Data Generation Using Geometrical Edge Probability for One-class Support Vector Machines

Pilwon Woo **Kichun Lee**[*]
Department of Industrial Engineering
Hanyang University
{wpw0501, skylee}@hanyang.ac.kr

Abstract

In the tasks of data mining and machine learning with unlabeled data, one-class support vector machines (OCSVMs) are one of the most widely used methods for anomaly detection as one class classification. However, the choice of hyperparameters, a critical step for learning effective OCSVM decision boundaries, remains open as a post analysis step and often undecided. To tackle this issue, this paper proposes a new data-generation method using geometrical edge probability suitable for OCSVM hyperparameter selection. It improves the limitations of existing methods in that the geometrical edge probability enables the generation of both pseudo target data and pseudo anomaly data. In addition, the method, combined with bootstrapping, is able to produce the distribution of anomaly data. We evaluate the proposed method for datasets with 16 different dimensions and show the performance improvement in the classification of target and anomaly data.

1 Introduction

One-class classification (OCC), also called anomaly (novelty) detection, deals with a problem in which only normal data without label information exist. In most cases, the procedure of obtaining label information associated with data incurs high cost. For an example, the cost of identifying people diagnosed with Parkinson's disease is far greater than that of finding people who are not. However, one-class data, unlabeled inherently, are not exempt from the existence of outliers due to variation and noise in data formation. In such cases, a model describing the distribution of one-class training data is necessary because two-class classification models are inappropriate. Among possible models handing one-class data, a one-class support vector machine (OCSVM) learns a decision boundary by finding support vectors in a feature space from training data (target data). Undoubtedly, OCSVMs have already been applied on numerous domains for outlier detection, equipped with a decision function to evaluate whether or not a future observation is an outlier. The application of kernel function, usually set by Gaussian, for the formation of a feature space, enables OCSVMs to learn flexible nonlinear decision boundaries. However, the learning of decision boundaries through OCSVM depends entirely on the choice of two hyperparameters: firstly, the regularization coefficient, denoted by ν and secondly the kernel parameter, denoted by σ. The regularization coefficient, ν, represents the upper limit of a training data to be deemed outside the decision boundary Schölkopf et al. (2001). The other, σ, represents the Gaussian kernel bandwidth which affects the locality degree of the decision boundary.

Figure 1 shows the effect of hyperparameters on OCSVM decision boundaries. If the value of ν is too large (**d** in Figure 1), the model tends to exclude too much data from the decision boundary, resulting in poor outlier detection performance. On the other hand, excessively large sigma (**a** in Figure 1) distorts the decision boundary from the training data, causing overfitting. On the other

[*]Corresponding author

hand, excessively small sigma (**c** in Figure 1) results in underfitting by producing too smooth decision boundaries to detect outliers.

Indeed, minimizing validation error is impractical in hyperparameter selection since no explicit outlier label is available by the nature of one class data. In this paper, we propose a new data-generation method for one class data using geometrical edge probability, suitable for OCSVM hyperparameter selection. The concept of geometrical edge probability takes advantage of the geometric and distributional features of training data. The proposed method generates pseudo samples, either normal or abnormal, in a stochastic manner. Thus, using the generated samples with labels of normal or abnormal, we are able to build a decision boundary to prevent overfitting while preserving the distribution of target data. In addition, we can also empirically generate outlier distributions in the given dataset by randomly selecting samples in the dataset.

The rest of this paper is organized as follows. Section 2 reviews the concept of sample generation in OCC problems, theoretical backgrounds of OCSVM, and the existing pseudo sample generation methods for OCSVM. In Section 3 we proposes a pseudo sample generation method. The process of hyperparameter selection is discussed in detail. In Section 4 we first provide visual analysis on 2-D datasets with various shapes, then compares the proposed methodology with other hyperparameter selection methods through various dimensions of data. Section 5 concludes with implications and future research directions.

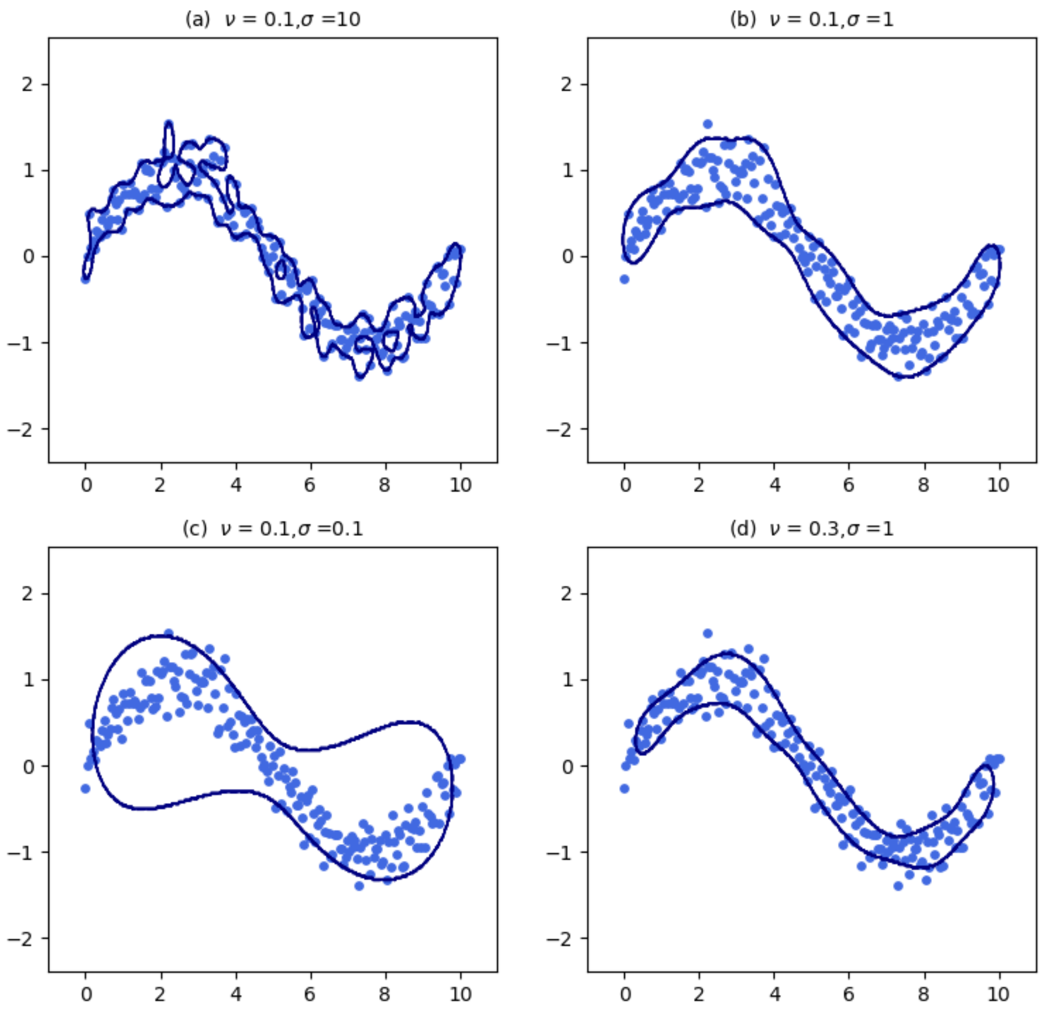

Figure 1: Hyperparameter effect on OCSVM decision boundaries

2 Related work

2.1 Sample generation in one-class classification

Sampling often improves the difficulty of learning that occurs when the label ratio of sample data is too different. Undersampling reduces the weight of class labels with a number of instances, and oversampling increases the weight of a class with a few samples. Numerous sampling methods have been proposed. Oversampling is divided into two types, Random oversampling and informative oversampling(Sonak & Patankar (2015)). Informative oversampling method synthetically generates minority class examples based on a pre-specified criterion (Ramyachitra & Manikandan (2014)) while random oversampling, simple but effective, does not depend on any criterion. In random sampling, we choose instances from the minority class at random with replacement, then duplicated and added them to a new training set. Oversampling methods exist in the literature such as SMOTE(Chawla et al. (2002)), borderline SMOTE(Han et al. (2005)), safe-level SMOTE(Bunkhumpornpat et al. (2009)) etc. SMOTE, one of the most well-known methods in informative oversampling, creates *synthetic* example rather than oversampling with replacement, on the line connecting the minority samples to their k-nearest minority class neighbors. Undersampling is also divided into two types. Random undersampling (RUS), a simple and classical method, randomly eliminates samples from the majority class till the data set gets balanced. However, potentially useful information in the discarded samples is lost. Researchers have made numerous attempts to improve the performance of RUS such as Tomek-Link (Tomek et al. (1976)), condensed nearest neighbor rule (Hart (1968)), one-sided selection (Kubat et al. (1997)) etc. Tomek-Link is considered as an enhancement of a nearest-neighbor rule, and one-sided selection (OSS) attempts to intelligently under-sample the majority class by removing majority class instances that are considered either redundant or noisy (Ganganwar (2012)).

However, the one-class classification problem starts when no label information is available: labeling for neither outliers nor normal observations exits. The absence of label information makes it difficult to apply previously suggested sampling methods based on an existing sample distribution. Therefore, in this paper we propose a sample generation method focusing on one-class classification problems by introducing a probabilistic measure to determine whether or not each instance is an outlier. We apply the proposed sampling method to OCSVM parameter selection.

2.2 One-class support vector machines

Extending binary support vector machines, the one-class support vector machine (OCSVM) model was proposed for unlabeled data by one class classification (Schölkopf et al., 2001). Given a training set of $\{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_N}\} \in \mathbf{X}$ and a feature-mapping function, $\phi(\cdot)$, the model forms a decision boundary describing the training set in input space X by finding a hyperplane, $\mathbf{w}^\intercal\phi(\mathbf{x}) - \rho = 0$, in the feature space. It finds the hyperplane in such a way that most of the training data in one region are separated by the hyperplane. This can be formulated into the following primal problem.

$$\begin{aligned} \min_{\mathbf{w},\xi,\rho} \quad & \frac{1}{2}\|\mathbf{w}\| + \frac{1}{\nu N}\sum_{i=1}^{N}\xi_i - \rho \\ \text{s.t.} \quad & \mathbf{w}^\intercal\phi(\mathbf{x_i}) - \rho + \xi_i \geq 0, \quad \xi_i \geq 0, \quad \forall i, \end{aligned}$$

in which ξ_i is a slack variable to allow soft-margin and ν, the regularization coefficient, is the maximum ratio of the training data that can be excluded from the decision boundary. When ν is small, the hyperplane puts more data above the hyperplane, leading to less outliers. As mentioned in the introduction, parameter ν plays a key role in the hyperplane formation. Through Lagrangian multipliers, we can solve the primal problem by the following dual problem.

$$\begin{aligned} \max_{\alpha} \quad & -\frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j K(\mathbf{x_i}, \mathbf{x_j}) \\ \text{s.t.} \quad & \sum_{i=1}^{N}\alpha_i = 1, \quad 0 \leq \alpha_i \leq \frac{1}{\nu N}, \quad \forall i, \end{aligned}$$

in which $K(\cdot,\cdot)$ represents a kernel function, $K(\mathbf{x_i}, \mathbf{x_j}) = \phi(x_i)^T\phi(x_i)$, the inner product of the data mapping to the feature space through mapping function ϕ. Among several kernel functions, the following Gaussian kernel is the most commonly adopted and flexible as it puts the mapped feature in the infinite dimension (Schölkopf (2001)): $K(\mathbf{x_i}, \mathbf{x_j}) = exp(-\frac{\|\boldsymbol{x_i}-\boldsymbol{x_j}\|^2}{\boldsymbol{\sigma^2}})$. The hyperparameter σ in the kernel function is also critical in the hyperplane formation since it scales the distance between two original observations: for lager σ, the decision boundary becomes rough, prone to to be over-fitting. To select critical hyperparameters, ν and σ, in practice, most researches utilize the combination of grid search and cross validation, but the nonexistence of label in one-class data is an inherent obstacle to the approach. In the next section, we discuss data generation in one-class classification problems.

2.3 Sample generations for OCSVM

Indeed, Fan et al. first proposed a sampling approach for one-class classification problems, an algorithm to generate artificial anomalies and coerce an inductive learner into discovering an accurate boundary between known classes (normal connections and known intrusions) and anomalies. However, the approach is not applicable when a feature variable is continuous. Tax & Duin (2001) also suggested support vector data description, generating artificial outliers, uniformly distributed in a hypersphere. However, uniformly generated samples in the hypersphere inevitably explodes in computation as the dimension increases. It also has a problem of setting a proper location in the hypersphere for its outlier creation. An appropriate location is necessary because pseudo outliers too distant from the training data may be unhelpful in learning the boundary. Later Bánhalmi et al. (2007) extended a training dataset with a counter-example set directly generated by the set of positive examples. Using the extended training set, they formed a binary classifier (by ν-SVM) to separate the positive and the negative instances. In this process, however, hard-margin linear SVMs need to be trained for all training samples, incurring great computation, and two other hyperparameters in the model remained undecided. Deng & Xu (2007) proposed a skewness-based outlier generation defining error function with two evaluation measures, rejected target and accepted artificial outliers. The error function requires a critical hyperparameter, called α that sets the tradeoff between the two evaluation measures.

All the proposed methods so far deal only with the generation of pseudo outliers. However, pseudo outliers alone are unable to prevent the decision-boundary learning from overfitting in the training data. Wang et al. (2018) proposed self-adaptive data shifting (SDS) that generates both pseudo outliers and pseudo target samples (belonging to the majority class, denoted by pseudo target hereafter). The SDS method first detects an edge pattern located at the edge of the data and creates pseudo outliers using the edge patterns and pseudo targets using the other samples not in the patterns. Using the edge-pattern detection (EPD) algorithm, Wang et al. (2018) showed comprehensive experimental results outperforming various other hyperparameter-selection methods for OCSVMs. However, the method has limitation in pseudo outlier creation as it has difficulty in setting edge patterns in a deterministic manner and the points located near the outer corner of a concave shape cannot be easily detected as an edge pattern. Besides, it involves two hyperparameters of k for $k - nn$ and threshold T. Moreover, even points on the outside of distribution are only used to generate pseudo targets unless determined to be as edge pattern. Thus, the method hardly captures the difference between samples at the edge and those near the less obvious edges.

3 Proposed method

Our strategy to generate both pseudo outliers and pseudo targets is to prepare a validation set and then to choose a model that minimizes classification error for the validation set. Following the point, we propose a method in which the geometric location of a sample has a relatively different impact on error computation. Specifically, we aim to make **pseudo outliers** generated from samples closer to the edge in the training data distribution have a higher impact on error computation than the **pseudo outliers** generated from samples relatively less closer to the edge. Similarly, we aim to make **pseudo targets** generated from samples located at the center of the distribution have a higher impact on error computation than **pseudo targets** generated from samples relatively located far outside of the distribution. Our view is that samples geometrically located outside of the distribution are more likely to produce pseudo outliers and samples located inside are less likely to produce pseudo outliers; and that outer samples will have a greater impact on outlier classification error than relatively inner samples.

For this reason, we model the probability of a point being close to edges, denoted by edge probability P_{edge}, in consideration of the location as follows: samples close to the edge of training data distribution had higher P_{edge} values. Then we generate a **pseudo outlier** if P_{edge} is less than a random variable $U \sim unif(0, 1)$.

By doing this, samples located near the outside of the distribution will generate more **pseudo outliers** than the sample located inside the distribution, and result in a more impact on error computation. Likewise, the same procedure is applied to the generation of **pseudo targets** using $P_{edge} > U$.

3.1 Edge Probability

We model the $P_{edge}(\mathbf{x_i}) \in [0, 1],\ i = 1, \dots, N$, which produces a high value for samples located near the outer edges in the data distribution using two geometric features of each sample. P_{edge} is later used for pseudo sample generation. First, we form a sphere in the input space $\mathbf{X}$ with a fixed radius with $\mathbf{x_i}$ as center. Then we calculate two geometric features of $\mathbf{x_i}$: the number of neighbors in the sphere, denoted by f_1, and the distance between the center of neighbors and $\mathbf{x_i}$, denoted by f_2.

Samples located near the outer edge of the distribution will contain fewer neighbors inside the sphere. In other words, samples near the edge have small f_1. The closer to the outer edge $\mathbf{x_i}$ becomes, the more neighbors inside the sphere will be skewed to one side. If $\mathbf{x_i}$ is inside the data distribution, far away from the outer edge, then the center of the neighbors will be quite much closer to $\mathbf{x_i}$. In other words, samples near the edge have large f_2. Since the range of the values of f_1 and f_2 is different, the two features are scaled between 0 and 1 in advance. Thus, we model P_{edge} proportional to $f_2 - f_1$ as follows:

$$P_{edge}(x_i) = \frac{(f_{2,i} - f_{1,i}) - min_j(f_{2,j} - f_{1,j})}{max_j(f_{2,j} - f_{1,j}) - min_j(f_{2,j} - f_{1,j})} \in [0, 1], \quad i = 1, \dots, N, \tag{1}$$

where N is the number of samples in training data. Once P_{edge} is set for a given sample $\mathbf{x}_i$, Prob$[U < P_{edge}] = P_{edge}$, where $U \sim unif(0, 1)$. A natural difference occurs in the number of neighbors in the sphere depending on the position of $\mathbf{x}_i$, so we set the sphere radius as the average of $r_1, r_2, \cdots, r_N$ where r_i represents the radius around x_i to cover $5 \ln N$ neighbors, following the routine in Li & Maguire (2010). In fact, the edge probability P_{edge} depends on f_2 due to the way to set the radius. Figure 2(**a**) shows assigned P_{edge} for each sample with gradation. The visualization confirms that the data located outside the distribution have a large value of P_{edge} as we intended. And We wanted to make the smaller value smaller and the larger value smaller according to the intention of modeling $P_{edge}(\mathbf{x_i}) \in [0, 1],\ i = 1, \dots, N$. Therefore we took the square root of $P_{edge}(\mathbf{x_i})$ greater than 0.5, which is the median of $U \sim unif(0, 1)$. And $P_{edge}(\mathbf{x_i})$ with a value lower than 0.5 is squared.

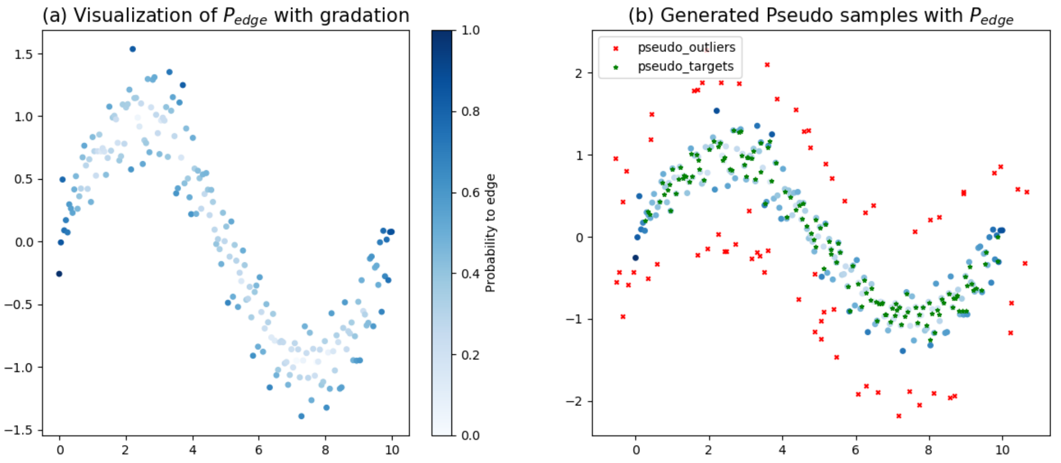

Figure 2: Visualization of P_{edge} and generated pseudo samples

3.2 Pseudo sample generation

In the following section, we discuss how to generate pseudo targets and pseudo outliers randomly with a proper direction and a magnitude.

3.2.1 Generating pseudo samples

After computing P_{edge} for each sample in the training data, we traverse each sample $\mathbf{x_i}$ and compare $P_{edge}(\mathbf{x_i})$ with a random number $U \sim unif(0,1)$.

If U is greater than $P_{edge}(\mathbf{x_i})$, **pseudo-outlier** is generated from $\mathbf{x_i}$. Otherwise, **pseudo-target** is generated from $\mathbf{x_i}$. We repeat the generation at $\mathbf{x_i}$ k times, and multiple pseudo samples are generated from $\mathbf{x_i}$. By repeating this process, samples with higher P_{edge} are likely to generate more pseudo outliers while samples with lower P_{edge} are likely to generate more pseudo targets. Noticeably, all samples have the possibility of generating pseudo outliers or pseudo targets. Samples with large P_{edge} tend to generate more pseudo outliers as a result of k generations than samples with smaller P_{edge}. Thus, a sample with P_{edge} close to 1 will generate many (almost k) pseudo outliers at one site through the generations, and if the decision boundary includes that site, compared to site with fewer pseudo outliers (by samples with relatively low P_{edge}), it will cause large validation error. Therefore, one can push the decision boundary to exclude the surroundings of the samples with large P_{edge}.

3.2.2 Direction and magnitude in pseudo sample generation

By following the idea of Fukunaga (1990), we approximate the gradient of the data density function of neighbor points inside the sphere and then create a **pseudo-target** in the direction of the gradient and a **pseudo-outlier** in the opposite direction. First we need the local region (sphere) $\Gamma(\mathbf{x})$ with radius $\mathbf{r}$ around sample $\mathbf{x}$.

$$\Gamma(\mathbf{x}) = \{\mathbf{y} \mid \|\mathbf{x}-\mathbf{y}\|^2 \le \mathbf{r^2}\} \tag{2}$$

Let data density at $\mathbf{y}$ be $\mathbf{p}(\mathbf{y})$, so the total mass of $\Gamma(\mathbf{x})$ is

$$\mathbf{u_0} = \int_{\Gamma(\mathbf{x})} \mathbf{p}(\mathbf{y})d\mathbf{y} \cong \mathbf{p}(\mathbf{x}) \cdot \mathbf{v}, \tag{3}$$

where $\mathbf{v}$ is the volume of $\Gamma(\mathbf{x})$. The expectation of direction vector $(\mathbf{y}-\mathbf{x})$ in $\Gamma(\mathbf{x})$, $\mathbf{M}(\mathbf{x})$, is computed as

$$\mathbf{M}(\mathbf{x}) = \mathbb{E}[(\mathbf{y}-\mathbf{x})\big|\Gamma(\mathbf{x})] = \int_{\Gamma(\mathbf{x})} (\mathbf{y}-\mathbf{x}) \cdot \frac{\mathbf{p(y)}}{\mathbf{u_0}} d\mathbf{y}. \tag{4}$$

By Taylor expansion, we have $\mathbf{p}(\mathbf{y}) \cong \mathbf{p}(\mathbf{x}) + (\mathbf{y}-\mathbf{x})^\intercal \nabla \mathbf{p}(\mathbf{x})$, we rewrite M(x) as follows:

$$\mathbf{M}(\mathbf{x}) = \mathbb{E}[(\mathbf{y}-\mathbf{x})\big|\Gamma(\mathbf{x})] \cong \int_{\Gamma(\mathbf{x})} (\mathbf{y}-\mathbf{x})(\mathbf{y}-\mathbf{x})^\intercal \frac{1}{\mathbf{v}} d\mathbf{y} \frac{\nabla \mathbf{p}(\mathbf{x})}{\mathbf{p}(\mathbf{x})} \propto \frac{\nabla \mathbf{p}(\mathbf{x})}{\mathbf{p}(\mathbf{x})}.$$

Regarding $p(x)$ as a constant, we finally approximates $\nabla \mathbf{p}(\mathbf{x})$ to

$$\nabla \mathbf{p}(\mathbf{x}) \cong s \cdot \mathbf{M}(\mathbf{x}) \cong \frac{s}{n} \sum_{j=1}^{n} (\mathbf{x} - \mathbf{x_j}), \tag{5}$$

where $\cong$ s is a constant and n is number of neighbors in sphere in $\Gamma(\mathbf{x})$. Therefore, we can approximate the direction to which the density rises most rapidly with neighbor points inside the sphere we formed. Then, we can generate a pseudo outlier toward the direction of $-\nabla \mathbf{p}(\mathbf{x})$ with the magnitude of the sphere radius. However, if the uniform number test is performed multiple times times on the same sample x, the pseudo outlier is generated only at the same location.This is because $\nabla \mathbf{p}(\mathbf{x})$ is always estimated in the same direction with same neighbor points in sphere $\Gamma(\mathbf{x})$. Meanwhile,it is more desirable to cover the empty space outside the training data by generating pseudo outliers at slightly different locations. So, we induced density gradient $\nabla \mathbf{p}(\mathbf{x})$ to be estimated with a slightly different direction each time. Each time we create a pseudo outlier from $\mathbf{x}$, first we get neighbors inside the sphere $\Gamma(\mathbf{x})$. When the number of neighbors is n, we get temporary neighbor samples with number of n by allowing duplicates from the neighbors(This is the key to change direction slightly). By estimating $\nabla \mathbf{p}(\mathbf{x})$ with the temporary neighbor samples following equation

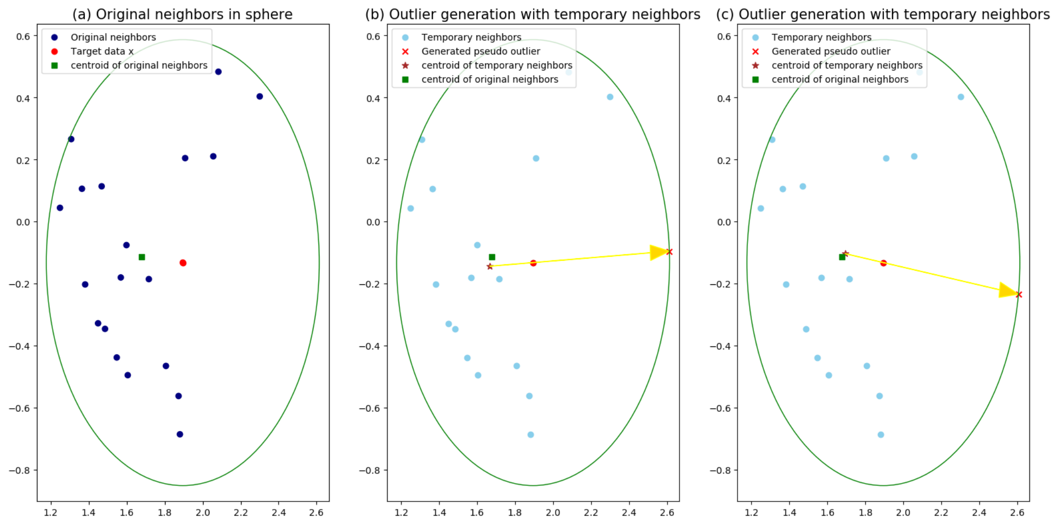

Figure 3: (a) is visualizations of sphere $\Gamma(\mathbf{x})$. From 'navy' neighbor points in (a), we get temporary neighbor samples. They are 'skyblue' points in (b) and (c). Then we estimate $\nabla\mathbf{p}(\mathbf{x})$ with the temporary neighbor samples and generate pseudo outliers as mentioned above. 'Gold' arrow is the direction of $-\nabla\mathbf{p}(\mathbf{x})$ we approximated with temporary neighbors. As we see in the figure (b) and (c), we can generate pseudo outlier with slightly different directions by sampling temporary neighbors.

(5), a pseudo outlier can be generated in slightly different directions. We visualized this process we mentioned above in 2-dimension in Figure 3.

To set the distance of pseudo target, we first compute inner product $\langle \frac{\nabla\mathbf{p}(\mathbf{x})}{\|\mathbf{p}(\mathbf{x})\|}, \mathbf{y}-\mathbf{x}\rangle$ for each $\mathbf{y} \in \Gamma(\mathbf{x})$. Then we take the minimum of the inner products as the magnitude that a pseudo target is generated from $\mathbf{x}$. Generating a pseudo target with the distance in the direction $\nabla\mathbf{p}(\mathbf{x})$ to which the data density rises most rapidly, we drive it to stay close to the original data $\mathbf{x}$.

After the generation step, we remove the pseudo outliers generated inside an obvious target area to avoid unnecessary validation errors. To compute the obvious target area, we fist obtain $distance_{nearest}$, which is the average of the distance between $\mathbf{x}$ and its nearest neighbor for all training data $\{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\} \in \mathbf{X}$. Then we remove the pseudo outliers within the $distance_{nearest}$ for each training data in $\{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\} \in \mathbf{X}$. We illustrate pseudo-targets and pseudo-outliers generated through the above procedures in Figure 2(b). As shown in the figure, sample $\mathbf{x}$ closer to the edge of the distribution generates more pseudo outliers at the same location because of large $P_{edge}(\mathbf{x})$. We can learn decision boundaries that include an appropriate $\mathbf{x}$ distribution while excluding pseudo outliers and thus avoiding overfit to the training data.

3.3 Hyperparameters setting in OCSVM

After the pseudo sample generation, we set the range of hyperparameter (ν, σ) and then train OCSVMs with all possible combinations of (ν, σ). After calculating the classification error for the pseudo targets and pseudo outliers, we select the optimal hyperparameter combination with the lowest classification error, completing the optimal OCSVM learning. We summarize the whole procedure in Algorithm 1.

4 Experiments

In this section, we discuss two experimental results of our proposed method. First, we illustrate the performance by visualizing the results in 2D data. Next we report the experiment results on 16 different benchmark datasets with various dimensions. For the grid search, we formed the combinations of hyperparameter (ν, σ) by $\sigma \in [10^{-4}, 10^{-3}, \ldots, 10^{4}]$ and $\nu \in [0.001, 0.05, 0.1]$.

Algorithm 1 OCSVM hyperparameter selection.

```
1:  Input: Training dataset $\mathbf{X_{train}} = \{\mathbf{x_i}\}_{i=1,...,N}$
           Range of hyperparameter $(\nu, \sigma)$
           Number of uniform number test $K$
2:  Output: Optimal Hyperparameter combination $(\nu, \sigma)_{opt}$
3:  Calculate $P_{edge}$ for each samples in $\mathbf{X_{train}}$
4:  for all $\mathbf{x_i} \in \mathbf{X_{train}}$ do
5:      for $k = 1, \dots, K$ do
6:          Generate $U \sim unif(0, 1)$
7:          if $U > P_{edge}(\mathbf{x_i})$ then
8:              Generate pseudo-outlier from $\mathbf{x_i}$
9:          else
10:             Generate pseudo-target from $\mathbf{x_i}$
11:         end if
12:     end for
13: end for
14: Remove the pseudo-outliers generated inside obvious target area
15: Set $Error_{best} = \infty$
16: for Each hyperparameter combination $(\nu, \sigma)$ from $(\nu, \sigma)_{range}$ do
17:     Train OCSVM model through training data with hyperparameter $(\nu, \sigma)$
18:     Compute the classification error rate $error_o$ on pseudo-outliers
19:     Compute the classification error rate $error_t$ on pseudo-targets
20:     Compute the current $Error_{current} = (error_o + error_t)/2$
21:     if $Error_{best} > Error_{current}$ then
22:         $(\nu, \sigma)_{opt}$ = $(\nu, \sigma)$
23:     else
24:         go back to the beginning of this loop
25:     end if
26: return $(\nu, \sigma)_{opt}$
```

4.1 Visualization on 2-D datasets

In Figure 4, we observe that the pseudo outlier and pseudo target are generated at proper position and distance by the proposed method for 2-D data of various shapes. In the donut shape data, the pseudo outliers generated at the concave, ring-shaped, boundary form appropriate boundary. The results confirm that the proposed method is practically applicable to one-class data of various shapes with multiple clusters. Additionally, in Figure 5, we bootstrap the training data 10 times and then generate pseudo outliers from each of the 10 samples using our proposed method. This experiment intends to add randomness to the data generation. Next, we aggregated the 10-generated pseudo outliers and visualized regions that are considered as an outlier region.

4.2 Comparison on benchmark datasets

Recently, the method, called SDS, by Wang et al. (2018) demonstrated good performance surpassing other OCSVM hyperparameter selection methods. To show the improvement of the proposed method in comparison with the SDS method, we adopted the same environment on the same benchmark datasets. For grid search, we formed the combinations of hyperparameter (ν, σ) with the $\sigma \in [10^{-4}, 10^{-4}, \dots, 10^{4}]$ and $\nu \in [0.001, 0.05, 0.1]$. We adopted 16 datasets from the UCI Machine Learning Repository and the LIBSVM repository. And We used f1 score and matthews correlation coefficient(MCC) as evaluation criterion. We summarize the number and dimension of each data in Table 1. We followed the same performance measurement as in Wang et al. (2018) for consistent comparison. We normalized all feature values to $[-1, 1]$. When multiple classes in the benchmark datasets exist, the multiple class types is divided in half. Then we set the former half of class as target, and the latter half of class as outlier. After dividing the target class into a training set and a test set, we obtained hyperparameters for the training set using the proposed method as in Algorithm 1. Then we computed the $f1$ score through a test dataset combining target and outlier data. We randomnly divided the target class into a training set and a test set 10 times, and repeat the same process by changing the target class and the outlier class. Then, we computed an average of the

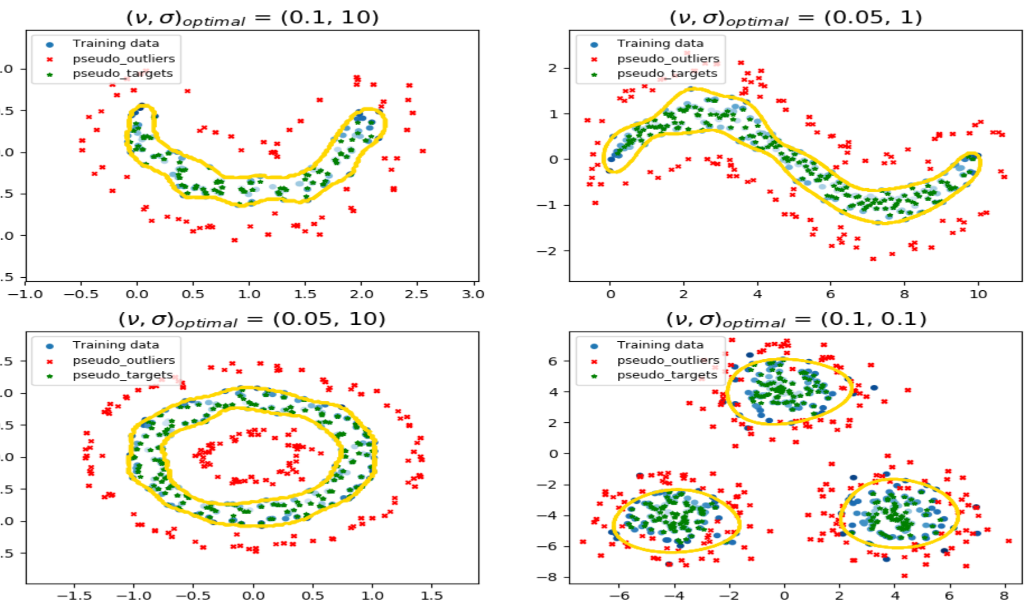

Figure 4: Experiments on 2-D datasets.

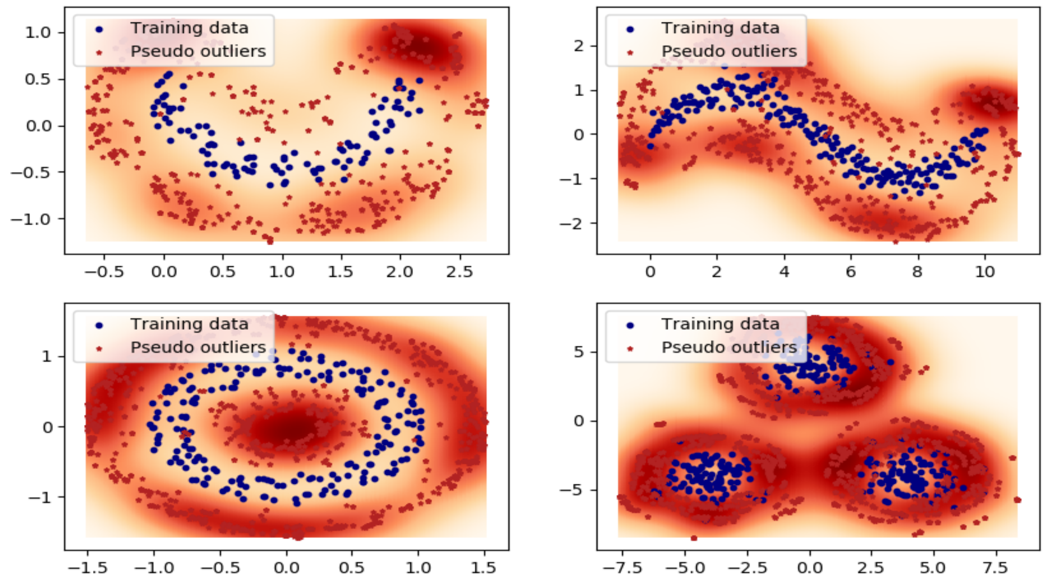

Figure 5: Visualizations of outlier regions

Table 1: Details of benchmark datasets

Dataset	**Feature dim.**	**# of data**
Abalone	8	4177
Australian	14	690
Balance	4	625
Diabetes	8	768
Glass	9	214
Heart	13	303
Landsat	36	2000
Letter	16	5000
Segment	18	2310
Sonar	60	208
SVMguide1	4	3089
Vehicle	18	846
Vowel	10	528
Vote	16	435
Waveform3	21	5000
Winequality	11	1599

Table 2: Average $f1$-score and MCC on benchmark datasets. p-value is in the bracket. $p < 0.05$ means significant difference from SDS.

Dataset	Criterion	**Proposed**	SDS	HC	HS	CS	SK	MSML	MIES	QR
Abalone	$f1$	**0.447(0.00)**	0.402	0.511	0.510	0.504	0.497	0.296	0.498	**0.512***
	MCC	0.127(0.01)	**0.171**	0.114	0.113	0.128	0.151	0.135	0.084	0.089
Australian	$f1$	**0.593(0.96)**	**0.592**	0.588	0.527	0.574	0.527	0.356	0.576	0.570
	MCC	0.310(0.03)	**0.335**	0.331	0.198	0.292	0.198	0.299	0.302	0.293
Balance	$f1$	**0.741(0.06)**	**0.808**	0.471	0.471	0.592	0.456	NaN	0.614	0.513
	MCC	0.520(0.00)	**0.739**	0.329	0.329	0.332	0.293	NaN	0.378	0.322
Diabetes	$f1$	0.474(0.00)	**0.508**	0.422	0.421	0.299	0.348	NaN	0.499	0.501
	MCC	**0.176(0.16)**	**0.168**	0.096	0.082	0.062	0.094	NaN	0.054	0.061
Glass	$f1$	**0.716(0.00)**	0.577	NaN	NaN	0.520	NaN	0.346	0.598	0.634
	MCC	**0.511(0.64)**	**0.495**	NaN	NaN	0.300	NaN	0.351	0.410	0.477
Heart	$f1$	**0.616(0.00)**	0.567	0.376	0.376	0.568	0.376	0.138	0.565	0.559
	MCC	**0.314(0.00)**	0.282	0.305	0.305	0.286	0.305	0.206	0.278	0.336
Landsat	$f1$	**0.817(0.00)**	0.711	0.665	0.645	0.713	0.695	NaN	0.658	0.636
	MCC	**0.639(0.00)**	0.611	0.471	0.457	0.547	0.523	NaN	0.467	0.442
Letter	$f1$	**0.698(0.00)**	0.601	0.519	0.519	0.515	0.068	0.058	0.514	0.494
	MCC	**0.498(0.00)**	0.349	0.145	0.145	0.123	0.079	0.140	0.120	0.138
Segment	$f1$	**0.775(0.89)**	**0.769**	0.589	0.588	0.576	0.431	0.436	0.581	0.589
	MCC	0.587(0.00)	**0.644**	0.348	0.346	0.309	0.441	0.444	0.325	0.348
Sonar	$f1$	**0.579(0.00)**	0.506	0.422	0.407	0.506	0.407	0.394	0.505	0.498
	MCC	**0.219(0.00)**	0.141	0.095	0.067	0.141	0.067	0.232	0.151	0.142
SVMguide1	$f1$	**0.845(0.67)**	**0.838**	0.720	0.662	0.727	0.741	0.217	0.755	0.610
	MCC	**0.713(0.05)**	**0.764**	0.568	0.460	0.590	0.634	0.277	0.622	0.345
Vehicle	$f1$	0.569(0.00)	**0.651**	0.564	0.501	0.497	0.442	NaN	0.505	0.523
	MCC	0.226(0.00)	**0.542**	0.276	0.055	0.032	0.033	NaN	0.078	0.129
Vowel	$f1$	**0.775(0.00)**	0.648	0.340	0.340	0.617	0.338	0.169	0.661	0.648
	MCC	**0.556(0.75)**	**0.552**	0.209	0.209	0.380	0.210	0.207	0.460	0.469
Vote	$f1$	**0.871(0.00)**	0.733	0.696	0.512	0.690	0.430	0.362	0.678	0.676
	MCC	**0.763(0.00)**	0.540	0.523	0.323	0.476	0.281	0.398	0.463	0.536
Waveform3	$f1$	0.649(0.00)	**0.674**	0.615	0.632	0.639	0.627	NaN	0.637	0.629
	MCC	0.381(0.00)	**0.455**	0.269	0.291	0.327	0.329	NaN	0.326	0.276
Winequality	$f1$	0.448(0.00)	**0.519**	0.500	0.500	0.497	0.226	0.241	0.501	0.501
	MCC	**0.149(0.50)**	**0.154**	0.053	0.046	0.044	0.182	0.181	0.037	0.039

$f1$-scores and MCCs computed in all procedures. Table 2 shows the result, and we mention that the performance results except for our proposed method are the same as Wang et al. (2018) reported. The other OCSVM hyperparameter selection methods in comparison are hyper-cube (HC, Tax & Duin (1999)), hyper-sphere (HS, Tax & Duin (2001)), consistency (CS, Tax & Muller (2004)), skewness (SK,Deng & Xu (2007)), Min#SV+MaxL(MSML, Khazai et al. (2011)), MIES(Wang et al. (2013)), and QMS+RDMMS (QR,Xiao et al. (2014)).

We performed two-sided T-test for the mean of one group to confirm the significance of the performance increase from SDS(Wang et al. (2018)). T-test was performed for both $f1$-score and MCC. All p-values recorded in Table 2 are T-test results for SDS values. Bold was taken for the higher values, but both were taken if not significantly different. However, in the case of Abalone dataset, the QR model showed higher performance than SDS and Proposed method, so it was bolded with *. As shown in Table 2, it can be seen that the performance of the proposed method shows significantly higher performance in most datasets for both $f1$-score and MCC. In particular, our method showed significantly higher performances in high-dimensional datasets such as Sonar(60) and Landsat(36). This can be attributed to the characteristics of proposed method, which can cover more empty area by generating more outliers at various locations compared to SDS.

5 Conclusions

This paper proposed a data generation method for optimal model selection in one-class support vector machines, by geometric features of the given dataset. We generated pseudo samples in a stochastic manner through an edge probability modeled to reflect two geometrical features of data samples. By doing so, we were able to assign each training sample to different impacts upon the decision boundary formation depending on their geometric location. Therefore, we tried to improve the limitation that only certain edge patterns affect the formation of decision boundaries by reliably determining edge patterns. By experimenting with various types of synthetic and benchmark datasets, we verified the effectiveness of the proposed method. We demonstrated its ability to infer outlier regions by adding randomness to the method. We also practically showed its performance in classification tasks. As future research, we envision the inclusion of other geometric features in the data generation procedure and their theoretical aspects.

References

András Bánhalmi, András Kocsor, and Róbert Busa-Fekete. Counter-example generation-based one-class classification. In *European Conference on Machine Learning*, pp. 543–550. Springer, 2007.

Chumphol Bunkhumpornpat, Krung Sinapiromsaran, and Chidchanok Lursinsap. Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In *Pacific-Asia conference on knowledge discovery and data mining*, pp. 475–482. Springer, 2009.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

Hongmei Deng and Roger Xu. Model selection for anomaly detection in wireless ad hoc networks. In *2007 IEEE Symposium on Computational Intelligence and Data Mining*, pp. 540–546. IEEE, 2007.

Wei Fan, M Miller, SJ Stolfo, Wenke Lee, and PK Chan. Using artificial anomalies to detect unknown and known network intrusions. In *Proceedings 2001 IEEE International Conference on Data Mining*, pp. 123–130. IEEE.

Keinosuke Fukunaga. Introduction to statistical pattern recognition. 1990.

Vaishali Ganganwar. An overview of classification algorithms for imbalanced datasets. *International Journal of Emerging Technology and Advanced Engineering*, 2(4):42–47, 2012.

Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing*, pp. 878–887. Springer, 2005.

Peter Hart. The condensed nearest neighbor rule (corresp.). *IEEE transactions on information theory*, 14(3):515–516, 1968.

Safa Khazai, Saeid Homayouni, Abdolreza Safari, and Barat Mojaradi. Anomaly detection in hyperspectral images based on an adaptive support vector method. *IEEE Geoscience and Remote Sensing Letters*, 8(4):646–650, 2011.

Miroslav Kubat, Stan Matwin, et al. Addressing the curse of imbalanced training sets: one-sided selection. In *Icml*, volume 97, pp. 179–186. Nashville, USA, 1997.

Yuhua Li and Liam Maguire. Selecting critical patterns based on local geometrical and statistical information. *IEEE transactions on pattern analysis and machine intelligence*, 33(6):1189–1201, 2010.

D Ramyachitra and P Manikandan. Imbalanced dataset classification and solutions: a review. *International Journal of Computing and Business Research (IJCBR)*, 5(4), 2014.

Bernhard Schölkopf. The kernel trick for distances. In *Advances in neural information processing systems*, pp. 301–307, 2001.

Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001.

Apurva Sonak and RA Patankar. A survey on methods to handle imbalance dataset. *Int. J. Comput. Sci. Mob. Comput*, 4(11):338–343, 2015.

David MJ Tax and Robert PW Duin. Data domain description using support vectors. In *ESANN*, volume 99, pp. 251–256, 1999.

David MJ Tax and Robert PW Duin. Uniform object generation for optimizing one-class classifiers. *Journal of machine learning research*, 2(Dec):155–173, 2001.

David MJ Tax and K-R Muller. A consistency-based model selection for one-class classification. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 3, pp. 363–366. IEEE, 2004.

Ivan Tomek et al. An experiment with the edited nearest-nieghbor rule. 1976.

Shijin Wang, Jianbo Yu, Edzel Lapira, and Jay Lee. A modified support vector data description based novelty detection approach for machinery components. *Applied Soft Computing*, 13(2):1193–1205, 2013.

Siqi Wang, Qiang Liu, En Zhu, Fatih Porikli, and Jianping Yin. Hyperparameter selection of one-class support vector machine by self-adaptive data shifting. *Pattern Recognition*, 74:198–211, 2018.

Yingchao Xiao, Huangang Wang, and Wenli Xu. Parameter selection of gaussian kernel for one-class svm. *IEEE transactions on cybernetics*, 45(5):941–953, 2014.