Sampling Method for Fast Training of Support Vector Data Description

Arin Chaudhuri ARIN.CHAUDHURI@SAS.COM
Deovrat Kakde DEV.KAKDE@SAS.COM
Maria Jahja MARIA.JAHJA@SAS.COM
Wei Xiao WEI.XIAO@SAS.COM
Seunghyun Kong SEUNGHYUN.KONG@SAS.COM
Hansi Jiang HANSI.JIANG@SAS.COM
Sergiy Peredriy SERGIY.PEREDRIY@SAS.COM

Abstract

Support Vector Data Description (SVDD) is a machine learning technique used for single class classification and outlier detection. The SVDD model for normal data description builds a minimum radius hypersphere around the training data. A flexible description can be obtained by use of Kernel functions. The data description is defined by the support vectors obtained by solving quadratic optimization problem which minimizes the volume enclosed by the hypersphere. The time required to solve the quadratic programming problem is directly related to the number of observations in the training data set. This leads to very high computing time for large training datasets. In this paper we propose a new iterative sampling-based method for SVDD training. The method incrementally learns the training data set description at each iteration by computing SVDD on an independent random sample selected with replacement from the training data set. The experimental results indicate that the proposed method is extremely fast and provides near-identical data description as compared to training using the entire data set in one iteration. Proposed method can be easily implemented as a wrapper code around the core module for SVDD training computations either in a single machine or a multi-machine distributed environment.

1. Introduction

Support Vector Data Description (SVDD) is a machine learning technique used for single class classification and outlier detection. SVDD technique is similar to Support Vector Machines and was first introduced by Tax and Duin (Tax & Duin, 2004). It can be used to build a flexible boundary around single class data. Data boundary is characterized by observations designated as support vectors. SVDD is used in domains where majority of data belongs to a single class. Several researchers have proposed use of SVDD for multivariate process control (Sukchotrat et al., 2009). Other applications of SVDD involve machine condition monitoring (Widodo & Yang, 2007; Ypma et al., 1999) and image classification (Sanchez-Hernandez et al., 2007).

1.1. Mathematical Formulation of SVDD

Normal Data Description:
The SVDD model for normal data description builds a minimum radius hypersphere around the data.

Primal Form:
Objective Function:

$$\min R^2 + C\sum_{i=1}^{n}\xi_i, \tag{1}$$

subject to:

$$\|x_i - a\|^2 \leq R^2 + \xi_i, \forall i = 1, \ldots, n, \tag{2}$$
$$\xi_i \geq 0, \forall i = 1, ...n. \tag{3}$$

where:
$x_i \in \mathbb{R}^m, i = 1, \ldots, n$ represents the training data,
R : radius, represents the decision variable,
ξ_i : is the slack for each variable,
a: is the center, a decision variable,
$C = \frac{1}{nf}$: is the penalty constant that controls the trade-off between the volume and the errors, and,
f : is the expected outlier fraction.

Dual Form:
The dual formulation is obtained using the Lagrange multipliers.

Objective Function:

$$\max \sum_{i=1}^{n} \alpha_i(x_i.x_i) - \sum_{i,j} \alpha_i\alpha_j(x_i.x_j), \tag{4}$$

subject to:

$$\sum_{i=1}^{n} \alpha_i = 1, \tag{5}$$

$$0 \leq \alpha_i \leq C, \forall i = 1, \ldots, n. \tag{6}$$

where:
$\alpha_i \in \mathbb{R}$: are the Lagrange constants,
$C = \frac{1}{nf}$: is the penalty constant.

Duality Information:
Depending upon the position of the observation, the following results hold good:

Center Position:

$$\sum_{i=1}^{n} \alpha_i x_i = a. \tag{7}$$

Inside Position:

$$\|x_i - a\| < R \rightarrow \alpha_i = 0. \tag{8}$$

Boundary Position:

$$\|x_i - a\| = R \rightarrow 0 < \alpha_i < C. \tag{9}$$

Outside Position:

$$\|x_i - a\| > R \rightarrow \alpha_i = C. \tag{10}$$

The radius of the hypersphere is calculated as follows:

$$R^2 = (x_k.x_k) - 2\sum_{i} \alpha_i(x_i.x_k) + \sum_{i,j} \alpha_i\alpha_j(x_i.x_j). \tag{11}$$

using any $x_k \in SV_{<C}$, where $SV_{<C}$ is the set of support vectors that have $\alpha_k < C$.

Scoring:

For each observation z in the scoring data set, the distance dist$^2(z)$ is calculated as follows:

$$dist^2(z) = (z.z) - 2\sum_{i} \alpha_i(x_i.z) + \sum_{i,j} \alpha_i\alpha_j(x_i.x_j). \tag{12}$$

The scoring dataset points with $dist^2(z) > R^2$ are designated as outliers.

The circular data boundary can include a significant amount of space with a very sparse distribution of training observations. Scoring with this model can increase the probability of false positives. Hence, instead of a circular shape, a compact bounded outline around the data is often desired. Such an outline should approximate the shape of the single-class training data. This is possible with the use of kernel functions.

Flexible Data Description:

The Support Vector Data Description is made flexible by replacing the inner product $(x_i.x_j)$ with a suitable kernel function $K(x_i, x_j)$. The Gaussian kernel function used in this paper is defined as:

$$K(x_i, x_j) = \exp \frac{-\|x_i - x_j\|^2}{2s^2} \tag{13}$$

where s: Gaussian bandwidth parameter.

The modified mathematical formulation of SVDD with kernel function is as follows:

Objective function:

$$\max \sum_{i=1}^{n} \alpha_i K(x_i, x_i) - \sum_{i,j} \alpha_i\alpha_j K(x_i, x_j), \tag{14}$$

Subject to:

$$\sum_{i=1}^{n} \alpha_i = 1, \tag{15}$$

$$0 \leq \alpha_i \leq C, \forall i = 1, \ldots, n. \tag{16}$$

The results 7 through 10 hold good when the kernel function is used in the mathematical formulation.
The threshold R^2 is calculated as :

$$R^2 = K(x_k, x_k) - 2\sum_{i} \alpha_i K(x_i, x_k) + \sum_{i,j} \alpha_i\alpha_j K(x_i, x_j) \tag{17}$$

using any $x_k \in SV_{<C}$, where $SV_{<C}$ is the set of support vectors that have $\alpha_k < C$.

Scoring:

For each observation z in the scoring dataset, the distance $dist^2(z)$ is calculated as follows:

$$dist^2(z) = K(z, z) - 2\sum_{i} \alpha_i K(x_i, z) + \sum_{i,j} \alpha_i\alpha_j K(x_i, x_j). \tag{18}$$

The scoring dataset points with $dist^2(z) > R^2$ are designated as outliers.

1.2. Mathematical Formulation of One-class Support Vector Machines (OCSVM)

The One-class Support Vector Machines (OCSVM) is a one-class classification technique similar to the SVDD. In-

stead of obtaining a bounding hypershpere around the training data set, the OCSVM algorithm finds the maximal margin hyperplane which best separates the training data from the origin (Scholkopf et al., 1999). Similar to SVDD, the OCSVM uses kernel functions to map training data set into a higher dimensional feature space. The dual formulation of OCSVM can be expressed as follows:

$$\max \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j K(x_i,x_j), \tag{19}$$

Subject to:

$$\sum_{i=1}^{l}\alpha_i = 1, \tag{20}$$

$$0 \le \alpha_i \le \frac{1}{\nu l}, \forall i = 1,\ldots,l. \tag{21}$$

where:
$\alpha_i \in \mathbb{R}$: are the Lagrange constants,
ν: is a parameter that controls the trade-off between maximizing the distance of the hyperplane from the origin and the number of data points contained by the hyperplane,
l: is the number of points in the training data set, and,
$K(x_i, x_j)$: is the kernel function.
The Gaussian kernel function, which is used in this paper is defined as:

$$K(x_i,x_j) = \exp\frac{-\|x_i - x_j\|^2}{2s^2} \tag{22}$$

where
s: Gaussian bandwidth parameter.

With Gaussian kernel function, the first term of the SVDD objective function defined in equation (14) evaluates to 1. Hence the OCSVM formulation defined in equation 19 to equation 21 is similar to the SVDD formulation defined in equation 14 to equation 16 when Gaussian kernel function is used. Hence, it should be noted that, although, the discussion in the remainder of this paper is presented in the context of SVDD, the sampling-based method can be used equally well when solving a OCSVM problem.

2. Need for a Sampling-based Approach

As outlined in Section 1.1, SVDD of training data set is obtained by solving a quadratic programming problem. The time required to solve the quadratic programming problem is directly related to the number of observations in the training data set. The actual time complexity depends upon the implementation of the underlying Quadratic Programming solver. We used LIBSVM (Chang & Lin, 2011) to evaluate SVDD training time as a function of the training data set size. Figure 1 shows processing time as a function of training data set size for the two Donut data set. Refer to Figure 3(c) for a scatterplot of two Donut data. In Figure 1 the x-axis indicates the training data set size and the y-axis indicates processing time in minutes. As indicated in Figure 1, the SVDD training time is low for small or moderately sized training data sets of size up to 15,000 observations, but for larger data sets, SVDD training time is extremely high.

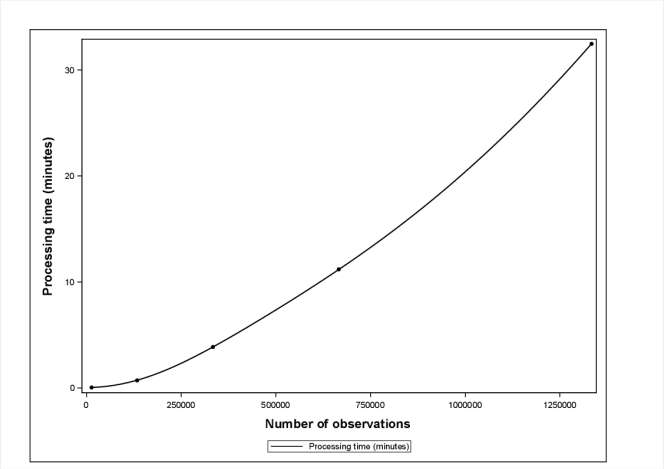


Figure 1. SVDD Training Time: Two Donut data

There are applications of SVDD in areas such as process control and equipment health monitoring where size of training data set can be very large, consisting of few million observations. The training data set consists of sensors readings measuring multiple key health or process parameters at a very high frequency. For example, a typical airplane currently has ˜6,000 sensors measuring critical health parameters and creates 2.5 terabytes of data per day. By 2020, this number is expected to triple or quadruple to over 7.5 terabytes (Ege, 2015). In such applications, multiple SVDD training models are developed, each representing separate operating mode of the equipment or process settings. The success of SVDD in these applications require algorithms which can train using huge amounts of training data in an efficient manner.
To improve performance of SVDD training on large data sets, we propose a new sampling based method. Instead of using all observations from the training data set, the algorithm computes the training data SVDD by iteratively computing SVDD on independent random samples obtained from the training data set. The method works well with extremely small sample sizes. We provide a criteria for convergence. At convergence, the proposed method provides a reasonable approximation of the data description that can be obtained by using all training data set observations in a single iteration.
The rest of the paper is organized as follows. Section 3 provides details of the proposed sampling-based iterative

method. Results of training with the proposed method are provided in section 4. The analysis of high dimensional data is provided in Section 5. Results of testing sampling method on random polygons is provided in section Section 6. Finally, conclusions are provided in section 7.

3. Sampling-based Method

The Decomposition and Combination method of Luo et.al.(Luo et al., 2010) and K-means Clustering Method of Kim et.al.(Kim et al., 2007), both use sampling for fast SVDD training, but are computationally expensive. The first method by Lou et.al. uses an iterative approach and requires one scoring action on the entire training data set per iteration. The second method by Kim et.al. is a classic divide and conquer algorithm. It uses each observation from the training data set to arrive at the final solution. Additional details about these methods are provided in Appendix A.

In this section we describe our sampling-based method for fast SVDD training. The method iteratively samples from the training data set with the objective of updating a set of support vectors called as the master set of support vectors (SV^*). During each iteration, the method updates SV^* and corresponding threshold R^2 value and center a. As the threshold value R^2 increases, the volume enclosed by the SV^* increases. The method stops iterating and provides a solution when the threshold value R^2, center a and hence the volume enclosed by SV^* converges. At convergence, the members of the master set of support vectors SV^*, characterize the description of the training data set. For all test cases, our method provided a good approximation to the solution that can be obtained by using all observations in the training data set.

Our method addresses drawbacks of existing sampling based methods proposed by Luo et.al.(Luo et al., 2010) and Kim et.al.(Kim et al., 2007). In each iteration, our method learns using very a small sample from the training data set and overall uses a very small subset of the training data set. The method does not require any scoring actions while it trains.

The sampling method works well across different sample sizes. The user is not required to make any decision regarding the right sample size. It provides a better alternative to SVDD training on one large sample from the training data set, where establishing a right size, especially with high dimensional data can be a challenge.

The important steps in this algorithm are outlined below:

Step 1: The algorithm is initialized by selecting a random sample S_0 of size n from the training data set of M observations ($n \ll M$). SVDD of S_0 is computed to obtain the corresponding set of support vectors SV_0. The set SV_0 initializes the master set of support vectors SV^*. The iteration number i is set to 1.

Step 2: During this step, the algorithm updates the master set of support vectors, SV^* until the convergence criteria is satisfied. In each iteration i, following steps are executed:

Step 2.1: A random sample S_i of size n is selected and its SVDD is computed. The corresponding support vectors are designated as SV_i.

Step 2.2: A union of SV_i with the current master set of support vectors, SV^* is taken to obtain a set $S_i^{'}$ ($S_i^{'} = SV_i \bigcup SV^*$).

Step 2.3: SVDD of $S_i^{'}$ is computed to obtain corresponding support vectors $SV_i^{'}$, threshold value $R_i^{2'}$ and center $a_i^{'}$. The set $SV_i^{'}$, is designated as the new master set of support vectors SV^*.

Convergence Criteria: At the end of each iteration i, following conditions are checked to determine if the convergence criteria is satisfied. The algorithm stops iterating and provides solution if convergence criteria is satisfied for t consecutive iterations.

a) $i = maxiter$, where $maxiter$ is the maximum number of iteration; or
b) $\Delta a_i \leq \epsilon \Delta a_{i-1}$, and;
$\left\| R_i^{2'} - R_{i-1}^{2'} \right\| \leq \epsilon R_{i-1}^{2'}$
where $\Delta a_i = \left\| a_i^{'} - a_{i-1}^{'} \right\|$ and ϵ is a constant

The pseudo-code for this method is provided in algorithm 1. The pseudo-code uses following conventions:
1) $S_i \leftarrow SAMPLE(T, n)$ indicate data set S_i obtained by selecting random sample of size n from data set T.
2) δS_i indicate SVDD computation on data set S_i.
3) $< SV_i, R_i^2, a_i > \leftarrow \delta S_i$ indicate the set of support vectors SV_i, threshold value R_i^2 and center a_i obtained by performing SVDD computations on data set S_i.

As outlined in steps 1 and 2, the algorithm obtains the final training data description by incrementally updating the master set of support vectors SV^* and there by in each iteration expanding the volume enclosed by the support vectors. During each iteration, the algorithm first selects a small random sample S_i, computes its SVDD and obtains corresponding set of support vectors, SV_i. The support vectors of set SV_i are included in the master set of support vectors SV^* to obtain $S_i^{'}$ ($S_i^{'} = SV_i \bigcup SV^*$). The set $S_i^{'}$ thus represents an incremental expansion of the current master set of support vectors SV^*. Since SV_i is set of support vectors of a very small random sample S_i, some members of SV_i can be potentially 'inside' the data boundary characterized by SV^*. The following SVDD computation on $S_i^{'}$ elimintes such 'inside' points. The corresponding set of support vectors, $SV_i^{'}$ is designated as

Algorithm 1 Sampling-based iterative method

Require: T (training data set) , n (sample size), convergence criteria, s (Gaussian band width parameter), f (fraction outliers) and t (required number of consecutive observations satisfying convergence criteria).

1: $S_0 \leftarrow SAMPLE(T, n)$
2: $< SV_0, R_0^2, a_0 > \leftarrow \delta S_0$
3: $SV^* \leftarrow \mathrm{SV}_0$
4: $i = 1$
5: **while** (Convergence criteria not satisfied for t consecutive obs) **do**
6: $S_i \leftarrow SAMPLE(T, n)$
7: $< SV_i, R_i^2, a_i > \leftarrow \delta S_i$
8: $S_i^{'} \leftarrow SV_i \bigcup SV^*$.
9: $< SV_i^{'}, R_i^{2'}, a_i^{'} > \leftarrow \delta S_i^{'}$
10: Test for convergence
11: $SV^* \leftarrow SV_i^{'}$
12: $i = i + 1$
13: **end while**
14: **return** SV^*

the new master set of support vectors SV^*. During initial iterations as SV^* gets updated, its threshold value $R_i^{2'}$ increases and hence the volume enclosed by the master set of support vectors continues to expand. At convergence, the data description obtained using SV^* approximates data description that can be obtained using all training data observations in a single iteration.

Each iteration of the our algorithm involves two SVDD computations and one union operation. The first SVDD computation is fast since it is perfomed on a small sample of training data set. For the remaining two operations, our method exploits the fact that for most data sets, support vectors computed using SVDD are typically a very small fraction of input data set. Hence the union operation and the scond SVDD computation both of which use sets of support vectors as input, are fast as well. Overall, since all three operations in our method are fast, time required to obtain final solution is considerably less as compared to using all observations.

3.0.1. Modified Sampling Algorithm

A modification of sampling algorithm is provided in this section. The modified algorithm is similar to the one outlined in section 3 except for step 2.1. In modified sampling algorithm, during each iteration, multiple SVDD computations on independent random samples are pefrormed before arriving at the set SV_i, at the end of step 2.1. The important steps in the algorithm are outlined below:

Step 1: The algorithm is initialized by selecting a random sample S_0 of size n from the training data set of M observations ($n \ll M$). SVDD of S_0 is computed to obtain the corresponding set of support vectors SV_0. The set SV_0 is designated as the master set of support vectors SV^*. The iteration number i is set to 1.

Step 2: During this step, the algorithm updates the master set of support vectors, SV^* until the convergence criteria is satisfied. In each iteration i, following steps are executed:

Step 2.1: Obtain q independent samples $S_{i1}, S_{i2}...S_{iq}$, compute their individual SVDD and obtain corresponding set of support vectors $SV_{i1}, SV_{i2}...SV_{iq}$. Take union of q sets of support vectors to obtain SV_i where $SV_i = \bigcup_{j=1}^{q} SV_{ij}$. **Step 2.2**: A union of SV_i with the current master set of support vectors, SV^* is taken to obtain a set $S_i^{'}$ ($S_i^{'} = SV_i \bigcup SV^*$).

Step 2.3: SVDD of $S_i^{'}$ is computed to obtain corresponding support vectors $SV_i^{'}$, threshold value $R_i^{2'}$ and center $a_i^{'}$. The set $SV_i^{'}$, is designated as the new master set of support vectors SV^*.

Convergence Criteria: At the end of each iteration i, following conditions are checked to determine if the convergence criteria is satisfied. The algorithm stops iterating and provides solution if convergence criteria is satisfied for t consecutive iterations.

a) $i = maxiter$, where $maxiter$ is the maximum number of iteration; or
b) $\Delta a_i \leq \epsilon \Delta a_{i-1}$, and;
$\left\| R_i^{2'} - R_{i-1}^{2'} \right\| \leq \epsilon R_{i-1}^{2'}$
where $\Delta a_i = \left\| a_i^{'} - a_{i-1}^{'} \right\|$ and ϵ is a constant

The pseudo-code for this method is provided in algorithm 2. The pseudo-code uses following conventions:
1) $S_i \leftarrow SAMPLE(T, n)$ indicate data set S_i obtained by selecting random sample of size n from data set T.
2) δS_i indicate SVDD computation on data set S_i.
3) $< SV_i, R_i^2, a_i > \leftarrow \delta S_i$ indicate the set of support vectors SV_i, threshold value R_i^2 and center a_i obtained by performing SVDD computations on data set S_i.

3.0.2. Distributed Implementation

For extremely large training datasets, efficiency gains using distributed implementation are possible. Figure 2 describes SVDD solution using the sampling method outlined in section 3 utilizing a distributed architecture. The training data set with M observations is first distributed over p worker nodes. Each worker node computes SVDD of its $\frac{M}{p}$ observations using the sampling method to obtain its own master set of support vectors SV_i^*. Once SVDD computations are completed, each worker node promotes its own master set of support vectors SV_i^*, to the controller node. The controller node takes a union of all worker node master sets of

Algorithm 2 Modified sampling-based iterative method

Require: T (training data set) , n (sample size), convergence criteria, q (number of sample SVDD computations per iteration), s (Gaussian band width parameter), f (fraction outliers) and t (required number of consecutive observations satisfying convergence criteria)

$S_0 \leftarrow SAMPLE(T, n)$
$< SV_0, R_0^2, a_0 > \leftarrow \delta S_0$
$SV^* \leftarrow SV_0$
$i = 1$
while (Convergence criteria not satisfied for t consecutive obs) **do**
  while $j \leq q$ **do**
    $S_{ij} \leftarrow SAMPLE(T, n)$
    $< SV_{ij}, R_{ij}^2, a_{ij} > \leftarrow \delta S_{ij}$
    if $j = 1$ **then**
      $SV_i \leftarrow SV_{ij}$
    else
      $SV_i \leftarrow SV_i \bigcup SV_{ij}$
    end if
    $j = j + 1$
  end while
  $S_i^{'} \leftarrow SV_i \bigcup SV^*$.
  $< SV_i^{'}, R_i^{2'}, a_i^{'} > \leftarrow \delta S_i^{'}$
  Test for convergence
  $SV^* \leftarrow SV_i^{'}$
  $i = i + 1$
end while
return SV^*

support vectors, SV_i^* to create data set $S^{'}$. Finally, solution is obtained by performing SVDD computation on $S^{'}$. The corresponding set of support vectors SV^* are used to approximate the original training data set description.

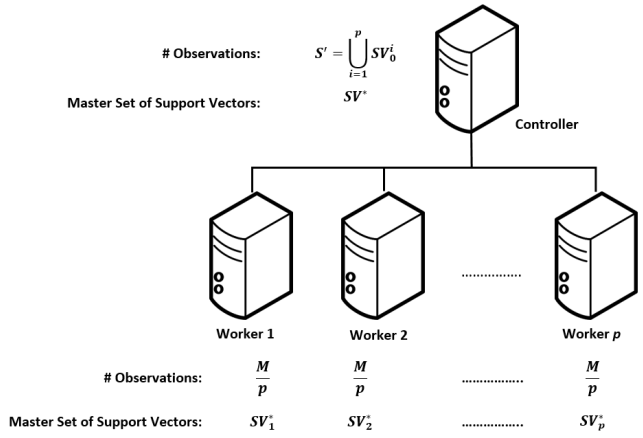


Figure 2. Distributed Implementation

4. Results

To test our sampling method, we experimented with three data sets of known geometry: Banana-shaped, Star-shaped, and a two Donut data. Figure 3(a)-3(c) illustrate these three data sets. For each data set, we first obtained SVDD using all observations. Table 1 summarizes the results.

For each data set, we varied the value of the sample size n from 3 to 20 and obtained multiple SVDD using the sampling method. For each sample size value, the total processing time and number of iterations till convergence were noted. Figure 4 and 5 illustrate the results. The vertical reference line indicates the sample size corresponding to the minimum processing time. Table 2 provides the minimum processing time, corresponding sample size and other details for all three data sets. Figure 7 shows the convergence of threshold R^2 for the Banana-shaped data trained using sampling method.

Results provided in Table 1 and Table 2 indicate that our method provides magnitude performance improvement as compared to training using all observations in a single iteration. The threshold R^2 values obtained using the sampling-based method are approximately equal to the values that can be obtained by training using all observations in a single iteration. Although the radius values are same, to confirm if the data boundary defined using support vectors is similar, we performed scoring on a 200×200 data grid. Figure 8 provides the scoring results for all data sets. The scoring results are similar.

Data	#Obs	R^2	#SV	Time
Banana	11,016	0.8789	21	1.98 sec
Two Donut	1,333,334	0.8982	178	32 min
Star	64,000	0.9362	76	11.55 sec

Table 1. SVDD Training using full SVDD method

Data	Sample Size	No. of Iterations	R^2	#SV	Time
Banana	6	119	0.8717	19	0.315
Two Donut	11	157	0.8967	37	0.285
Star	11	141	0.9317	44	0.284

Table 2. SVDD Results using Sampling Method

Note: *In the remainder of this paper, we refer training method using all observations in one iteration as full SVDD method.*

5. Analysis of High Dimensional Data

Section 4 provided comparison of our sampling method with the alternate method which uses all observations in the training data set. For two-dimensional data sets the performance of sampling method can be visually judged using the scoring results. We tested the sampling method with high dimensional datasets, where such visual feedback about classification accuracy of sampling method is not available. We compared classification accuracy of the sampling method with the accuracy of training with all observations. We use the F_1-measure to quantify the classification accuracy. (Zhuang & Dai, 2006). The F_1-measure is defined as follows:

$$F_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{23}$$

where:

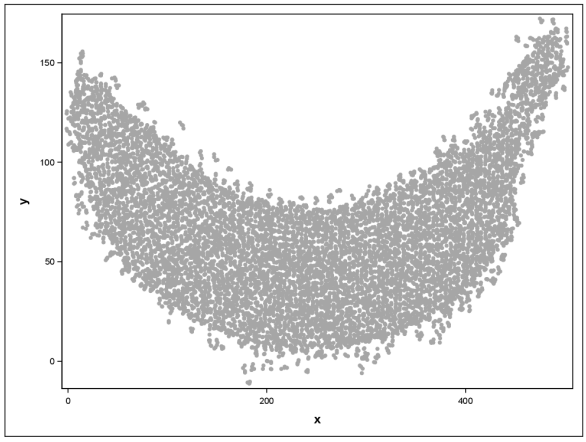
$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \tag{24}$$

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}. \tag{25}$$

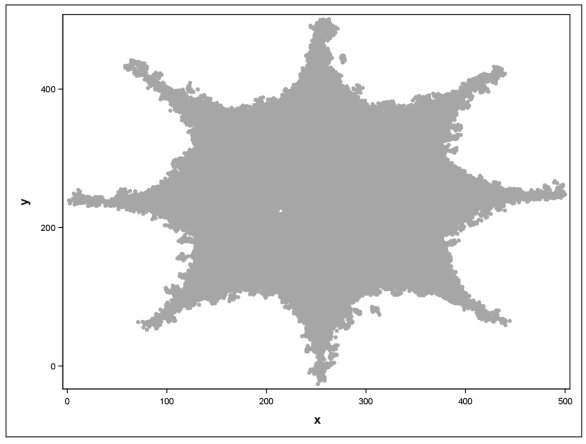
Thus high precision relates to a low false positive rate, and high recall relates to a low false negative rate. We chose the F_1-measure because it is a composite measure that takes into account both the Precision and the Recall. Models with higher values of F_1-measure provide a better fit.

5.1. Analysis of Shuttle Data

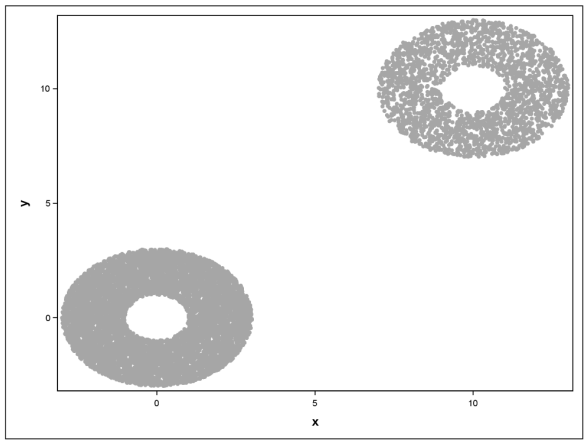
In this section we provide results of our experiments with Statlog (shuttle) dataset (Lichman, 2013). This is a high dimensional data consists of nine numeric attributes and one class attribute. Out of 58,000 total observations, 80% of the observations belong to class one. We created a training data set of randomly selected 2,000 observations belonging to class one. The remaining 56,000 observations were used to create a scoring data set. SVDD model was first trained using all observations in the training data set. The training results were used to score the observations in the scoring data set to determine if the model could accurately classify an observation as belonging to class one and the accuracy of scoring was measured using the F_1-measure. We then trained using the sampling-based method, followed by scoring to compute the F_1-measure again. The sample size for the sampling-based method was set to 10 (number of variables + 1). We measured the performance of the sampling method using the F_1-measure ratio defined as $F_{\text{Sampling}}/F_{\text{Allobs}}$ where F_{Sampling} is the F_1-measure obtained when the value obtained using the sampling method for training, and F_{Allobs} is the value of F_1-measure computed when all observations were used for training. A value close to 1 indicate that sampling method is competitive with full SVDD method. We repeated the above steps varying the training data set of size from 3,000 to 40,000 in the increments of 1,000. The corresponding scoring data set size changed from 55,000 to 18,000. Figure 9 provides the plot of F_1-measure ratio. The plot of $F1$-measure ratio is constant, very close to 1 for all training data set sizes, provides the evidence that our sampling method provides near identical classification accuracy as compared to full SVDD method. Figure 10 provides the plot of the processing time for the sampling method and training using all obsrvations. As the training data set size increased, the processing time for full SVDD method increased almost linearly to a value of about 5 seconds for training data set of 40,000 observations. In comparison, the processing time of the sampling based method was in the range of 0.24 to 0.35 sec. The results prove that the sampling-based method is efficient and it provides near identical results to full SVDD method.



(a) Banana-shaped data

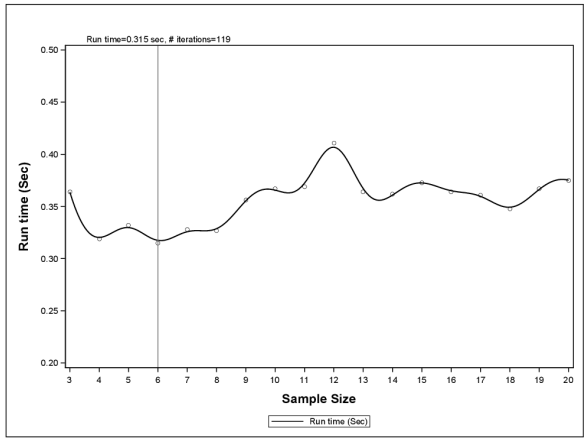


(b) Star-shaped data

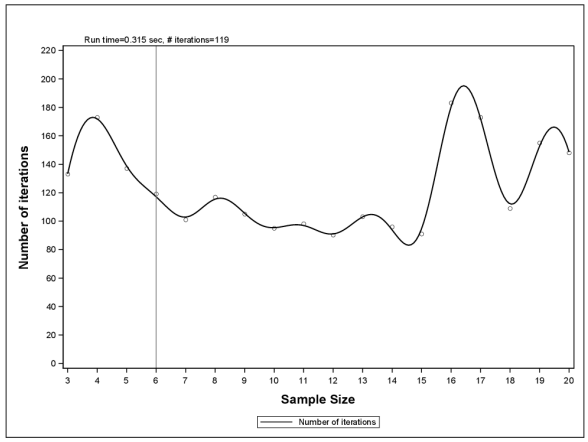


(c) Two donut data

Figure 3. Scatter plots

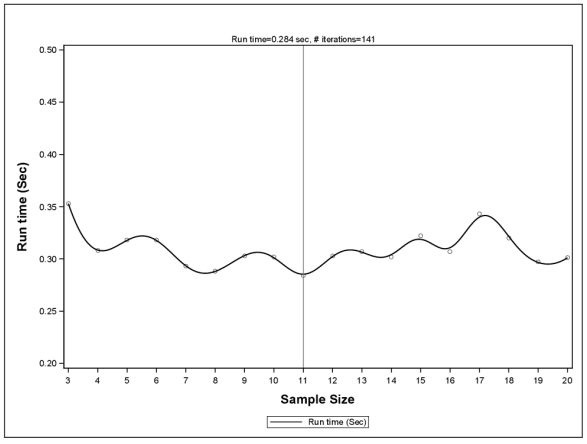


(a) Run time vs. sample size

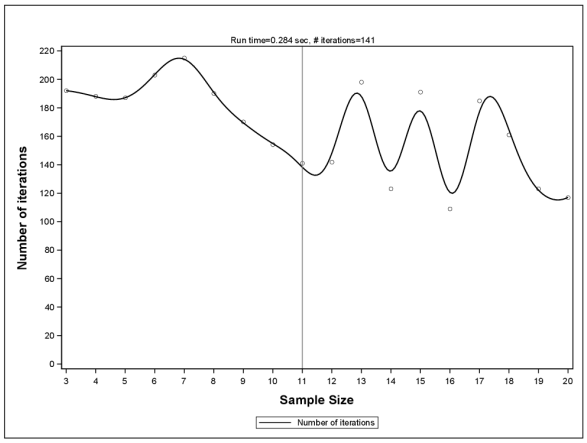


(b) # iterations vs. sample size

Figure 4. Banana-shaped data

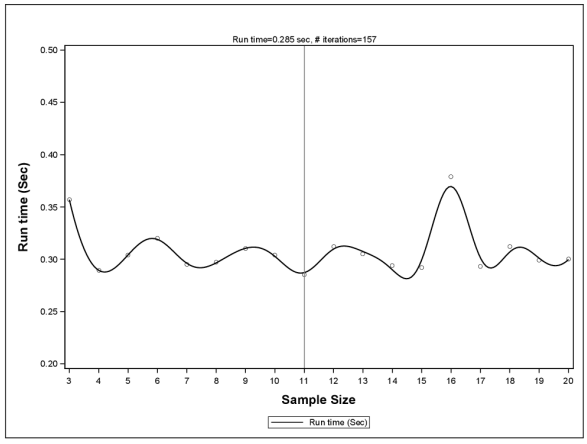


(a) Run time vs. sample size

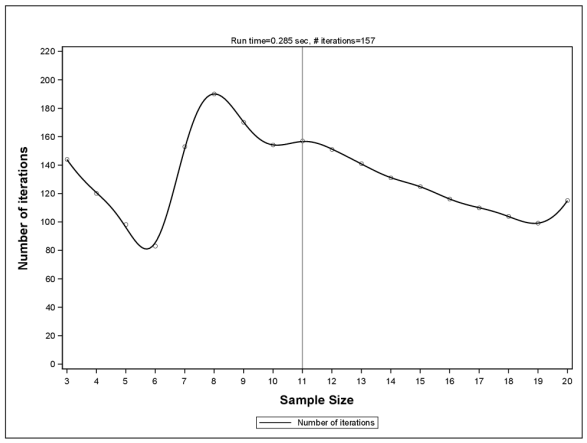


(b) # iterations vs. sample size

Figure 5. Star-shaped data



(a) Run time vs. sample size



(b) # iterations vs. sample size

Figure 6. Two Donut data

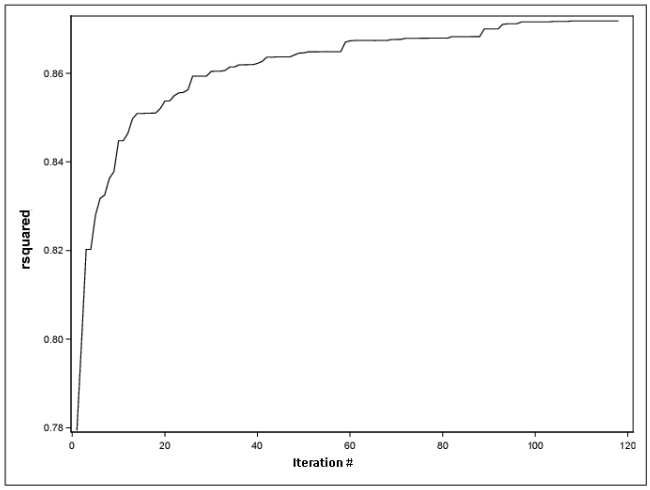


Figure 7. Plot of threshold R^2 - Banana shaped data (Sample size = 6)

5.2. Analysis of Tennessee Eastman Data

In this section we provide results of our experiments with high dimensional Tennessee Eastman data. The data was generated using the MATLAB simulation code (Ricker, 2002) which provides a model of an industrial chemical process (Downs & Vogel, 1993). The data was generated for normal operations of the process and twenty faulty processes. Each observation consists of 41 variables, out of which 22 are measured continuously, on an average, every 6 seconds and remaining 19 sampled at a specified interval either every 0.1 or 0.25 hours. We interpolated the 22 observations which are measured continuously using SAS® *EXPAND* procedure. The interpolation increased the observation frequency and generated 20 observations per second. The interpolation ensured that we have adequate data volume to compare performance our sampling method with full SVDD method.

We created a training data set of 5,000 randomly selected observations belonging to the normal operations of the process. From the remaining observations, we created a scoring data of 228,000 observations by randomly selecting 108,000 observations belonging to the normal operations and 120,000 observations belonging to the faulty processes. A SVDD model was first trained using all observations in the training data set. The training results were used to score the observations in the scoring data set to determine if the model could accurately classify an observation as belonging to the normal operations. The accuracy of scoring was measured using the F_1-measure. We then trained using the sampling method, followed by scoring to compute the F_1-measure again. The sample size for the sampling based method was set to 42 (number of variables + 1). Similar to the Shuttle data analysis, we measured the performance of

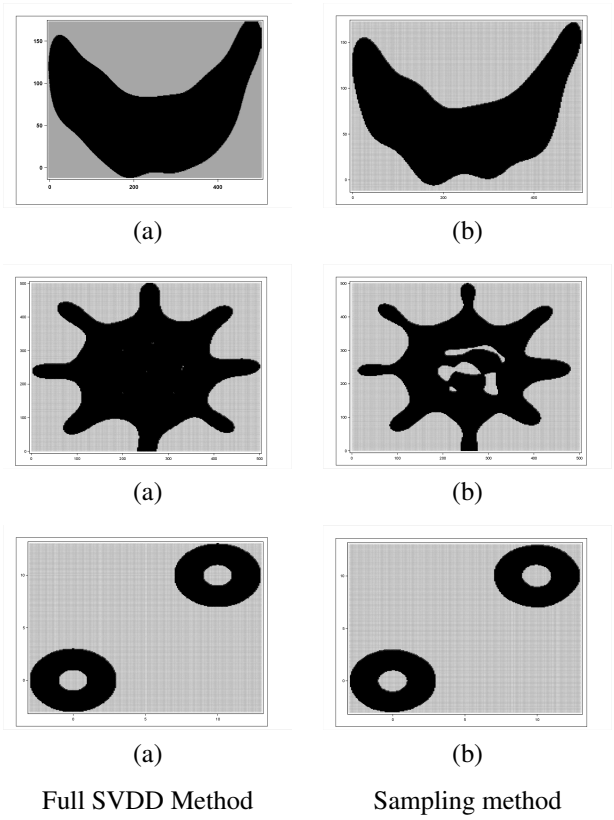


Figure 8. Scoring results. Above figures show results of scoring on a 200x200 data grid. Light gray color indicates outside points and black color indicates inside points. Figure (a) used full SVDD method for training. Figure (b) used sampling method for training.

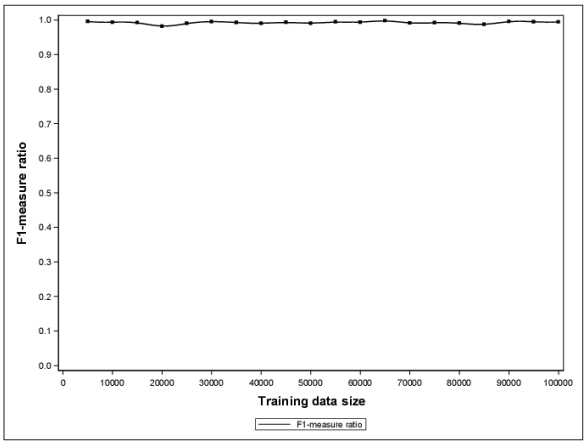


Figure 9. $F1$-measure plot: Shuttle data. *Sample size for sampling method=10*

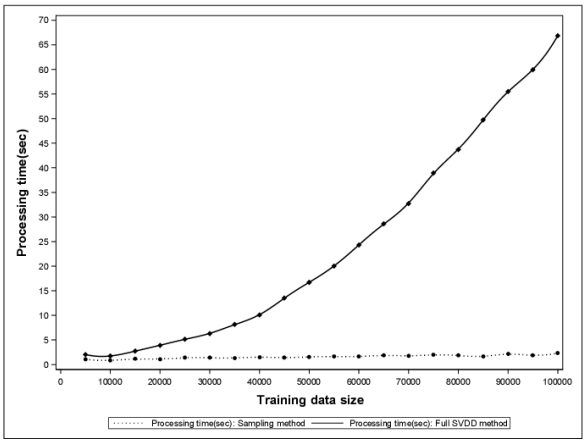


Figure 10. Processing time plot: Shuttle data. *Sample size for sampling method=10*

the sampling method using the F_1-measure ratio defined as $F_{\text{Sampling}}/F_{\text{Allobs}}$ where F_{Sampling} is the F_1-measure obtained when the value obtained using the sampling method for training, and F_{Allobs} is the value of F_1-measure computed when all observations were used for training. A value close to 1 indicate that sampling method is competitive with full SVDD method. We repeated the above steps varying the training data set of size from 10,000 to 100,000 in the increments of 5,000. The scoring data set was kept unchanged during each iteration. Figure 11 provides the plot of F_1-measure ratio. The plot of F_1-measure ratio was constant, very close to 1 for all training data set sizes, provides the evidence that the sampling method provides near identical classification accuracy as compared to full SVDD method. Figure 12 provides the plot of the processing time for the sampling-based method and the all obsrvation method. As the training data set size increased, the processing time for

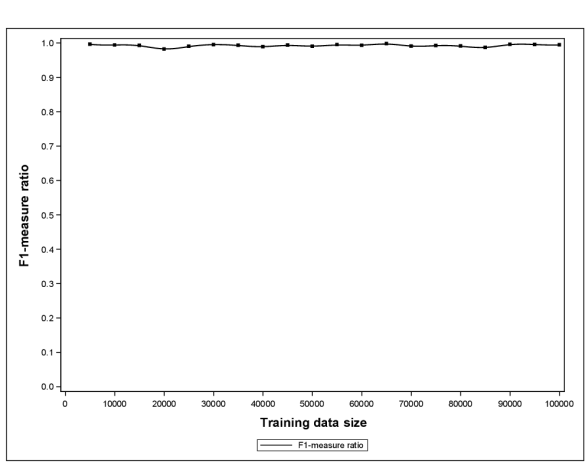


Figure 11. $F1$-measure ratio plot: Tennessee Eastman data. Sample size for sampling method=42

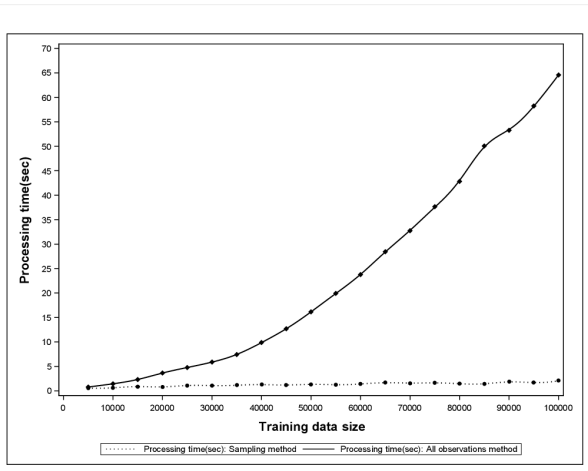


Figure 12. Processing time plot: Tennessee Eastman data. Sample size for sampling method=42

full SVDD method increased almost linearly to a value of about one minute for training data set of 100,000 observations. In comparison, the processing time of the sampling based method was in the range of 0.5 to 2.0 sec. The results prove that the sampling-based method is very efficient and it provides near identical results as compared to full SVDD method.

6. Simulation Study

In this section we measure the performance of Sampling method when it is applied to randomly generated polygons. Given the number of vertices, k,we generate the vertices of a randomly generated polygon in the anticlockwise sense as $r_1 \exp i\theta_{(1)}, \ldots, r_k \exp i\theta_{(k)}$. Here $\theta_{(i)}$'s are the order statistics of an i.i.d sample uniformly drawn from $(0, 2\pi)$

and r_i's are uniformly drawn from an interval $[\mathrm{r_{min}}, \mathrm{r_{max}}]$. For this simulation we chose $\mathrm{r_{min}} = 3$ and $\mathrm{r_{max}} = 5$ and varied the number of vertices from 5 to 30. We generated 20 random polygons for each vertex size. Figure 13 shows two random polygons. Having determined a polygon we randomly selected 600 points uniformly from the interior of the polygon to construct a training data set. Then for each polygon we found its bounding rectangle and divided it into a 200×200 grid. We then labeled each point on this grid as an "inside" or an "outside" point to create the scoring data set. We then fit SVDD on the training data set and scored using the corresponding scoring data set and calculated the F_1-measure. The process of training and scoring was first performed using the all observation method, followed by the sampling method. For sampling method we used sample size of 5. We trained and scored each instance of a polygon 10 times by changing the value of the Gaussian bandwidth parameter, s. We used s values from the following set.
$s = [1, 1.4, 1.8, 2.3, 2.7, 3.2, 3.6, 4.1, 4.5, 5]$.
Similar to the analysis of shuttle data and Tennessee Eastman data, we measured the performance of the sampling method using the F_1-measure ratio.

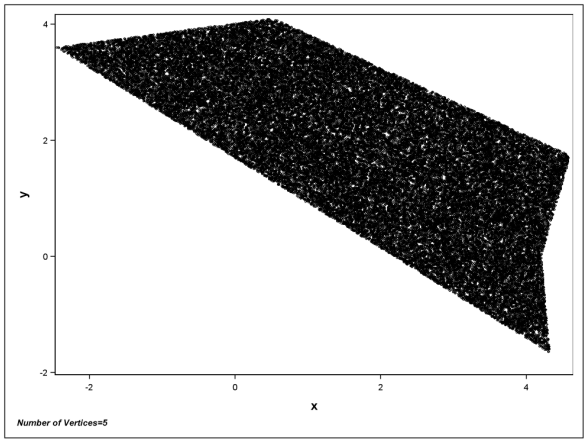
The Box-whisker plots in Figure 14 to 16 summarizes the simulation study results. The x- axis shows the number of vertices of the ploygon and y-axis shows the F_1-measure ratio. The bottom and the top of the box shows the first and the third quartile values. The ends of the whiskers represent the minimum and the maximum value of the F_1-measure ratio. The diamond shape indicates the mean value and the horizontal line in the box indicates the second quartile.

6.1. Comaprison of Best Value of s

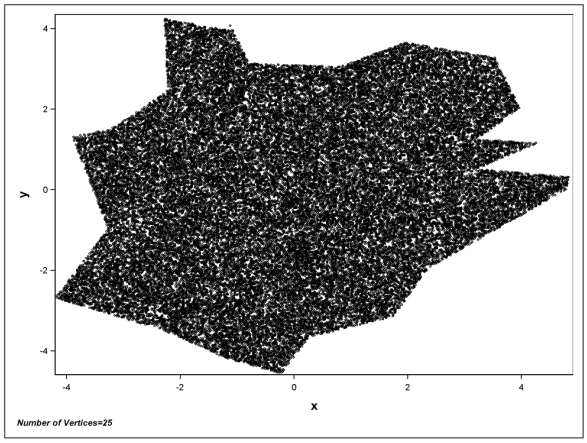
For each instance of a polygon, we looked at the best s value, which provided the maximum F_1 measure. The plot in Figure 14 shows the plot of F_1 measure ratio computed using the maximum values of F_1 measures. The plot shows that F_1-measure ratio is greater than ≈ 0.95 across all values of number of vertices. The F_1 measure ratio in the top three quartiles is greater than ≈ 0.97 across all values of the number of vertices. Using best possible value of s, the sampling method provides comparable results with full SVDD method.

6.2. Results Using Same Value of s

We evaluated sampling method against full SVDD method, for the same value of s. The plots in Figure 15 illustrate the results for different six different values of s. The plot shows that F_1-measure ratio is greater than 0.9 across number of vertice and s. In Figures 15 (c) to (f), the



(a) Number of Vertices = 5



(b) Number of Vertices = 25

Figure 13. Random Polygons

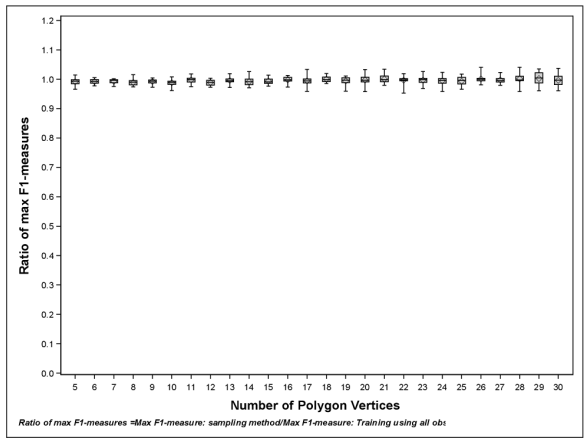
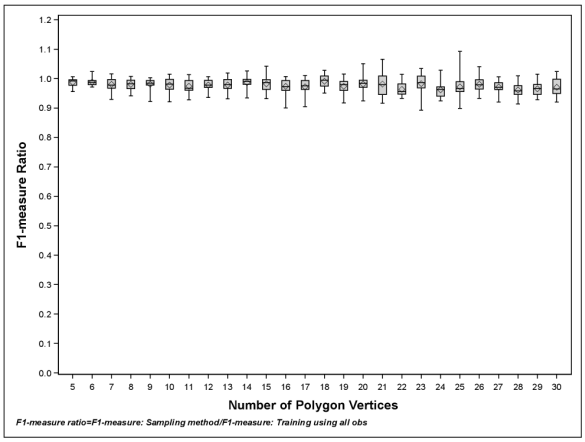


Figure 14. Box-whisker plot: Number of vertices vs. Ratio of max $F1$ measures

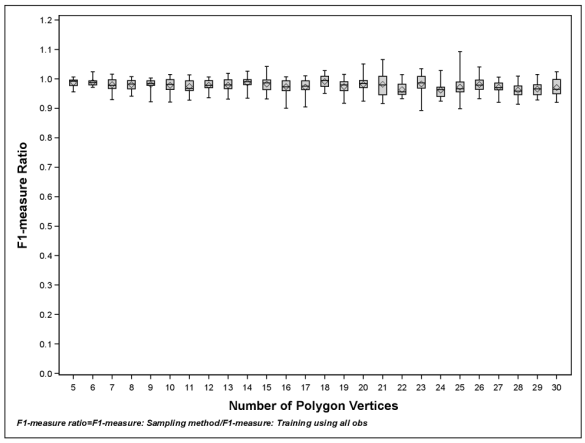
top three quartiles of F_1 measure ratio was consistently greater than ≈ 1. Training using sampling method and full SVDD method, using same s value, provide similar results.

6.3. Overall Results

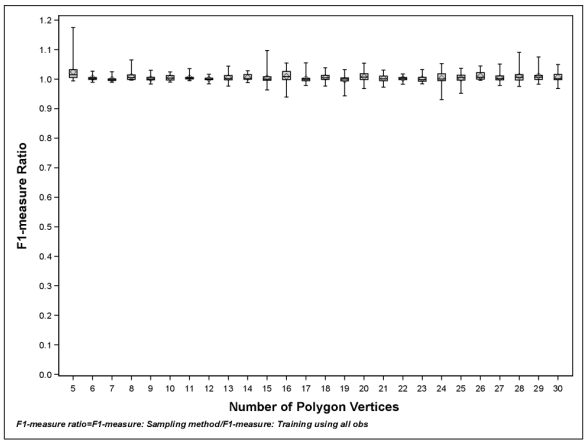
Figure 16 provides summary of all simulation performed for different polygon instances and varying values of s. The plot shows that F_1-measure ratio is greater than 0.9 across number of vertice. The F_1 measure ratio in the top three quartiles is greater than ≈ 0.98 across all values of the number of vertices. The accuracy of sampling method is comaprable to full SVDD method.



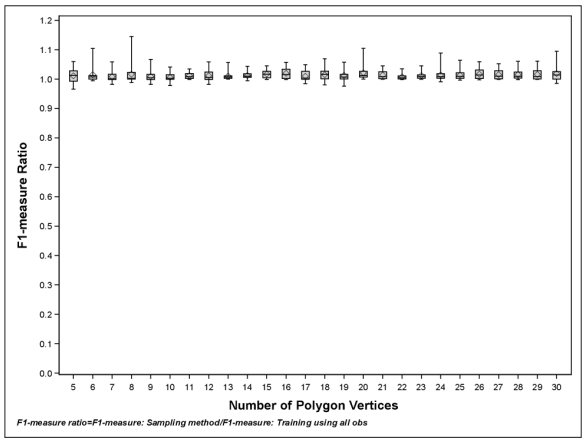
(a)s=1



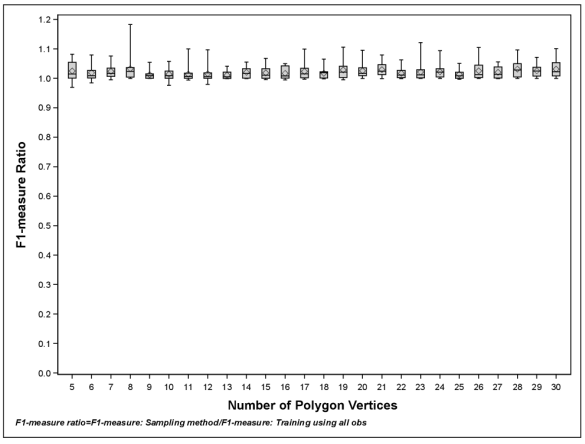
(b)s=1.4



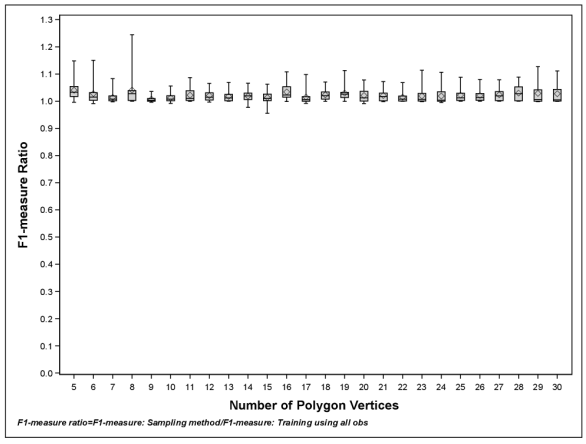
(c)s=2.3



(d)s=3.4



(e)s=4.1



(f)s=5.0

Figure 15. Box-whisker plot: Number of vertices vs. $F1$ measure ratio for different s values

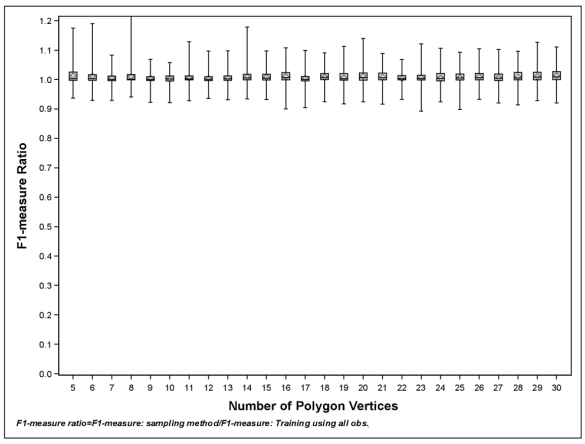


Figure 16. Box-whisker plot: Number of vertices vs. $F1$ measure ratio

7. Conclusion

We propose a simple sampling-based iterative method for training SVDD. The method incrementally learns during each iteration by utilizing information contained in the current master set of support vectors and new information provided by the random sample. After a certain number of iterations, the threshold R^2 value and the center a start to converge. At this point, the SVDD of the master set of support vectors is close to the SVDD of training data set. We provide a mechanism to detect convergence and establish a stopping criteria.

The simplicity of proposed method ensures ease of implementation. The implementation involves writing additional code for calling SVDD training code iteratively, maintaining a master set of support vectors and implementing convergence criteria based on threshold R^2 and center a. We do not propose any changes to the core SVDD training algorithm as outlined in section 1.1

The method is fast. The number of observations used for finding the SVDD in each iteration can be a very small fraction of the number of observations in the training data set. The algorithm provides good results with sample size as small as $m + 1$, where m is the number of variables in the training data set. Small sample size ensures that each iteration of the algorithm is extremely fast.

The proposed method provides a fast alternative to traditional SVDD training method which uses information from all observations in one iteration. In applications where training data set is large, fast approximation is often desired than obtaining exact description, which can be significantly slower. Within the broader realm of Internet of Things (IoT) we expect to see multiple applications of SVDD especially to monitor industrial processes and equipment health. Such applications require fast periodic training using large data sets, which can be performed very efficiently using our sampling method outlined in this paper.

A. Appendix

A.1. Review of Existing Methods

A brief description of two methods which use sampling for SVDD training is provided in this section.

A.1.1. Decomposition and Combination method of Luo et.al.(Luo et al., 2010)

Let S denote the set of observations in the training dataset.
Step 1: Select a random sample of size m from S. Let this sample be W_1. Perform SVDD on W_1. Let P_1 denote the corresponding set of support vectors and points enclosed by the support vectors. Let X_1 be the support vectors of (W_1, P_1). Let V_1=$S \backslash P_1$ denote the set of points which belong to S and outside of P_1. Let v_1 denote number of points in V_1. If $v_1 \leq m$, then P_1 is the final solution. If $v_1 \geq m$ then select a random set of m points from V_1. Let this set be W_2.
Step 2: Perform SVDD on W_2. Let P_2 denote the corresponding set of support vectors and points enclosed by the support vectors. Let X_2 be the support vectors of (W_2, P_2).
Step 3: Construct set R as $(X_1 \cup X_2)$. Perform SVDD on R. Let P denote the corresponding set of support vectors and points enclosed by the support vectors. Let Q denote support vectors of (R, P). Let V=$S \backslash P_1$ denote the set of points which belong to S and outside of P. Let v denote number of points in V. If $v \leq m$, then (S, P) is the final solution. If $v \geq m$ then select a random set of m points from V_1. Let this set be W_2.
Step 4: Repeat step 2-3 until solution is found.

This method involves combining SVDD results of the random sample (W_1) selected from training dataset and a sample selected from data points which fall outside the description (W_2). To determine points outside the data description, and obtain W_2, training dataset S needs to be scored using training results of W_1. Until the solution is obtained, each iteration involves one scoring action on the entire training dataset. For large training data sets, multiple scoring actions can make this method very slow.

A.1.2. K-means Clustering Method of Kim et.al.(Kim et al., 2007)

In this method, the training dataset of size N is first partitioned into k subsets using K-means clustering. For each subset a local description using SVDD is obtained. Support vectors of each subset are then combined to obtain, what authors refer to as a working set. SVDD is then applied

on the working set to obtain data description of the entire training dataset. If α is the average number of support vectors for each sub-problem's description, the complexity of the algorithm is expressed as:
$\mathcal{O}(kN) + k\mathcal{O}((N/k)^3) + \mathcal{O}((\alpha k)^3)$
where αk is the size of the working set.
This is a classic *divide and conquer* algorithm. Information from each observation in the training dataset is utilized to find solution. For large dense data sets, use of all observations is unnecessary and can make this method very slow.

A.2. Prior Art

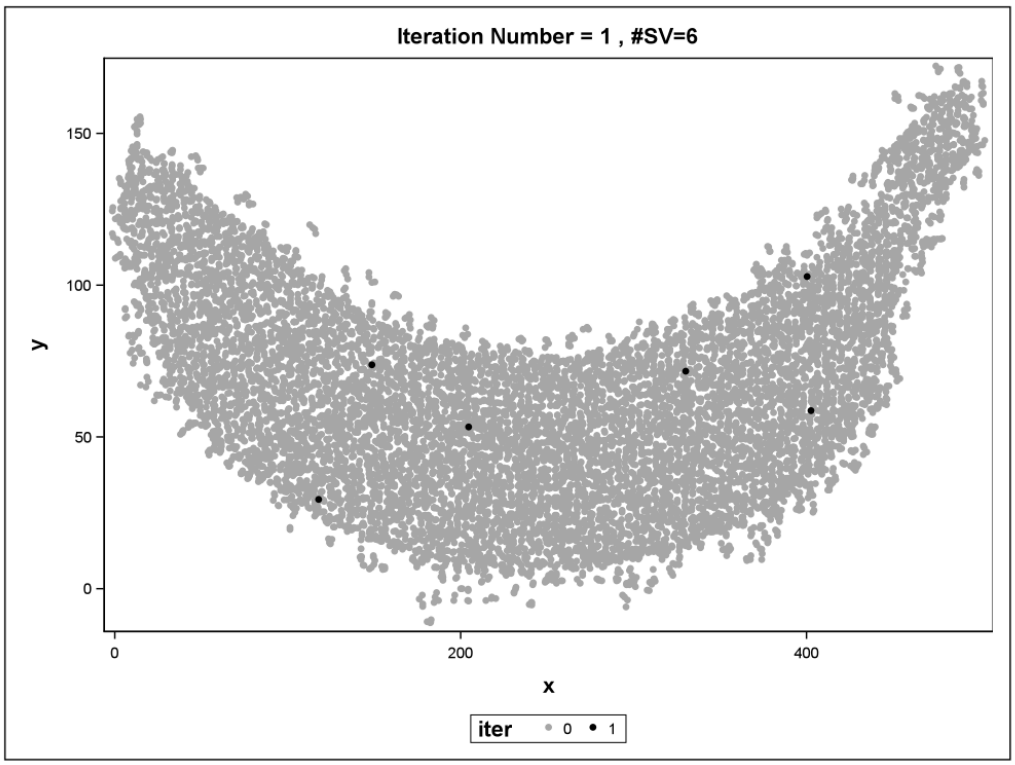
Following table summarizes results of our prior art search.

Paper	Prior-art	Reason
A Fast SVDD algorithm based on decomposition and combination for fault detection by Luo et. al.	No	This method involves combining SVDD results of the random sample selected from training dataset and a sample selected from data points which fall outside the description. To determine points outside the data description,training dataset needs to be scored once per iteration. Our method does not involve scoring.
Fast support vector data description using k-means clustering by Kim et. al.	No	This is a classic devide and conqure algorithm. It does not involve sampling. The method does not provide any convergence criteria.
Incremental SVDD training: Improving efficiency of background modelling in videos by Tavakkoli et.al.	No	Method works by iteratively updating a set of support vectors. Method works only for two dimensional data and it does not provide a convergence criteria.
Fast training of SVDD by extracting boundary targets by Liang et. al.	No	Method obtains solution by identifying a subset of training data which can potentially contain the support vectors. This is a totally different approach than our method.
K-Farthest-Neighbors-Based concept boundary determination for support vector data description by Xiao et.al.	No	Method obtains solution by identifying a subset of training data which can potentially contain the support vectors and identifying which cannot. This is a totally different approach than our method.
Effective training set sampling strategy for SVDD anomaly detection in hyperspectral imagery by Ergul et. al.	No	Method uses principal component analysis results to decide if a data point should be part of the sample. Our approach uses random sampling.

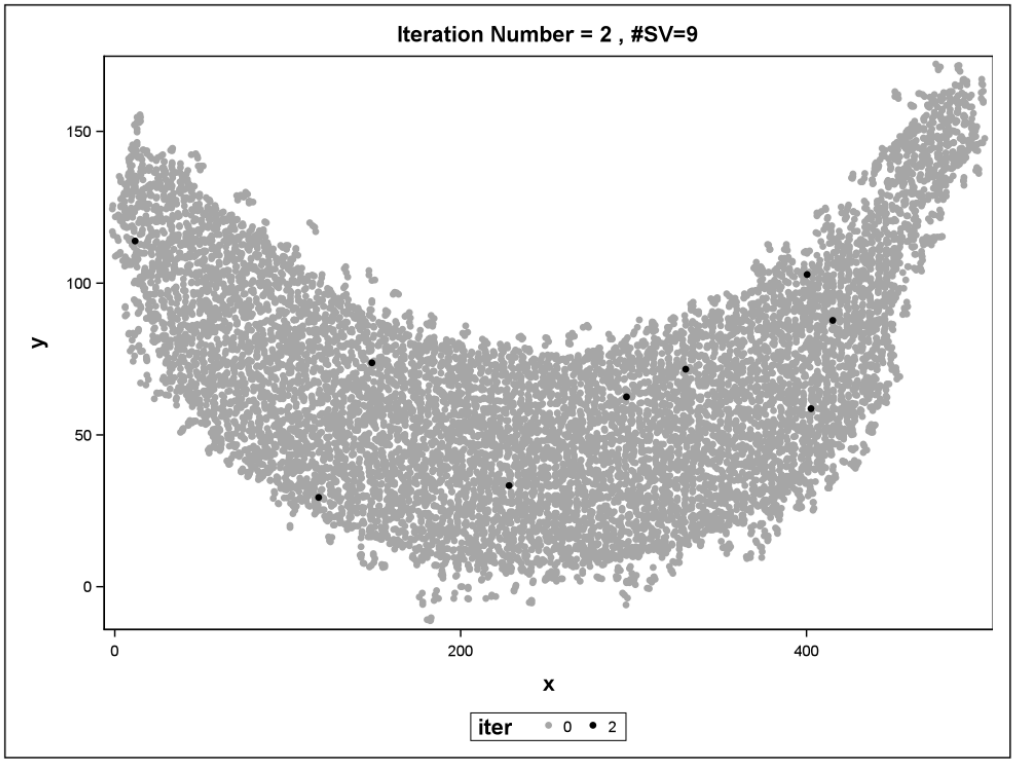
Table 3. Prior-art search summary

A.3. Iterative solution using sampling method:

The development of final solution for the Banana-shaped data set is illustrated below. Figure 17 shows several plots indicating support vectors against the backdrop of the training data set at the end of an iteration. The black markers indicate the support vectors and the gray markers indicate the training data points. The iteration number and number of support vectors is indicated in each plot. At lower iteration numbers the support vectors were in the interior of the Banana-shape. As number of iterations increased, the sampling method moved the the support vectors towards the data boundary. At and near convergence, the support vectors were primarily along the data boundary.



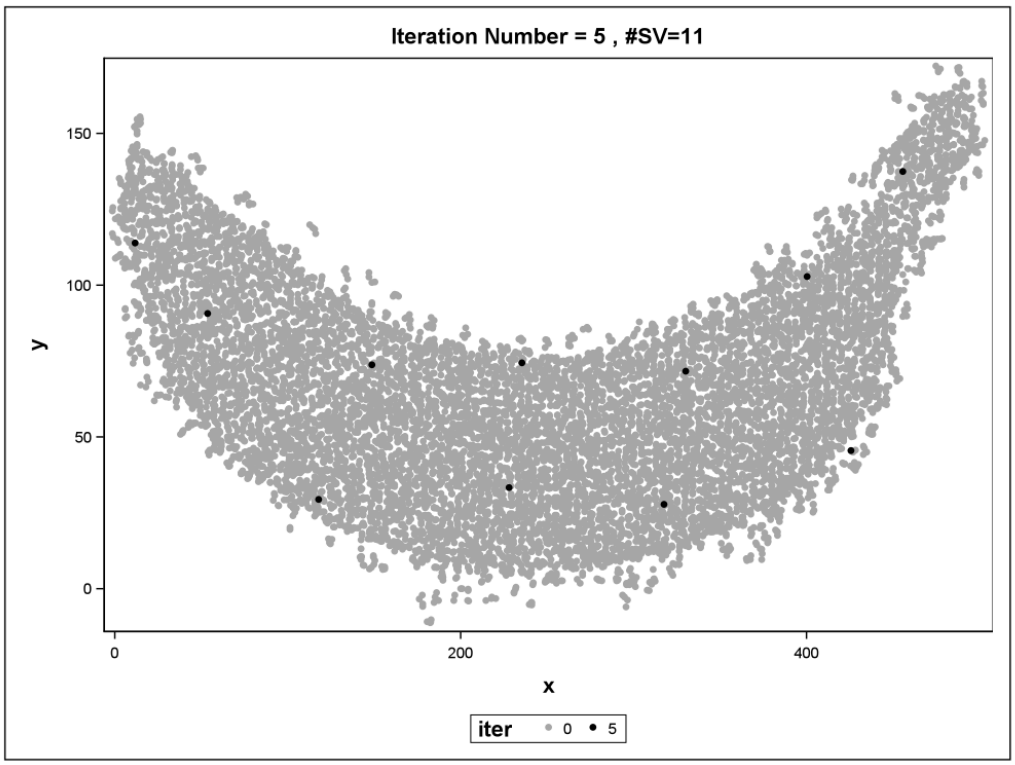
(a)



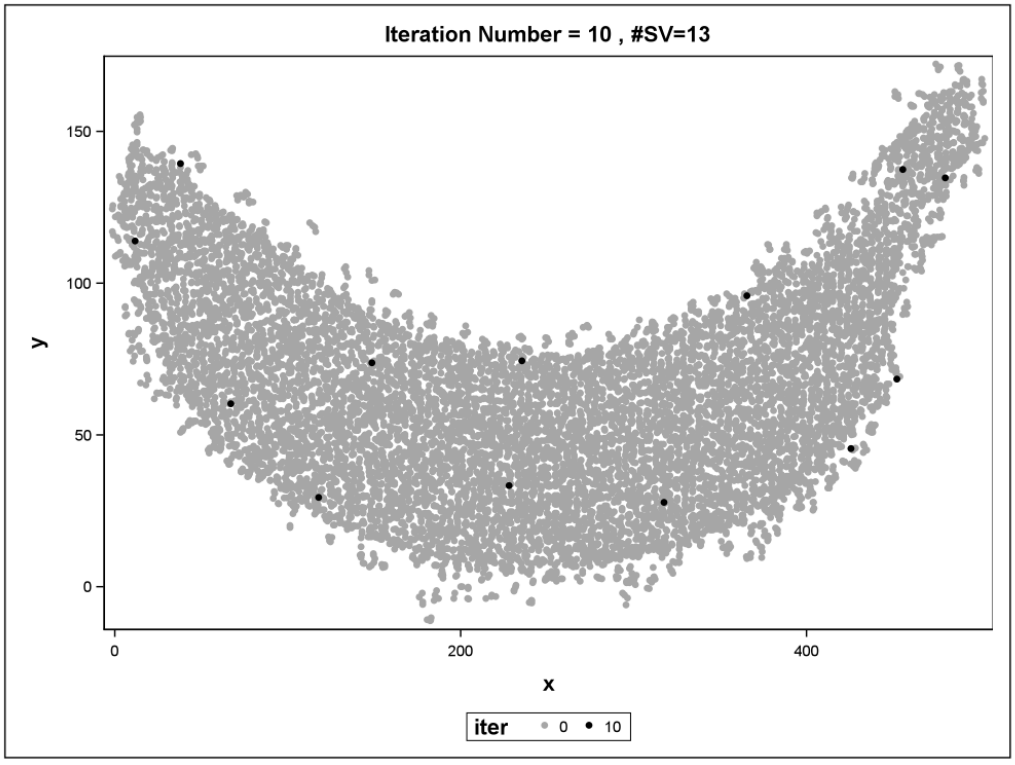
(b)

Figure 17: Sampling Method Results

Gray points indicate training data obsevations. Black points indicate support vecotrs at the iteration end.



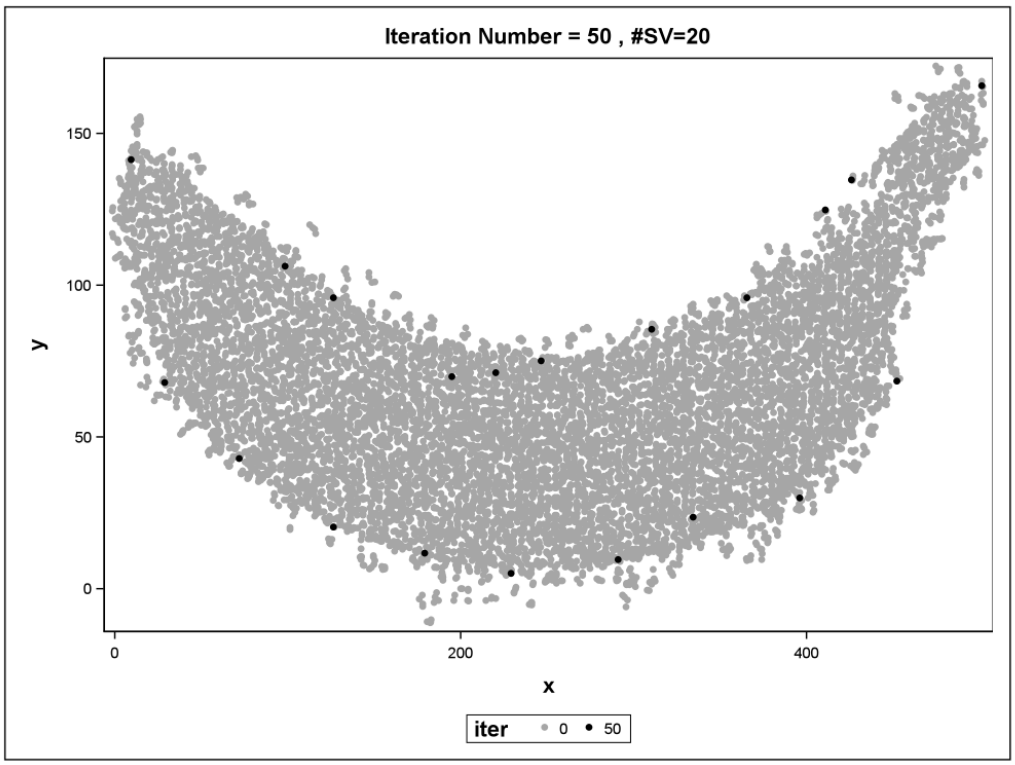
(c)



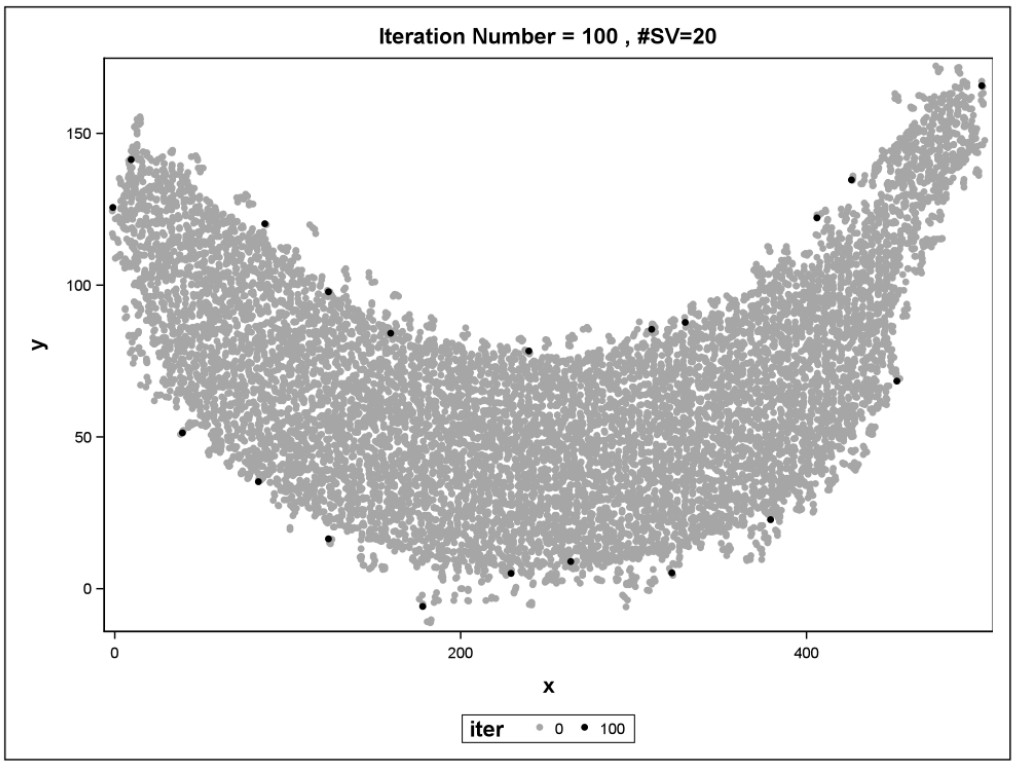
(d)

Figure 17: Sampling Method Results

Gray points indicate training data obsevations. Black points indicate support vecotrs at the iteration end.



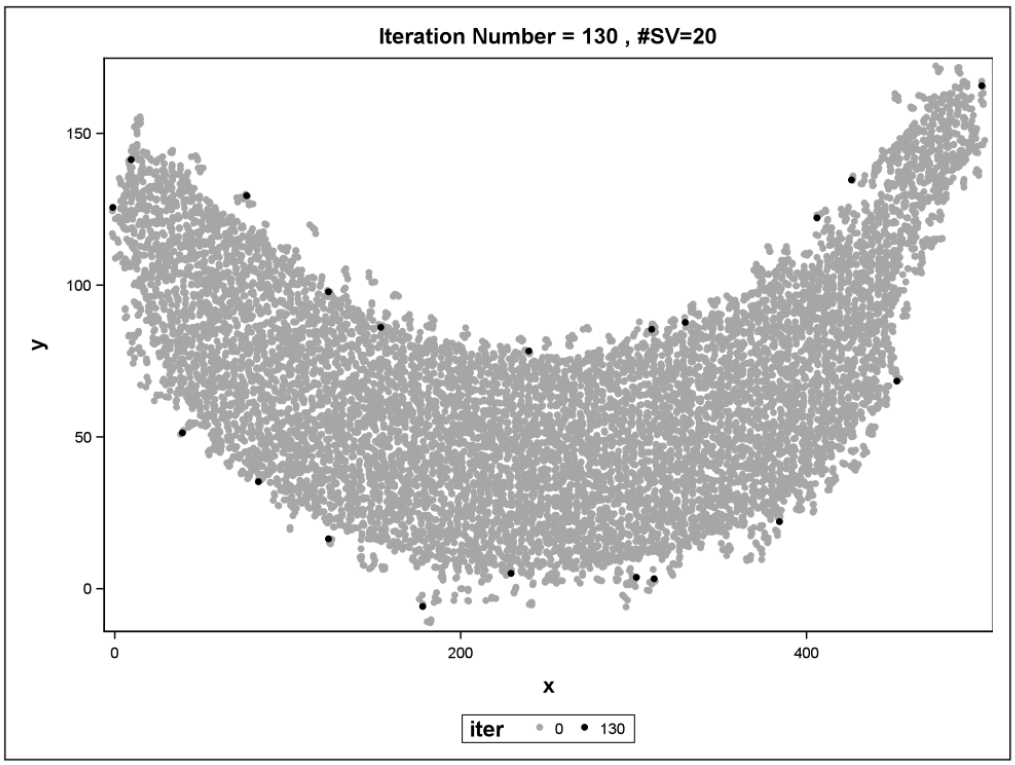
(e)



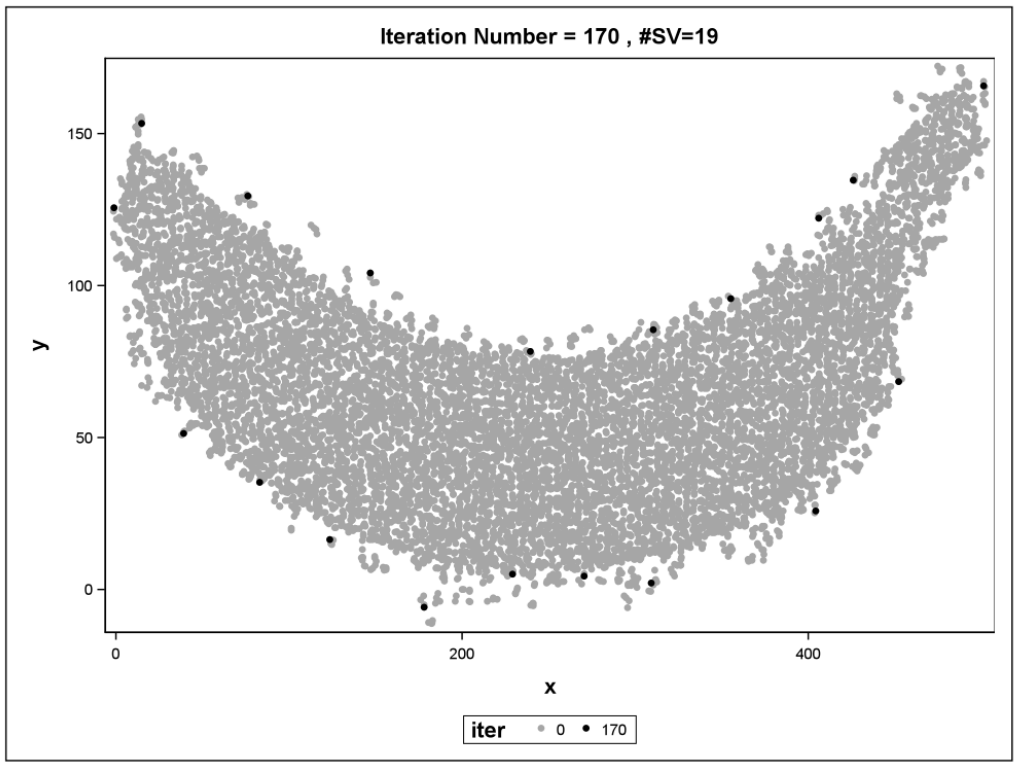
(f)

Figure 17: Sampling Method Results

Gray points indicate training data obsevations. Black points indicate support vecotrs at the iteration end.



(g)



(h)

Figure 17: Sampling Method Results

Gray points indicate training data obsevations. Black points indicate support vecotrs at the iteration end.

References

Chang, Chih-Chung and Lin, Chih-Jen. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

Downs, James J and Vogel, Ernest F. A plant-wide industrial process control problem. *Computers & chemical engineering*, 17(3):245–255, 1993.

Ege, Gul. Multi-stage modeling delivers the roi for internet of things. http://blogs.sas.com/content/subconsciousmusings/2015/10/09/multi-stage-modeling-delivers-the-roi-for-internet-of-things/--is-epub/, 2015.

Kim, Pyo, Chang, Hyung, Song, Dong, and Choi, Jin. Fast support vector data description using k-means clustering. *Advances in Neural Networks–ISNN 2007*, pp. 506–514, 2007.

Lichman, M. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

Luo, Jian, Li, Bo, Wu, Chang-qing, and Pan, Yinghui. A fast svdd algorithm based on decomposition and combination for fault detection. In *Control and Automation (ICCA), 2010 8th IEEE International Conference on*, pp. 1924–1928. IEEE, 2010.

Ricker, N. Lawrence. Tennessee eastman challenge archive, matlab 7.x code, 2002. URL http://depts.washington.edu/control/LARRY/TE/download.html. [Online; accessed 4-April-2016].

Sanchez-Hernandez, Carolina, Boyd, Doreen S, and Foody, Giles M. One-class classification for mapping a specific land-cover class: Svdd classification of fenland. *Geoscience and Remote Sensing, IEEE Transactions on*, 45 (4):1061–1073, 2007.

Scholkopf, Bernhard, Williamson, Robert C, Smola, Alexander J, Shawe-Taylor, John, Platt, John C, et al. Support vector method for novelty detection. In *NIPS*, volume 12, pp. 582–588. Citeseer, 1999.

Sukchotrat, Thuntee, Kim, Seoung Bum, and Tsung, Fugee. One-class classification-based control charts for multivariate process monitoring. *IIE transactions*, 42(2): 107–120, 2009.

Tax, David MJ and Duin, Robert PW. Support vector data description. *Machine learning*, 54(1):45–66, 2004.

Widodo, Achmad and Yang, Bo-Suk. Support vector machine in machine condition monitoring and fault diagnosis. *Mechanical Systems and Signal Processing*, 21(6): 2560–2574, 2007.

Ypma, Alexander, Tax, David MJ, and Duin, Robert PW. Robust machine fault detection with independent component analysis and support vector data description. In *Neural Networks for Signal Processing IX, 1999. Proceedings of the 1999 IEEE Signal Processing Society Workshop.*, pp. 67–76. IEEE, 1999.

Zhuang, Ling and Dai, Honghua. Parameter optimization of kernel-based one-class classifier on imbalance learning. *Journal of Computers*, 1(7):32–40, 2006.