Fast Support Vector Data Descriptions for Novelty Detection Project Report[*]

Aravind Sankar
CS11B033
Department of Computer Science
IIT Madras
aravindsankar28@gmail.com

Adit Krishnan
CS11B063
Department of Computer Science
IIT Madras
adit101293@gmail.com

Abstract

Our term paper titled ”Fast SVDD for Novelty Detection” proposes a way to reduce the testing time complexity of the SVDD technique by preventing the need for computing the Kernel function value for all support vectors. The details of this are illustrated in the following section. We have tested this against C-SVDD and ANNs for three different datasets and provided our observations and interpretations for the same.

1 Brief Summary

1.1 Introduction

In this paper, the authors address the common well-studied problem of Novelty Detection. In this task, we have a 2-class problem denoted by the normal and abnormal class points, where the number of points of abnormal class is lesser than that of the normal class. We note that this problem is different from the traditional classification problem in that only points of the normal class are used for the purpose of training, essentially a One-Class classification problem.

The Support Vector Data Description (SVDD)[1] is an important and popular technique which has been used to solve the problem of Novelty Detection. In short, SVDD tries to construct a soft minimal hypersphere enclosing the points of the target class (normal) in the kernel feature space (ϕ). This hypersphere boundary in the kernel feature space is used for prediction to identify the outlier points (novel data). It can also be shown that the decision boundary between the 2 classes is a hyperplane in the kernel feature space if a normalized kernel function is used. SVDD like any other SVM, has a prediction time complexity linear in the number of support vectors (N_s) because N_s evaluations of the kernel function is required for predicting the label of an unseen data point.

The authors attempt to address this key issue in this paper by proposing Fast SVDD (F-SVDD) [2] which replaces the kernel expansion (of N_s terms) in the decision function with just 1 term. This is achieved by first looking for a vector (called agent of the center) in the ϕ-space whose preimage $\hat{x}$ exists in the x-space. Then, they express the sphere center a_ϕ as a scalar multiple of this vector in the ϕ-space. Now, it can be seen the decision function involves only one kernel function between $\hat{x}$ and any unseen data point x. This means that the prediction time complexity is constant, irrespective

[*]As a part of the course Kernel Methods for Pattern Analysis - CS6011 Jan - May 2015

of the size of the training dataset and the number of support vectors. The authors also propose an efficient method for solving the preimage problem, which is non-iterative and faster than the existing techniques. The approaches used to solve these problems will be explained in the further sections.

1.2 SVDD and some properties

In this section, we introduce the SVDD problem definition for the purpose of notation. The optimization problem that we solve for involves a_ϕ (center of the sphere), $\{\xi_i\}_{i=1}^N$ (slacks) and R (radius of the hypersphere) as the optimization variables while C is a user specified cost parameter. We assume that we have a training dataset $D = \{x_i\}_{i=1}^N$. The primal and dual problems are shown below :

Primal problem (with constraints) :

$$\begin{gathered} \min R^2 + C\sum_{i=1}^{N}\xi_i \\ \text{subject to} \\ ||\phi(x_i) - a_\phi||^2 \leq R^2 + \xi_i \\ \xi_i \geq 0 \;\; \forall i = 1, ...N \end{gathered} \tag{1}$$

Dual problem:

$$\begin{gathered} \max 1 - \sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j K(x_i, x_j) \\ \text{subject to} \\ \sum_{i=1}^{N}\alpha_i = 1 \\ 0 \leq \alpha_i \leq C \;\; \forall i = 1, ..., N. \end{gathered} \tag{2}$$

As usual, the lagrange multipliers $\{\alpha_i\}_{i=1}^N$ are associated with the primal constraint involving ξ_i's. Here, the discriminant function $g(x)$ is given by :

$$g(x) = ||\phi(x_i) - a_\phi||^2 - R^2 = c - 2\sum_{i=1}^{N_s}\alpha_i K(x, x_i)$$

where c is a constant.

The sphere center a_ϕ is given by :

$$a_\phi = \sum_{i=1}^{N}\alpha_i\phi(x_i) = \sum_{i=1}^{N_s}\alpha_i\phi(x_i)$$

We also compute $||a_\phi||$ as it will be used later,

$$||a_\phi||^2 = ||\sum_{i=1}^{N_s}\alpha_i\phi(x_i)||^2 = \alpha^T K\alpha \tag{3}$$

where K is the kernel gram matrix.

Let the soft minimal hypersphere (constructed by SVDD over the training points) be denoted as B_S. We assume the use of a gaussian kernel function (can be extended for any normalized kernel also) so that $K(x,x) = 1$. So, all points $\phi(x)$ lie on a unit hypersphere (centered at origin O_ϕ) in the kernel feature space, denoted by B_ϕ. This leads to interesting geometrical properties in the kernel feature space. One main property is mentioned below :

Property : The center a_ϕ of the SVDD hypersphere B_S must lie inside the unit hypersphere B_ϕ.

This property shows that the exact preimage of a_ϕ in the x-space does not exist. This means that, either only an approximate preimage can be found, or a different way needs to be found. F-SVDD does this by identifying the *agent* of the center whose pre-image exists in the x-space. The following sections explain the process of finding the agent and the corresponding preimage.

1.3 Fast SVDD

The main objective of F-SVDD is to improve classification speed by reducing the number of computations at prediction time. By the above property, the preimage of the center a_ϕ does not exist. This algorithm first defines the agent of the center.

1.3.1 Agent of the center

Let the agent of the center be denoted by Ψ_a (in the kernel feature space). We want Ψ_a to have a pre-image $\hat{x}$ in the x-space. This directly implies that Ψ_a should lie on the unit hypersphere B_ϕ centered at Origin in the kernel feature space. This gives us

$$||\psi_a|| = 1 \tag{4}$$

Ψ_a is identified as the point closest to a_ϕ that lies on B_ϕ. This point is obtained by extending a_ϕ to intersect B_ϕ. obviously is in the direction of a_ϕ. Thus, we can express Ψ_a as :

$$\Psi_a = \gamma a_\phi$$

where γ is a scalar.

$$\begin{aligned} ||\Psi_a|| &= \gamma ||a_\phi|| \\ \gamma &= \frac{1}{||a_\phi||} \text{ (Using 4)} \\ \gamma &= \frac{1}{\sqrt{\alpha^T K \alpha}} \text{ (Using 3)} \end{aligned} \tag{5}$$

We also note that $\gamma > 1$ as $||a_\phi|| < 1$.

This shows us that if we are able to obtain the preimage of the agent, the computation of our discriminant function reduces to just 1 term, which involves just $\hat{x}$. In the next section, we describe how the pre-image is computed for the agent.

1.3.2 Preimage of the agent :

Formally, we define the preimage problem as, given a feature vector ψ in the kernel feature space, find $\hat{x} \in \mathbf{R}^d$ (x-space) such that $\psi = \phi(x)$. The preimage $\hat{x}$ is found by solving the problem :

$$\hat{x} = \min_{\hat{x}} ||\Psi_a - \phi(\hat{x})||^2$$

It can shown that solving this problem is equivalent to finding the approximate preimage of a_ϕ, i.e.

$$\min \rho = ||a_\phi - \phi(\hat{x})||^2$$

To minimize ρ, we set the derivative of the objective function (ρ) to 0 and we obtain :

$$\hat{x} = \frac{\sum_{i=1}^{N} \alpha_i K(\hat{x}, x_i) x_i}{\sum_{i=1}^{N} K(\hat{x}, x_i)}$$

We see that $\hat{x}$ appears on both sides of the above equation. One of the ways to solve such an equation is shown below :

Fixed point iteration

Fixed point iteration is an existing technique for computing fixed points of iterated functions. It can be used here to solve for $\hat{x}$. Initially, $\hat{x}$ is set to a random value x_0 and then an iterative procedure is followed to get the final fixed value of $\hat{x}$. The equation below shows the procedure :

$$\hat{x}_{t+1} = \frac{\sum_{i=1}^{N} \alpha_i K(\hat{x}_t, x_i) x_i}{\sum_{i=1}^{N} \alpha_i K(\hat{x}_t, x_i)}$$

Though the above method will converge to the optimal value of $\hat{x}$ (in general), it faces the problems of local minima and it also requires multiple initial guesses for $\hat{x}$ which leads to an increase in computation time. In order to solve these issues, the authors have proposed a direct pre-image finding method, which is given below :

Direct Method

In this technique, some simple properties of kernel functions are used to arrive at a closed form expression for $\hat{x}$.

$$\hat{x} = \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j K(x_i, x_j) x_i}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j K(x_i, x_j)}$$

We can clearly see that the problems faced by the fixed point iteration method have been eliminated here.

1.3.3 Discriminant function

Finally, the discriminant function $g(x)$ is given by

$$\begin{aligned} g(x) &= ||\phi(x) - a_\phi||^2 - R^2 = ||\phi(x) - \frac{\phi(\hat{x})}{\gamma}|| - R^2 \\ &= c' - \frac{2}{\gamma} K(x, \hat{x}) \\ &\text{where } c' = 1 - R^2 + 1/\gamma^2 \text{ is a constant.} \end{aligned} \tag{6}$$

Thus, we can see that the $g(x)$ has only one kernel function term, which makes the prediction time complexity a constant.

2 Experiments

In this section, we describe the experiments that were performed to compare the performance of F-SVDD with C-SVDD. Specifically, we compare F-SVDD - 1 (direct preimage finding method), F-SVDD - 2 (fixed point iteration method) and C-SVDD (basic SVDD formulation) for the performance in terms of generalization performance (accuracy) and classification speed (training and testing). In addition to this, we also compare the generalization performance against a non-kernel method which is that of a Multi-Layer Feed Forward Neural Network (MLFFNN) [3]. MATLAB was used to implement all of the three SVDD algorithms, specifically using the optimization toolbox. MLFFNN was implemented using the default MATLAB toolbox.

We use 2 real world datasets obtained from the UCI ML[1] repository and one synthetic dataset (from our assignment) for our evaluation. We also use another synthetic dataset to illustrate the differences between F-SVDD and C-SVDD.

All the experiments were performed on a system with Intel 2.3 GHz i5 processor and 6 GB main memory.

In all our experiments (expect overlapping dataset) involving SVDD and associated algorithms, the best values of the parameters C and σ were found by performing 10-fold cross validation. These values were obtained by performing cross validation for the C-SVDD algorithm. For maintaining fairness of results, the same values of C and σ were used for the F-SVDD algorithms also. For the MLFFNN, we performed cross-validation to estimate the number of nodes in the hidden layer.

In the paper, it has been mentioned that all of the training points are used as support vectors for C-SVDD, i.e. they do not perform a thresholding on the lagrange multipliers obtained due to precision issues. This could be one of the reasons why the testing times reported for C-SVDD in the paper are quite high in comparison to the F-SVDD method.

The neural network is traditionally used only for multi-class (at least 2) classification problems. In our scenario, we are required to use ony point of the target class for training. So, it becomes a one-classification problem. We use a back propagation neural network with 1 hidden layer in our experiments. The number of nodes in the hidden layer is estimated using cross-validation. We use a sigmoid activation function (logistic) at the hidden and output layers. So, the range of the outputs is from $[0, 1]$. Since the neural network tries to only fit the points of the target class, most of the outputs will be very close to 1. So, we cannot use the default threshold of 0.5 to classify points as normal/abnormal. Hence, threshold of 0.9 is used in all of our experiments. A very high value of threshold indicates that we are trying to fit a tight decision surface around the target class points because the values of the output node decrease as points move away from the core of the target class points.

Evaluation metric :

The metric used for evaluating the generalization performance is TAER (Total Average Error Rate) given by :

$$\texttt{TAER} = \frac{\texttt{Number of misclassified points}}{\texttt{Total number of points}}$$

The values of TAER reported in the results section are the average values obtained over 50 random samples of training data (the rest of the points used for testing).

We also evaluate the most important objective of this paper, which is the classification speed. We report both the training and testing times for each of the SVDD algorithms.

In the following subsections, we analyze the results obtained on each of the three datasets.

[1]http://archive.ics.uci.edu/ml/

2.1 Overlapping classes dataset :

This is the same dataset that was used in our assignments. We use this for visualization and interpreting since it is 2 dimensional. In this dataset, we have 4 classes which are overlapping and hence linearly non-separable. So, a 100 % accuracy cannot be expected for any classification/ novelty detection task on this dataset. The scatter plot of the dataset is shown below :

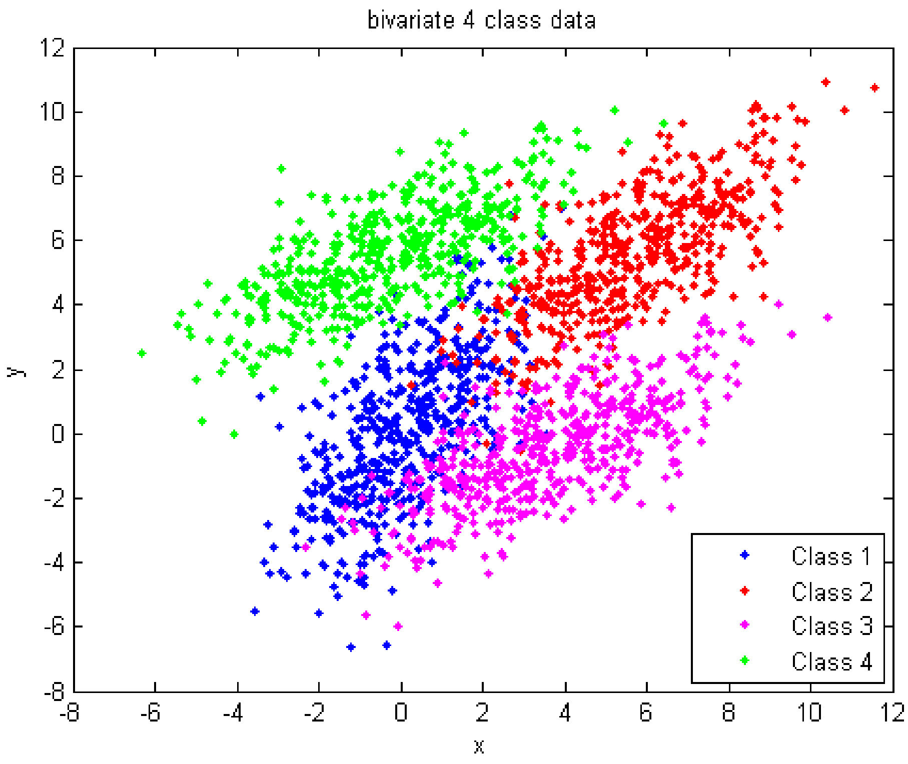


Figure 1: Overlapping classes dataset with 4 classes

The above plot shows the un-normalized dataset, while we have used the min-max normalization to reduce the range of each coordinate to $[0, 1]$.

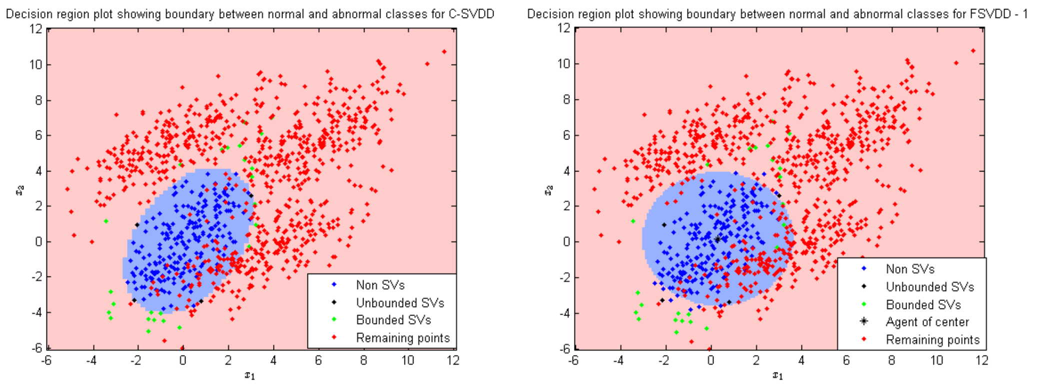
For the task of Novelty detection , we need to define our target and outlier class points. So, we perform 4 experiments where in experiment$_i$ points of class i are taken as the target (or normal) points, while the rest are considered as outliers. The size of training set was 250, while the sizes of validation and test sets were 600 and 400 respectively.

In this case, we are given explicit validation and test datasets. So, we find out the best values of C and σ only over the given validation set. The best value of σ obtained was 0.3371. Also, the value of C was estimated for each of the 4 experiments and the range of values of C obtained was from 0.02 to 0.04.

Table 1: Comparison of Average Error rates on Overlapping classes dataset

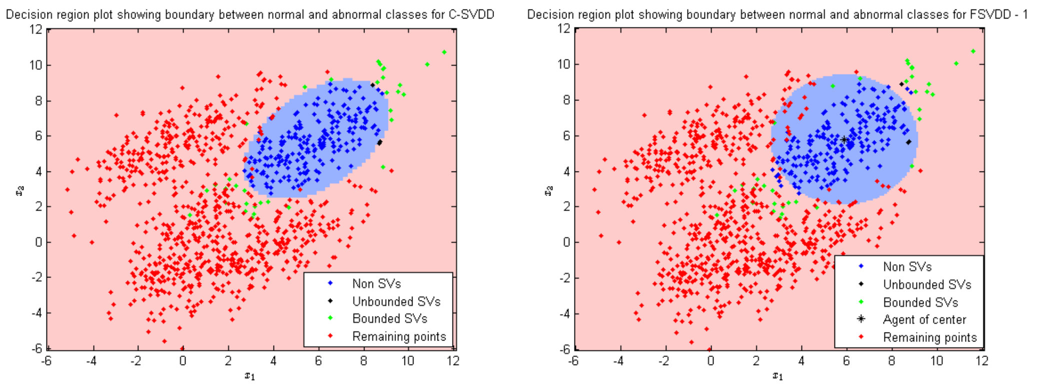
Target class	Evaluation set	C-SVDD	F-SVDD-1	F-SVDD-2	MLFFNN
1	Train	0.1240	0.1480	0.1360	0.0640
(N = 250)	Validation	0.1117	0.1750	0.1733	0.2017
	Test	0.1475	0.1925	0.1925	0.2450
2	Train	0.1160	0.1640	0.1640	0.0520
(N = 250)	Validation	0.1117	0.0867	0.0817	0.1633
	Test	0.1117	0.0800	0.0800	0.1725
3	Train	0.1880	0.2080	0.2080	0.0840
(N = 250)	Validation	0.0817	0.1100	0.1100	0.2600
	Test	0.0875	0.1200	0.1200	0.2875
4	Train	0.1040	0.1520	0.1760	0.1565
(N = 250)	Validation	0.0467	0.1217	0.1250	0.3340
	Test	0.0500	0.1450	0.1525	0.3560
	Average test error	0.0992	0.1344	0.1363	0.2650

The decision region plots obtained for C-SVDD and F-SVDD for each of the 4 classes are shown below :



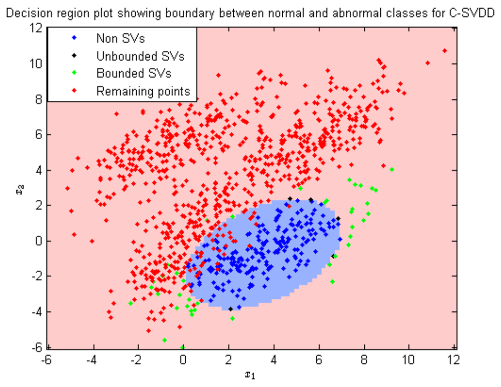
(a) C-SVDD (b) F-SVDD

Figure 2: Decision region plot with class-1 points as target class

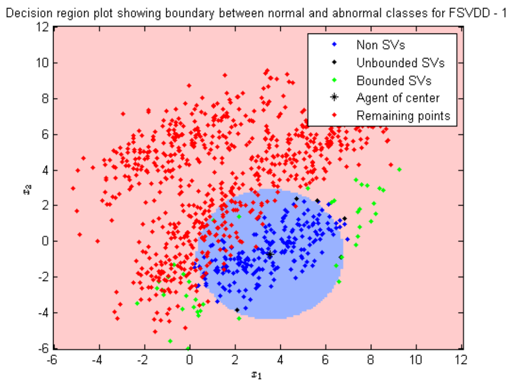


(a) C-SVDD (b) F-SVDD

Figure 3: Decision region plot with class-2 points as target class

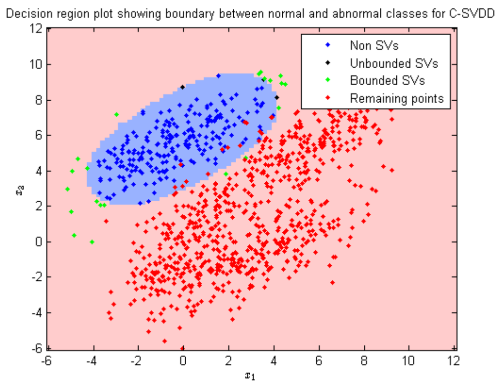


(a) C-SVDD

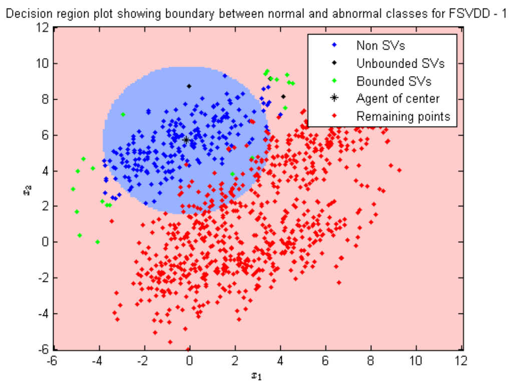


(b) F-SVDD

Figure 4: Decision region plot with class-3 points as target class



(a) C-SVDD



(b) F-SVDD

Figure 5: Decision region plot with class-4 points as target class

Observations :

1. We observe that the performance of F-SVDD (both 1 and 2) is slightly inferior in comparison to that of C-SVDD. C-SVDD outperforms F-SVDD-1 by 3.5 % on an average. This is because of the nature of the decision surfaces obtained using F-SVDD i.e spherical in the x-space (will be explained later).
2. FSVDD-1 and FSVDD-2 perform very similarly in most of the cases and the difference in their average error rates is 0.0019   0.19 % which is negligible. This is because, the $\hat{x}$ found by Fixed point iteration was very close to the one found by direct method. As the dataset is quite simple, the effect of different starting points did not play a role here.
3. The total error rate for class 1 was observed to be more than that of the other classes, for both the algorithms. This is clearly because of the points of this class have more overlap with the other class points. The error rates observed for the other class points were very similar.
4. The agent of the center $\hat{x}$ found by F-SVDD is also marked in the decision region plot, denoted by a black cross. It is the center of the spherical decision surface that we obtain in the x-space.

5. The performance of neural network on the test and validation data is very poor, indicating poor generalizability.
6. SVDD based techniques perform much better than neural networks for one class problems. The main problems in nerual networks lie in the choice of the function to be used and the threshold of the output value to decide whether it has classified to the positive class or not.
7. We also see from the values that NN is overfitting to the training data which is not the case with SVDD.

The running times that were observed for each of the algorithms are shown below :

Table 2: Running time (in seconds) comparison for Overlapping classes dataset

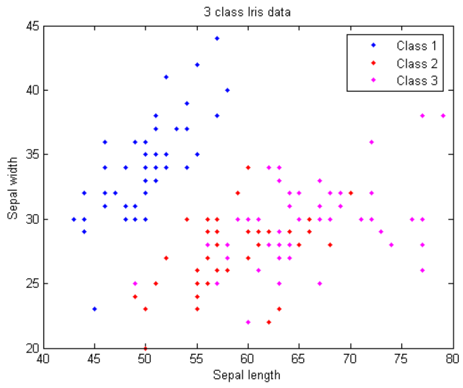
Target class	C-SVDD		F-SVDD-1		FSVDD-2	
	Training	Testing	Training	Testing	Training	Testing
1	0.3351	0.6312	0.3744	0.0187	1.0764	0.0156
2	0.3551	0.7176	0.3588	0.0187	1.1232	0.0094
3	0.3414	0.7722	0.3476	0.0171	1.4881	0.0094
4	0.3120	0.7753	0.3744	0.0187	1.4661	0.0094

Observations :

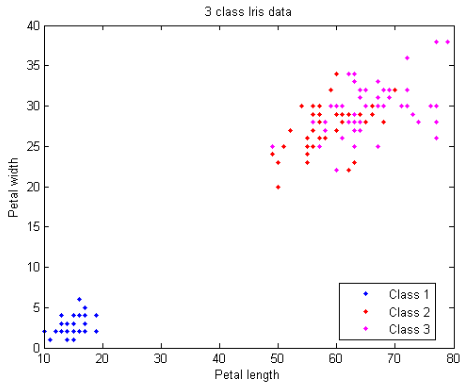
1. The training time for F-SVDD-1 was found be higher than that of C-SVDD, which is because of the additional computation that is required for the computation of the agent of the center $\hat{x}$. But, we note that the difference in time taken is very less on an average.
2. On the contrary, the F-SVDD-2 takes abound 1.28 seconds on an average for training. This is much higher than the other two algorithms and has a lot more variation in the times for each of the classes. The excess time is due to the fact that the fixed point method requires a lot of iterations to converge.
3. The testing time for C-SVDD is found to be quite high due to the N_s computations for the predicting of an unseen data point. In comparison, we observe that the testing time for F-SVDD-1 and 2 are negligible, as they involve only 1 computation for a new point. This shows the tremendous increase in classification speed in the F-SVDD algorithms.

2.2 Iris Dataset :

[2] This is a well known pattern recognition data set containing 3 classes of 50 instances each. This dataset is 4 dimensional with one linearly separable class while the other 2 overlap. Class 1 can be separated well from the other 2 classes. The dataset has 150 points in total, containing 50 points in each class. The scatter plot of the dataset is shown below illustrates the separability when projected on the petal dimensions.



(a) Scatter plot w.r.t Sepal width and length



(b) Scatter plot w.r.t Petal width and length

Figure 6: Scatter plot of the Fisher Iris dataset

For the task of Novelty detection , we performed 3 sets of experiments considering each of the three classes as the target class and the rest of the points as outliers. For the purpose of discussion, let us choose class 1 as the target class. We choose 25 points of class 1 to train, and divide the rest into validation and test sets. The best values of C and σ obtained were $C = 0.4$ and $\sigma = 22.3$.

Based on the featurewise plots from the previous page, it is very apparent that the first class is quite distinct from the other two classes in every feature value. It is easy to separate class 1 from the rest. The more crucial thing here is to separate classes 2 and 3.

The performance of the algorithms on this dataset are shown below :

Table 3: Comparison of Average Error rates on Fisher dataset

Target class	C-SVDD	F-SVDD-1	F-SVDD-2	MLFFNN
1 ($N_{train} = 25$)	0.0224	0.0275	0.0296	0.0105
2 ($N_{train} = 25$)	0.0632	0.0780	0.0864	0.1492
3 ($N_{train} = 25$)	0.0616	0.0884	0.0832	0.0800
Average TAER	0.0491	0.0646	0.0664	0.0799

Observations :

1. The error rates for class 2 and 3 are higher than that of class 1. This is because points of class 1 are separable (from others) while classes 2 and 3 overlap, which can seen very clearly in Figure 6. The reason for this as pointed out earlier is that the F-SVDD algorithm forces a hyperspherical decision boundary for the classes in the original space. This performs poorly if classes are intermingled.

[2]https://archive.ics.uci.edu/ml/datasets/Iris

2. The error rates for F-SVDD-1 follow that of C-SVDD closely, with the average TAER begin only 1.5 % more than C-SVDD. This shows a good generalization performance for F-SVDD.
3. The error rates for F-SVDD-2 are only slightly inferior to that of F-SVDD-1, which shows that the fixed point method almost always converges to the optimal $\hat{x}$ correctly.
4. ANN performs very well for class 1 which is quite separable. It also performs decently for class 3 and very poorly for class 2. This is due to the threshold value being attained by some overlapping points of the adjacent classes also. What we observed from the confusion matrix was that a large number of class 3 points also satisfied the threshold for class 2.

The observed running times on this dataset for the different algorithms are shown below :

Table 4: Running time comparison for Fisher Iris dataset

Target class	C-SVDD		F-SVDD-1		FSVDD-2	
	Training	Testing	Training	Testing	Training	Testing
1 ($N_{test} = 125$)	0.02028	0.0109	0.02496	0.0031	0.0468	0.0015
2 ($N_{test} = 125$)	0.02184	0.00936	0.0281	0.0015	0.0577	0.0015
3 ($N_{test} = 125$)	0.02808	0.01248	0.0218	0.0031	0.0748	0.0015

Observations :

1. As in the earlier case, we observe that the training time for F-SVDD-1 is slightly higher than that of C-SVDD because of the computation of the preimage $\hat{x}$. The training time for F-SVDD-2 however, is much higher owing to the large number of iterations required for the convergence of the fixed point method.
2. Testing times follow the same trends, with F-SVDD-1 and F-SVDD-2 taking negligible time for classification, in comparison to C-SVDD.

2.3 Wine recognition Dataset :

[3] - This is a 13 dimensional dataset of chemical constituents, whose classes are not linearly separable. Among the 13 features, we found that 2 attributes - number 5 (Magnesium) and number 13 (Proline) had totally different ranges compared to the other attributes. To avoid problems due to this, we normalized the entire dataset using min-max normalization.

Due to this, for this class alone, our error rates are even lower than the ones presented in the paper. Not normalizing this dataset would cause the highly varying ranges of the different attributes to impact the result negatively.

This dataset has 178 points, with 59 instances of class 1, 71 instances of class 2 and 48 instances of class 3. As in the previous cases, we perform 3 experiments considering each of the 3 classes as the target class.

Table 5: Comparison of Average Error rates on Wine Recognition dataset

Target class	C-SVDD	F-SVDD-1	F-SVDD-2	MLFFNN
1 ($N_{train} = 29$)	0.0464	0.0554	0.0632	0.0268
2 ($N_{train} = 35$)	0.2052	0.2426	0.2592	0.2517
3 ($N_{train} = 24$)	0.0424	0.0482	0.0548	0.2013
Average TAER	0.0980	0.1154	0.1257	0.1599

Observations :

1. In this dataset, the points of class 2 overlap a lot with the other two classes. This clearly noticable from the results as well.
2. Both F-SVDD techniques perform very poorly for class 2. They are forcing a hyperspherical surface as opposed to the flatter decision surface of C-SVDD and hence provide relatively lower accuracy, given that unevenly overlapping data is poorly separated by a hypersphere.
3. F-SVDD-2 performs significantly poorer than F-SVDD-1 presumably because the overlap causes the fixed point iterations to fail to converge in a fixed number of iterations (max of 100 in our case).
4. ANN performs very well here for the first class. This is similar to the observation in the above datasets as well. Overlap causes many points of adjacent classes also to fall into the threshold value, for the NN output value causing the poor accuracy.

Table 6: Running time (in seconds) comparison for Wine recognition dataset

Target class	C-SVDD		F-SVDD-1		FSVDD-2	
	Training	Testing	Training	Testing	Training	Testing
1 ($N_{test} = 149$)	0.0245	0.0172	0.0254	0.0043	0.03744	0.0015
2 ($N_{test} = 143$)	0.02496	0.0156	0.0281	0.0031	0.0390	0.0031
3 ($N_{test} = 154$)	0.0234	0.01404	0.0249	0.0046	0.0374	0.0046

Observations :

1. With regard to running times the trend is very similar to what was observed for the previous two datasets. Training times are fastest for C-SVDD followed by F-SVDD-1 followed by F-SVDD-2.
2. The decrease factor in testing times here for F-SVDD comprated to C-SVDD is even more than the previous cases because computing the Gaussian kernel function is more expensive when the number of dimensions of the data go up.

[3]https://archive.ics.uci.edu/ml/datasets/Wine

3 Comparison and inferences

In this section, we explain the difference in decision regions obtained by C-SVDD and F-SVDD. In the conventional SVDD algorithm, the discriminant function $g(x)$ is given by :

$$g(x) = c - 2\sum_{i=1}^{N_s} \alpha_i K(x, x_i)$$

.

So, the decision surface which is given by $g(x) = 0$ is :

$$\sum_{i=1}^{N_s} \alpha_i K(x, x_i) = c/2$$

This leads to a non-linear decision surface in the x-space when a gaussian kernel is used.

For F-SVDD, the discriminant function is given by 6 and hence the decision boundary is :

$$K(x, \hat{x}) = \frac{\gamma c'}{2}$$

If we use a gaussian kernel this reduces to a very simple form as shown below :

$$||x - \hat{x}||^2 = 2\sigma^2(1 - \log \gamma c') = c''$$

The equation above represents a circle in the x-space centered at $\hat{x}$. This is the reason why spherical boundaries are observed in the x-space when a gaussian kernel is used. This proves to a limitation in certain cases, where the points of the target class cannot be compactly covered using a spherical decision surface in the x-space. A simple example of such a scenario is shown below :

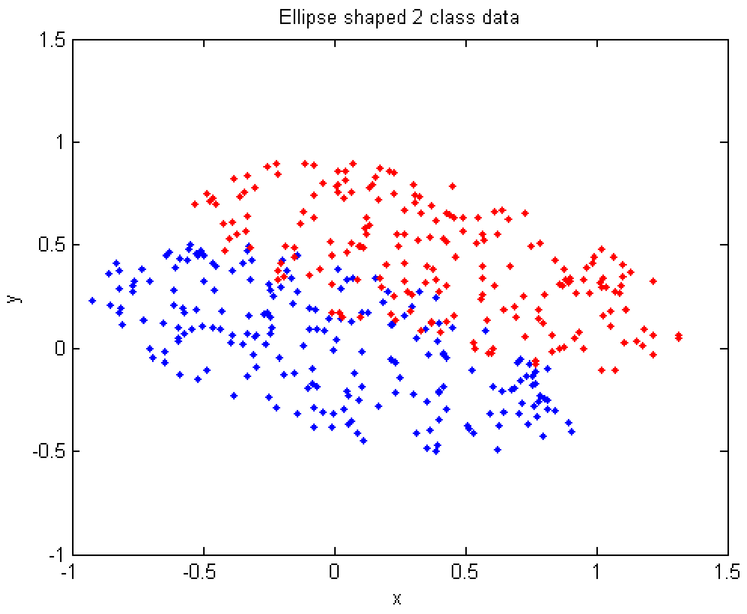
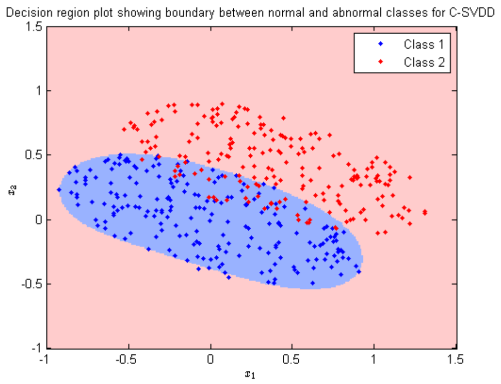
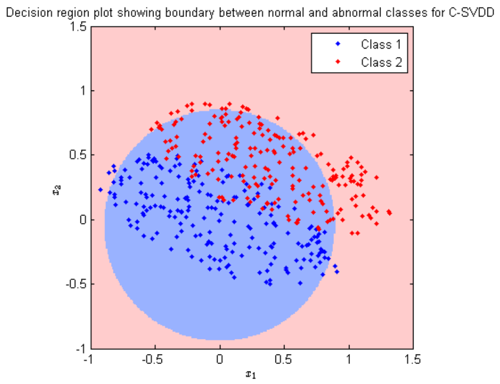


Figure 7: 2-Ellipse classes dataset with 2 classes

The above figure shows a synthetically generated dataset (2-Ellipses) with 2 intersecting ellipses in 2 dimensions. Clearly, the usage of a spherical decision surface will not be adequate to the separate the points of the 2 classes. To verify this, we plot the decision surfaces obtained on training C-SVDD and F-SVDD on this 2-Ellipses dataset as shown below :



(a) C-SVDD



(b) F-SVDD

Figure 8: Decision region plots obtained on the 2-ellipse dataset

As expected, the spherical decision surface constructed by F-SVDD is inadequate to separate the 2 classes but on the contrary, C-SVDD constructs a very good decision elliptical decision surface. This is one of the limitations of F-SVDD in spite it's superior classification speed. But, this problem could be solved by using a different normalized kernel function instead of gaussian. But as we do not know about the data distribution before hand, the task becomes quite challenging.

4 Conclusion

From the experimental results obtained, we observe that F-SVDD proves to be very efficient algorithm for the task of novelty detection. In comparision to C-SVDD, the generalization performance dips slightly, but there is an overwhelming increase in the classification speed. The testing time complexity of F-SVDD is $O(1)$ which makes suitable for real-time applications where speed is very important. In real-time systems, accuracy do not matter to a great extent, thus making F-SVDD very useful in such scenarios.

References

[1] Tax, David MJ, and Robert PW Duin. "Support vector data description." Machine learning 54.1 (2004): 45-66.

[2] Liu, Yi-Hung, Yan-Chen Liu, and Yen-Jen Chen. "Fast support vector data descriptions for novelty detection." Neural Networks, IEEE Transactions on 21.8 (2010): 1296-1313.

[3] Hornik, Kurt, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators." Neural networks 2.5 (1989): 359-366.