lemmas to obtain $\mathbf{A}_{k+1}^{-1}$ if $\mathbf{A}_k^{-1}$ is known and vice versa. In FISVDD, we need to update the inverse of the similarity matrix when a new data point comes in. The proof involves only matrix calculations and is skipped.

Lemma 2. *Suppose $\mathbf{A}_k$ and $\mathbf{A}_{k+1}$ are both Gaussian similarity matrices and*

$$\mathbf{A}_{k+1} = \begin{bmatrix} \mathbf{A}_k & \mathbf{v} \\ \mathbf{v}^T & 1 \end{bmatrix}. \tag{3.1}$$

If $\mathbf{A}_k^{-1}$ is known, then $\mathbf{A}_{k+1}^{-1}$ is given by

$$\mathbf{A}_{k+1}^{-1} = \begin{bmatrix} \mathbf{A}_k^{-1} + \mathbf{p}\mathbf{p}^T/\beta & -\mathbf{p}/\beta \\ -\mathbf{p}^T/\beta & 1/\beta \end{bmatrix}, \tag{3.2}$$

where $\mathbf{p} = \mathbf{A}_k^{-1}\mathbf{v}$ and $\beta = 1 - \mathbf{v}^T\mathbf{A}_k^{-1}\mathbf{v} = 1 - \mathbf{v}^T\mathbf{p}$.

Lemma 2 provides a method to compute $\mathbf{A}_{k+1}^{-1}$ by using $\mathbf{A}_k^{-1}$ and an incremental vector $\mathbf{v}$. Note that to compute $\mathbf{A}_{k+1}^{-1}$, we only need to compute $\mathbf{p} = \mathbf{A}_k^{-1}\mathbf{v}$. Also note that β is the Schur complement (Meyer 2000) of $\mathbf{A}_k^{-1}$ in $\mathbf{A}_{k+1}^{-1}$. Since $\mathbf{A}_{k+1}$ is spd, β is positive (Gallier 2010). The inverse of Lemma 2 is straightforward and shown below.

Lemma 3. *Suppose $\mathbf{A}_{k+1}$ is spd and its inverse is given by*

$$\mathbf{A}_{k+1}^{-1} = \begin{bmatrix} \mathbf{P}_{k\times k} & \mathbf{u} \\ \mathbf{u}^T & \lambda \end{bmatrix}. \tag{3.3}$$

Then the inverse of $\mathbf{A}_k$ is

$$\mathbf{A}_k^{-1} = \mathbf{P} - \mathbf{u}\mathbf{u}^T/\lambda. \tag{3.4}$$

Lemma 2 and Lemma 3 together play an essential role in FISVDD to increase efficiency. It can be seen from the lemmas that only $O(k^2)$ multiplications are needed to obtain the updated matrix inverse. Next, we prove that if a positive solution is obtained for the linear system $\mathbf{A}\boldsymbol{\alpha} = \mathbf{e}$, then all data points in the system are support vectors. This is from the property that all support vectors satisfy $d^2 = R^2$.

Theorem 4. *A set of data points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$ are all support vectors if and only if*

$$\mathbf{A}_k\boldsymbol{\alpha} = \mathbf{e} \tag{3.5}$$

has a positive solution, where $\mathbf{e}$ indicates a vector that contains all 1's with proper dimension.

Proof. Suppose that $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$ are all support vectors. Then they all satisfy $d^2(\mathbf{x}_i) = R^2$ in Eq. 2.9, and thus the $d^2(\mathbf{x}_i)$'s are all equal. From Eq. 2.7, the middle terms,

$$\sum_i \alpha_i K(\mathbf{z}, \mathbf{x}_i), \tag{3.6}$$

are all equal for any support vector $\mathbf{z}$. Putting Eq. 3.6 together for all support vectors results in the left-hand side of Eq. 3.5. Therefore, Eq. 3.5 has a positive solution. On the other hand, Eq. 3.5 implies that all $\mathbf{x}_i$'s satisfy $d^2(\mathbf{x}_i) = R^2$ and thus are all support vectors. □

If a new data point $\mathbf{x}_{k+1}$ is added to the existing support vector set but the $(k+1)$th position in the solution to the linear system $\mathbf{A}_{k+1}\boldsymbol{\alpha} = \mathbf{e}$ is not positive, then the new data point is an interior point. This is proven in the next theorem.

Theorem 5. *Suppose data points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$ form a support vector set. Then a new data point $\mathbf{x}_{k+1}$ is an interior point if and only if $\mathbf{A}_{k+1}\boldsymbol{\alpha} = \mathbf{e} \Rightarrow \alpha_{k+1} \leq 0$.*

Proof. Suppose that $\mathbf{A}_{k+1}\boldsymbol{\alpha} = \mathbf{e} \Rightarrow \alpha_{k+1} \leq 0$. By Lemma 2, we have

$$\alpha_{k+1} = [\mathbf{A}_{k+1}^{-1}\mathbf{e}]_{k+1} = \begin{bmatrix} -\mathbf{p}^T/\beta & 1/\beta \end{bmatrix} \mathbf{e}. \tag{3.7}$$

Because $\alpha_{k+1} \leq 0$, we have

$$\alpha_{k+1} = \frac{1 - \mathbf{e}^T\mathbf{A}_k^{-1}\mathbf{v}}{1 - \mathbf{v}^T\mathbf{A}_k^{-1}\mathbf{v}} \leq 0. \tag{3.8}$$

Because $\beta = 1 - \mathbf{v}^T\mathbf{A}_k^{-1}\mathbf{v} > 0$, we have

$$1 - \mathbf{e}^T\mathbf{A}_k^{-1}\mathbf{v} \leq 0. \tag{3.9}$$

We want to prove that $d^2 - R^2 \leq 0$ for $\mathbf{x}_{k+1}$. Note that

$$\begin{aligned} (d^2 - R^2)/2 &= \boldsymbol{\alpha}_k^T\mathbf{A}_{k(*i)} - \boldsymbol{\alpha}_k^T\mathbf{v} \\ &= (\mathbf{A}_k^{-1}\mathbf{e})^T\mathbf{A}_{k(*i)} - (\mathbf{A}_k^{-1}\mathbf{e})^T\mathbf{v} \\ &= \mathbf{e}^T\mathbf{A}_k^{-1}\mathbf{A}_{k(*i)} - \mathbf{e}^T\mathbf{A}_k^{-1}\mathbf{v} \\ &= 1 - \mathbf{e}^T\mathbf{A}_k^{-1}\mathbf{v}, \end{aligned} \tag{3.10}$$

where $\mathbf{A}_{k(*i)}$ is the ith column of $\mathbf{A}_k$. By Eq. 3.9, we have $d^2 - R^2 \leq 0$.

On the other hand, suppose $\mathbf{x}_{k+1}$ is strictly inside the boundary. Then we have

$$(d^2 - R^2)/2 = 1 - \mathbf{e}^T\mathbf{A}_k^{-1}\mathbf{v} \leq 0. \tag{3.11}$$

Then

$$\alpha_{k+1} = \frac{1 - \mathbf{e}^T\mathbf{A}_k^{-1}\mathbf{v}}{1 - \mathbf{v}_k^T\mathbf{A}_k^{-1}\mathbf{v}} \leq 0. \tag{3.12}$$

□

Theorem 5 says that if we put a new data point $\mathbf{x}_i$ into an existing support vector set to form an expanded set and the $(k+1)$th position in the solution to the expanded system $\mathbf{A}_{k+1}\boldsymbol{\alpha} = \mathbf{e}$ is less than 0, then $\mathbf{x}_i$ is an interior point and thus can be ignored. Because we can permute the rows and columns in $\mathbf{A}_{k+1}^{-1}$, by Theorem 5 if $\alpha_i \leq 0$ for $1 \leq i \leq k$, we can take $\mathbf{x}_i$ out of the expanded set and solve the shrunken $k \times k$ linear system. We can continue shrinking the system until there are no negative entries in $\boldsymbol{\alpha}$; then a support vector set is obtained. We summarize this shrinking step in the next corollary.

Corollary 6. *A data point $\mathbf{x}_i$ is an interior point if and only if $\mathbf{A}_{k+1}\boldsymbol{\alpha} = \mathbf{e} \Rightarrow \alpha_i \leq 0$ and the shrunken $k \times k$ linear system has a positive solution.*

Finally, we state and prove an observation that relates the objective function value, the 1-norm of the *unnormalized* $\boldsymbol{\alpha}$ vector, and the scoring threshold. The observation is substantial for implementing FISVDD. With it a lot of unnecessary computations can be saved. This observation can be also used to make sure that the objective function value in FISVDD is not larger than the objective function value obtained in the previous iteration so the FISVDD model is improved.

Corollary 7. *The objective function value in Eq. 2.6 with positive* $\boldsymbol{\alpha}$, $\|\boldsymbol{\alpha}\|_1 = 1$, *satisfies*

$$L = \frac{1}{\|\boldsymbol{\alpha}_0\|_1}, \tag{3.13}$$

where $\boldsymbol{\alpha} = \boldsymbol{\alpha}_0/\|\boldsymbol{\alpha}_0\|_1$. *Moreover, it holds that*

$$L = \sum_i \alpha_i K(\mathbf{z}, \mathbf{x}_i), \tag{3.14}$$

where the $\mathbf{x}_i$*'s are the support vectors and* $\mathbf{z}$ *is any one of the support vectors.*

Proof. To prove Eq. 3.13, note that by Theorem 4, $\boldsymbol{\alpha}_0$ satisfies $\mathbf{A}\boldsymbol{\alpha}_0 = \mathbf{e}$. Then

$$\begin{aligned} L = \boldsymbol{\alpha}^T \mathbf{A} \boldsymbol{\alpha} &= \frac{\boldsymbol{\alpha}_0^T}{\|\boldsymbol{\alpha}_0\|_1} \mathbf{A} \frac{\boldsymbol{\alpha}_0}{\|\boldsymbol{\alpha}_0\|_1} \\ &= \frac{\boldsymbol{\alpha}_0^T \mathbf{e}}{\|\boldsymbol{\alpha}_0\|_1^2} = \frac{\|\boldsymbol{\alpha}_0\|_1}{\|\boldsymbol{\alpha}_0\|_1^2} = \frac{1}{\|\boldsymbol{\alpha}_0\|_1}. \end{aligned} \tag{3.15}$$

To prove Eq. 3.14, note that $\sum_i \alpha_i K(\mathbf{z}, \mathbf{x}_i)$ is the first term of the right-hand side of Eq. 2.9. So proving Eq. 3.14 is equivalent to proving

$$\sum_{i,j} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) = \sum_i \alpha_i K(\mathbf{z}, \mathbf{x}_i), \tag{3.16}$$

where $\mathbf{x}_i$, $\mathbf{x}_j$ are support vectors, and $\mathbf{z}$ is any one of the support vectors. The following equation can be derived:

$$\begin{aligned} \sum_{i,j} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) &= \sum_j \alpha_j \Big(\sum_i \alpha_i K(\mathbf{x}_i, \mathbf{x}_j) \Big) \\ &= \Big(\sum_i \alpha_i K(\mathbf{z}, \mathbf{x}_i) \Big) \Big(\sum_j \alpha_j \Big) \\ &= \sum_i \alpha_i K(\mathbf{z}, \mathbf{x}_i). \end{aligned} \tag{3.17}$$

The second equality is derived from the fact that the term in parentheses is a constant for any support vector $\mathbf{x}_j$, and the third equality is derived from the fact that the sum of all α_i's is 1. □

Corollary 7 shows a direct relationship between the objective function value, the 1-norm of the solution vector to the linear system $\mathbf{A}\boldsymbol{\alpha} = \mathbf{e}$, and the scoring threshold. The objective function value is a very important term of an SVDD model and can be requested by the user at any time. When the solution vector of the linear system is derived, the inverse of its 1-norm directly gives the objective function value, and the calculations in Eq. 2.6 are avoided. At the same time, L is also the scoring threshold for the current model. Only the second term in Eq. 2.9 needs to be computed when a new data point needs to be scored. The results from Corollary 7 help make our FISVDD algorithm more efficient.

4 Fast Incremental SVDD Learning Algorithm

We propose a fast incremental algorithm of SVDD (FISVDD). The central idea of FISVDD is to minimize the objective function (2.6) by quickly updating the inverse of similarity matrices in each iteration. Suppose that we begin with a support vector set $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$. When a new data point $\mathbf{x}_{k+1}$ comes in, by Theorem 4 the linear system $\mathbf{A}_{k+1}\boldsymbol{\alpha} = \mathbf{e}$ will have a positive solution if the $k+1$ data points form a new support vector set, and the normalized $\boldsymbol{\alpha}$ vector gives the α_i's. However, if at least one of the entries in the solution is negative, that indicates there is at least one interior point in the set. Then we are able to drop the negative α_i that has the largest $|d^2 - R^2|$ magnitude and solve the shrunken $k \times k$ linear system. If the system has a positive solution, then we have found a support vector set. Otherwise, we can continue to drop the next negative α_i that has the largest $|d^2 - R^2|$ magnitude and solve the $(k-1)\times(k-1)$ linear system, and so on. It is worth noting that if more than one variable is dropped from the system, the dropped data points should be re-scored against the new boundary to determine whether the KKT conditions are violated. If the KKT conditions are violated, then the system will expand again. We provide details below.

The FISVDD Algorithm

The FISVDD algorithm is shown in Algorithm 3. It contains three parts of FISVDD: expanding (which is shown in Algorithm 1), shrinking (which is shown in Algorithm 2), and bookkeeping.

Stage 1, Expanding When a new data point $\mathbf{x}_{k+1}$ comes in, it is scored to determine whether it falls in the interior. If so, it is immediately discarded. Otherwise, it is combined with existing support vectors to form an expanded set. The corresponding inverse matrix of the similarity matrix and its row sums are then updated by Lemma 2. If all row sums are positive, then $\mathbf{x}_{k+1}$ is another support vector and the normalized $\boldsymbol{\alpha}$ vector contains the updated α_i's. If $\alpha_{k+1} \leq 0$, then $\mathbf{x}_{k+1}$ is taken out of the expanded set and the support vector set returns to the previous set. If $\alpha_{k+1} > 0$ but there is at least one $\alpha_i \leq 0$, then there is at least one interior point in the expanded set and the shrinking step is called. The expanding step is given in Algorithm 1.

Stage 2, Shrinking If $\alpha_{k+1} > 0$ but at least one $\alpha_i < 0$, then at least one existing support vector in the support vector set has become an interior point. We need to identify and discard such vectors. By Corollary 6, we can shrink the support vector set one vector at a time until a positive $\boldsymbol{\alpha}$ is obtained. It is possible that there are several negative entries in the $\boldsymbol{\alpha}$ vector, but after taking out one negative entry all other entries are positive. Hence, it is recommended to

Algorithm 1 Expand

Input: $\mathbf{x}_{k+1}, \boldsymbol{\alpha}, \mathrm{SV}, \sigma, \mathbf{A}^{-1}$
$\mathbf{v} \leftarrow K(\mathbf{x}_{k+1}, \mathrm{SV}, \sigma)$
$\mathbf{A}_{\mathrm{old}}^{-1} \leftarrow \mathbf{A}^{-1}$
$\mathbf{A}^{-1} \leftarrow$ Eq. 3.2
$\boldsymbol{\alpha}_{\mathrm{old}} \leftarrow \boldsymbol{\alpha}$
$\boldsymbol{\alpha} \leftarrow$ row sums of $\mathbf{A}^{-1}$
if $\alpha_{k+1} \leq 0$ **then**
  $\mathbf{A}^{-1} \leftarrow \mathbf{A}_{\mathrm{old}}^{-1}$
  $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha}_{\mathrm{old}}$
else
  SV $\leftarrow$ SV + $\mathbf{x}_{k+1}$
end if
Return: $\boldsymbol{\alpha}, \mathrm{SV}, \mathbf{A}^{-1}$

take out one vector at a time rather than taking out several vectors. Moreover, taking out several vectors at once slows the algorithm because then we need to calculate the inverse of matrices whose rank is larger than 1. Although there is no certain way of choosing which vector to remove first, in FISVDD we choose the negative α_i that has the largest magnitude. From Eq. 3.8 and Eq. 3.10 and permuting columns and rows in $\mathbf{A}_{k+1}$, we have

$$\alpha_{k+1} = \frac{d^2 - R^2}{2(1 - \mathbf{v}^T \mathbf{A}_k^{-1} \mathbf{v})}, \tag{4.1}$$

where α_{k+1} is the α_i of interest permuted to the $(k+1)$th position. It can be seen from Eq. 4.1 that if the denominators of the data points that have negative α_i's are close, then a data point that has a larger $|\alpha_i|$ tends to have a larger $|d^2 - R^2|$, which means it lies farther from the boundary. Intuitively, a data point farther from the boundary is more likely to be a true interior point. Although not guaranteed, the data point farthest from the boundary is typically the one we want to remove first.

Algorithm 2 Shrink

Input: $\boldsymbol{\alpha}, \mathrm{SV}, \mathbf{A}^{-1}, \mathrm{Backup}$
flag $\leftarrow 1$
while flag $= 1$ **do**
  $p \leftarrow \arg\min \boldsymbol{\alpha}$
  Backup $\leftarrow$ Backup $+ \mathbf{x}_p$
  SV $\leftarrow$ SV $- \mathbf{x}_p$
  $\mathbf{A}^{-1} \leftarrow$Eq. 3.4
  $\boldsymbol{\alpha} \leftarrow$ row sums of $\mathbf{A}^{-1}$
  if $\min \boldsymbol{\alpha} > 0$ **then**
    flag $\leftarrow 0$
  end if
end while
Return $\boldsymbol{\alpha}, \mathrm{SV}, \mathbf{A}^{-1}, \mathrm{Backup}$

Bookkeeping When the shrinking algorithm is performed, some of the previous support vectors are taken out of the support vector set if they have negative α_i's. However, having a negative α_i in the middle of a shrinking process does not rule a support vector out from the final set. A data point is considered to be an interior point only if it satisfies $(d^2 - R^2) < 0$ when scored with the final support vector set. Therefore, it is necessary to recheck whether the data points taken out of the support vector set are truly interior points. In FISVDD, we build a backup set when the shrinking stage begins. When a data point is taken out of the support vector set, it is put into the backup set. Then the inverse matrix is "downdated" with Eq. 3.4 and its row sums are calculated. The shrinking continues until there are no negative entries in the $\boldsymbol{\alpha}$ vector. The backup set keeps growing as the linear system shrinks. When there are no negative values in $\boldsymbol{\alpha}$, we have found a support vector set, although it might not be the final one. Then the data points in the backup set are scored with the support vector set one by one in a first in, first out order. To increase the algorithm's efficiency, the backup set is scanned only once. If $(d^2 - R^2) > 0$ for a data point, then the expanding algorithm is called again, and the data point is removed from the backup set and placed back into the support vector set. The expanding finishes when all data points in the backup set have $(d^2 - R^2) \leq 0$. Although the same check can be performed on all prior data, doing so would cost too much memory and the gains are far less significant. So the backup set is emptied when each new data point arrives.

For completeness, we add a check to the unnormalized $\boldsymbol{\alpha}$ vector to make sure that the result in each iteration is improved from the previous iteration. By Corollary 7, the result is improved if the 1-norm of the unnormalized $\boldsymbol{\alpha}$ vector increases. At the end of each iteration, this norm is compared with the norm in the previous iteration. If the norm decreases, then the result from the previous iteration is restored. None of our experiments have ever violated this condition.

To summarize, FISVDD is fast and computationally efficient because the algorithm ignores interior points and is built solely on matrix manipulations. First, FISVDD tries to obtain the optimal solution in each iteration without using the interior points, similar to the idea mentioned in Syed et al. (1999). Results from many experiments show that if a proper Gaussian bandwidth is chosen, then the number of support vectors should be far smaller than the total number of observations. FISVDD takes advantage of this fact by calculating only the similarities between the new data points and the support vectors.

Secondly, it can be seen from Algorithm 3 that FISVDD is based only on matrix manipulation. Matrix inverse updating steps are the core of FISVDD, which lets the system itself choose which data points to move between support vector sets and interior point sets. Sometimes the choice of the system might not be optimal, but the existence of backup sets allows the system to correct itself and removes a significant number of calculations.

5 Implementation Details

In this section we discuss several important details for implementing FISVDD.

Algorithm 3 Fast Incremental Support Vector Data Description (FISVDD)

Input: Initialize($\boldsymbol{\alpha}, \mathrm{SV}, \mathbf{A}^{-1}, \sigma$)
for $i \leftarrow 1$, n **do**
  $Q \leftarrow$ Eq. 2.9
  if $Q \leq 0$ **then**
    pass
  else
    $\boldsymbol{\alpha}, \mathrm{SV}, \mathbf{A}^{-1} \leftarrow \mathrm{Expand}(\mathbf{x}_{k+1}, \boldsymbol{\alpha}, \mathrm{SV}, \sigma, \mathbf{A}^{-1})$
    if $\min \boldsymbol{\alpha} < 0$ **then**
      $\mathrm{Backup} \leftarrow$ Empty set
      $\boldsymbol{\alpha}, \mathrm{SV}, \mathbf{A}^{-1}, \mathrm{Backup}$
      $\leftarrow \mathrm{Shrink}(\boldsymbol{\alpha}, \mathrm{SV}, \mathbf{A}^{-1}, \mathrm{Backup})$
      if $\mathbf{card}(\mathrm{Backup}) > 1$ **then**
        for $j \leftarrow 1$, $\mathbf{card}(\mathrm{Backup})$ **do**
          $Q \leftarrow$Eq. 2.9
          if $Q > 0$ **then**
            $\boldsymbol{\alpha}, \mathrm{SV}, \mathbf{A}^{-1}$
            $\leftarrow \mathrm{Expand}(\mathrm{Backup}_j, \boldsymbol{\alpha}, \mathrm{SV}, \sigma, \mathbf{A}^{-1})$
          end if
        end for
      end if
    end if
    $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha}/\|\boldsymbol{\alpha}\|_1$
  end if
end for

Initialization

A key advantage of FISVDD is that the similarity matrix $\mathbf{A}$ is directly calculated only at initialization. As stated in Section 4, each iteration calculates only the similarities between a new data point and the existing support vectors. These are used to update the inverse of the similarity matrix; the similarity matrix is calculated only at initialization. Once the burn-in data points are selected, their similarity matrix $\mathbf{A}$ and its inverse $\mathbf{A}^{-1}$ are calculated. After the row sums of $\mathbf{A}^{-1}$ are calculated, the shrinking step in Algorithm 2 is used to pick out the interior points. Then the vector that contains the normalized row sums of $\mathbf{A}^{-1}$ is the initial $\boldsymbol{\alpha}$.

Memory

For any online method, it is important to make sure that both of the following conditions hold:

- The complexity in each step is small.
- Memory usage will never expand out of control even for very large data.

For FISVDD, the two challenges are handled smoothly. The first part is easy to see: The key parts in the algorithm (expanding and shrinking the linear systems) require only $O(k^2)$ multiplications each time, where k is the number of support vectors. In addition, k should be far less than the total number of the whole data set if a proper Gaussian kernel bandwidth σ is chosen.

For the second part, the number of support vectors can indeed grow large with streaming data. To avoid the potential threat of memory expanding out of control, we set a parameter, M, for the maximal number of support vectors, where M depends on availability of memory. When M is reached, the number of support vectors will not grow large. If a new data point $\mathbf{x}_{k+1}$ satisfies $d^2 > R^2$, then one of the three situations will occur:

- $\alpha_{k+1} > 0$ but at least one of the α_i's is less than or equal to 0. In this case, the algorithm runs normally to select the interior points.
- All α_i's are greater than 0, but α_{k+1} is the smallest among all α_i's. In this case, α_{k+1} is discarded.
- All α_i's are greater than 0, and α_{k+1} is not the smallest among all α_i's. In this case, the support vector that has the smallest α_i is replaced by $\mathbf{x}_{k+1}$, and the new α_i's are updated.

By handling these three cases, the number of support vectors will not exceed M, and the memory usage in each step is controlled.

Outliers and Close Points

Until now, our analysis focused primarily on describing the boundary of the streaming data. Another important feature of SVDD is that it finds outliers in the data so that further investigations can be taken. In Laskov et al. (2006) and Scheinberg (2006), data points are classified as outliers based on α_i values. FISVDD assumes that outliers are far from normal data and hence do not influence the support vectors and the α_i's. In addition, we assume that the boundary that is determined by the support vectors is robust to outliers. Note that if a data point is far from the support vectors, the $\mathbf{v}$ vector in Eq. 3.1 should be close to a zero vector, which indicates that the largest value in $\mathbf{v}$ should be close to 0. In FISVDD, a data point $\mathbf{z}$ is classified as an outlier if it satisfies the following condition for a preselected parameter $\epsilon_1 > 0$:

$$\max \mathbf{v} < \epsilon_1. \tag{5.1}$$

If $\mathbf{z}$ is classified as an outlier, then it is passed to further investigation, and no α value is assigned to it.

Another special case we have to consider is a new data point that is very close to one of the existing support vectors. Although in practice it is rare that a new data point is exactly the same as an existing support vector, it is possible that they are very close to each other. In this case, the similarity matrix $\mathbf{A}$ will be ill-conditioned and $\mathbf{A}^{-1}$ might be not accurate. We can avoid this situation by also looking at the maximal entry value in $\mathbf{v}$. If a new data point is very close to one of the support vectors, then the maximal entry value in $\mathbf{v}$ will be close to 1. In FISVDD, a point is discarded if it satisfies the following condition for a preselected parameter $\epsilon_2 > 0$:

$$\max \mathbf{v} > 1 - \epsilon_2. \tag{5.2}$$

Finally, note that these preprocessing steps can help prevent unnecessary calculations if the Gaussian kernel bandwidth σ is not a proper bandwidth. If σ is too small, then every data point tends to be a support vector and the similarity between every pair of data points is close to 0. If σ is too large, then the similarity between every pair of data points is close to 1. Introducing ϵ_1 and ϵ_2 can prevent these cases.

6 Experiments

We examined the performance of FISVDD with four real data sets: shuttle data (Lichman 2013), mammography data (Woods et al. 1993), forest cover (ForestType) data (Rayana 2016), and the SMTP subset of KDD Cup 99 data (Rayana 2016). The purpose of our experiments is to show that compared to the incremental SVM method (which can achieve global optimal solutions), the FISVDD method does not lose much in either objective function value or outlier detection accuracy while it demonstrates significant gains in efficiency. Our experiments used 4/5 of the normal data, randomly chosen, for training. The remaining normal data and the outliers together form the testing sets. All duplicates in the data sets are removed beforehand. Proper Gaussian bandwidths are selected by using fivefold cross validation, although selecting a proper Gaussian bandwidth is beyond the scope of this paper. SAS/IML® software is used in performing the experiments. In this paper, we compare FISVDD with the one-class incremental SVM method (Laskov et al. 2006), a well-known technique for performing global optimal SVDD. For each method, the following quantities are measured in Table 1:

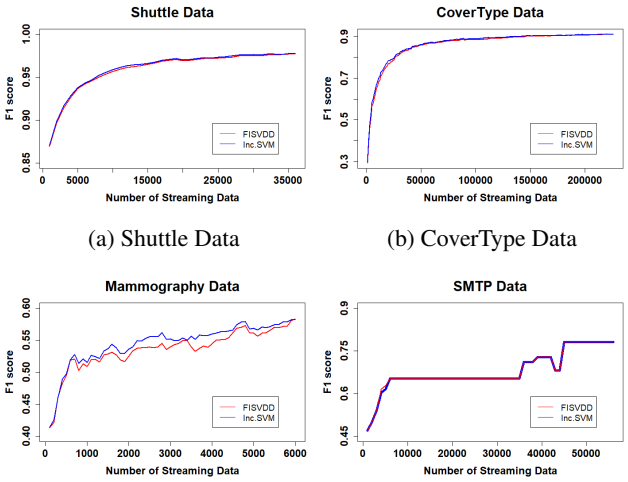
- Time: The time used to learn the SVDD model.
- Objective function value (OFV): The objective function values that were obtained with Eq. 2.6 after each iteration.
- Number of support vectors (#sv): The number of support vectors when the training phase is finished. This number is related to the efficiency of the testing phase. When more support vectors exist, more calculations are required in testing.

The time consumed by the incremental SVM method with interior points discarded after each iteration is listed in parentheses. Table 1 also lists the settings for the experiments, including Gaussian bandwidth (Sigma), number of training observations (#Train obs), number of testing observations (#Test obs), and number of variables (#Var).

Table 1: Experimental Results of FISVDD and Incremental SVM on Different Data Sets

Data	Sigma	Method	#Train obs	#Test obs	#Var	OFV	Time (s)	#sv
Shuttle	5.5	FISVDD	36469	21531	9	1.7378e−3	251.01	1736
		Inc. SVM				1.7369e−3	22923.57 (312.65)	1926
CoverType	470	FISVDD	226641	59407	10	1.14158e−2	19.47	432
		Inc. SVM				1.14155e−2	12954.81 (29.45)	470
Mammography	0.8	FISVDD	6076	1773	6	9.8134e−3	1.19	317
		Inc. SVM				9.8008e−3	67.01 (1.58)	317
SMTP	6	FISVDD	56967	14263	3	0.393	0.27	5
		Inc. SVM				0.393	2.49 (0.38)	5

Table 1 shows that for the same Gaussian bandwidth, the FISVDD method is much faster than the incremental SVM method, with only a tiny sacrifice in the objective function value. Because incremental SVM achieves global optimal solutions, the solutions provided by FISVDD are very close to the global optimal solutions. Even with interior points discarded after each iteration, FISVDD is faster than incremental SVM for the data sets in our experiments. As explained in Section 4, FISVDD is faster because it is based solely on matrix manipulation and thus many calculations are saved.



(a) Shuttle Data (b) CoverType Data

(c) Mammography Data (d) SMTP Data

Figure 1: F-1 Measure for Different Data Sets

Figure 1 shows plots of the F-1 measure (Tan, Steinbach, and Kumar 2007) of the accuracy of FISVDD and incremental SVM with different training sizes. The plots show that by discarding interior points at the end of each iteration, there is almost no loss in the quality of outlier detection.

7 Conclusion

This paper introduces a fast incremental SVDD learning algorithm (FISVDD), which is more efficient than existing SVDD algorithms. In each iteration, FISVDD considers only the incoming data point and the support vectors that were determined in the previous iteration. The essential calculations of FISVDD are contributed from incremental and decremental updates of a similar matrix inverse $\mathbf{A}^{-1}$. This algorithm builds on an observation that is natural in SVDD models but has not been fully utilized by existing SVDD algorithms: that all support vectors on the boundary have the same distance to the center of sphere in a higher-dimensional feature space as mapped by the Gaussian kernel function. FISVDD uses the signs of entries in the row sums of $\mathbf{A}^{-1}$ to determine the interior points and support vectors and uses their magnitudes to determine the Lagrange multiplier α_i for each support vector. Experimental results indicate that FISVDD gains much efficiency with almost no loss in accuracy and objective function value.

Acknowledgement

We would like to thank Anne Baxter, Maria Jahja, and Cong Meng for their help in this paper. We would also like to thank Minghui Liu, Joshua Griffin, Yuwei Liao, and Seunghyun Kong for discussions that are related to SVDD.

References

Ben-Hur, A.; Horn, D.; Siegelmann, H. T.; and Vapnik, V. 2001. Support vector clustering. *Journal of Machine Learning Research* 2(Dec):125–137.

Cauwenberghs, G., and Poggio, T. 2001. Incremental and decremental support vector machine learning. In *Advances in Neural Information Processing Systems*, 409–415.

Evangelista, P.; Embrechts, M.; and Szymanski, B. 2007. Some properties of the Gaussian kernel for one class learning. *Artificial Neural Networks–ICANN 2007* 269–278.

Gallier, J. 2010. The Schur complement and symmetric positive semidefinite (and definite) matrices. *Penn Engineering*.

Gu, B.; Sheng, V. S.; Tay, K. Y.; Romano, W.; and Li, S. 2015. Incremental support vector learning for ordinal regression. *IEEE Transactions on Neural Networks and Learning Systems* 26(7):1403–1416.

Kakde, D.; Chaudhuri, A.; Kong, S.; Jahja, M.; Jiang, H.; and Silva, J. 2017. Peak criterion for choosing Gaussian kernel bandwidth in support vector data description. In *Prognostics and Health Management (ICPHM), 2017 IEEE International Conference on*, 33–41. IEEE.

Laskov, P.; Gehl, C.; Krüger, S.; and Müller, K.-R. 2006. Incremental support vector learning: Analysis, implementation and applications. *Journal of Machine Learning Research* 7(Sep):1909–1936.

Lichman, M. 2013. UCI machine learning repository.

Meyer, C. D. 2000. *Matrix analysis and applied linear algebra*, volume 2. Siam.

Rayana, S. 2016. ODDS library.

Scheinberg, K. 2006. An efficient implementation of an active set method for SVMs. *Journal of Machine Learning Research* 7(Oct):2237–2257.

Schölkopf, B.; Williamson, R. C.; Smola, A. J.; Shawe-Taylor, J.; and Platt, J. C. 2000. Support vector method for novelty detection. In *Advances in Neural Information Processing Systems*, 582–588.

Smola, A. J., and Schölkopf, B. 1998. *Learning with kernels*. GMD-Forschungszentrum Informationstechnik.

Syed, N. A.; Huan, S.; Kah, L.; and Sung, K. 1999. Incremental learning with support vector machines.

Tan, P.-N.; Steinbach, M.; and Kumar, V. 2007. *Introduction to data mining*. Pearson Education India.

Tax, D. M. J., and Duin, R. P. W. 2004. Support vector data description. *Machine learning* 54(1):45–66.

Tax, D. M. J., and Laskov, P. 2003. Online SVM learning: from classification to data description and back. In *Neural Networks for Signal Processing, 2003. NNSP'03. 2003 IEEE 13th Workshop on*, 499–508. IEEE.

Woods, K. S.; Doss, C. C.; Bowyer, K. W.; Solka, J. L.; Priebe, C. E.; and Kegelmeyer Jr, W. P. 1993. Comparative evaluation of pattern recognition techniques for detection of microcalcifications in mammography. *International Journal of Pattern Recognition and Artificial Intelligence* 7(06):1417–1436.

Xiao, Y.; Wang, H.; Zhang, L.; and Xu, W. 2014. Two methods of selecting Gaussian kernel parameters for one-class SVM and their application to fault detection. *Knowledge-Based Systems* 59:75–84.