Kernel Choice in One-Class Support Vector Machines for Novelty Detection

Qinong Tian[1*], Peixi Liu[2], Tian Wu[3], Ziyue Chen[4]

[1]School of Mathematics and Statistics, Xi'an Jiao Tong University, Xi'an, 710049, China
[2]Faculty of Science and Technology, University of Macao, Macao, 999078, China
[3]Department of Statistics, Cornell College, Mount Vernon, lowa, 52314, USA
[4]Kings College Alicante, Alicante, 03016, Spain

Corresponding author. e-mail: tianqinong@stu.xjtu.edu.cn

Abstract. This work concentrates on novelty detection, a semi-supervised learning problem concerned with deciding if the new observation is sufficiently different from the ones seen so far. This paper mainly considers a variant of the support vector classification approach, which estimates the contours of the distribution of the initial observations and then can be used to decide if the new observations are abnormal. We try to estimate a negative function on the outlier points in the input space and a positive on the complement. A kernel expansion gives this decision function. The effectiveness of this kernel method is closely related to the choice of kernel functions and hyperparameters. Due to the demand for general and effective hyperparameter selection regulations, we investigate three approaches, including GridSearch in Python, median heuristic and Bayesian kernel learning. Some relevant experiments are performed in this paper. According to the experiments, we have learned that the choice of kernels and parameters can greatly influence the detection result.

Keywords: Unsupervised Learning, Novelty Detection, One-Class Support Vector Machines, Kernel Methods, Hyperparameter Selection.

1 Introduction

In practical applications, there are many situations that we already have a set of samples and we pay attention to whether a new observation is abnormal, i.e. whether the new observation is from a different distribution [1]. This problem is generally considered to be the novelty detection issue. Novelty detection is a semi-supervised learning.

There have been some attempts to solve the novelty detection problem, including methods of robust covariance, isolation forest and local outlier factor [2-4]. The original SVM algorithm using kernel methods has received great attention and is useful in many conditions, so it is reasonable to attempt to transfer this kernel method from supervised learning to semi-supervised learning of novelty detection problem. This paper will focus on an approach of One-class SVM which was derived from the original SVM algorithm. While the choice of kernels and hyperparameters are important in controlling the

V. Escudero et al. (Eds.): BADS 2022, AHCS 11, pp. 1200–1209, 2023.
https://doi.org/10.2991/978-94-6463-102-9_124

performance of our method, we have limited understanding of how to tune them. This paper concentrates on four kernel functions, especially RBF kernels. This paper also investigated three parameter selection approaches, including GridSearch in Python, Median heuristic and Bayesian kernel learning method.

In this work, the algorithm of the One-class SVM is investigated. Some experiments about kernel functions and parameter choice in kernels are conducted. First, we use different datasets of different distributions and features to compare the performance of four kernel functions. We aim to find which kernel function performs the best in most situations. Then the paper implements the median heuristic method and the GridSearch in Python to estimate the parameter choice in RBF kernels. Finally an experiment comparing different novelty detection approaches is displayed. The general content of this work is concluded and some future work is pointed out.

2 Algorithm

One-class SVM model is investigated in [5]. We consider the training data

$$\boldsymbol{x}_1, \ldots, \boldsymbol{x}_l \in \mathcal{X}, \tag{1}$$

where $l \in \mathbb{N}$ is the number of observations. Let Φbe a feature map $\mathcal{X} \to F$, which is a map into a dot space F. The dot product in the image of Φcan be computed by calculating kernels

$$k(\boldsymbol{x}, \boldsymbol{y}) = (\Phi(\boldsymbol{x}) \cdot \Phi(\boldsymbol{y})). \tag{2}$$

Our strategy is to map the data into the feature space corresponding to the kernel, and to separate them from the origin with maximum margin. We develop a function that takes the value +1 in a region capturing most of the data points and -1 on the complement. The negative or positive value of $f(\boldsymbol{x})$ can be used to determine whether a new point $\boldsymbol{x}$is an outlier or not. Different types of kernel function in this algorithm can lead to various estimators in input space.

To separate the data from the origin, we need to solve the following quadratic program which is similar to the process of original support vector machine:

$$\min_{\omega \in F, \xi \in \mathbb{R}^l, \rho \in \mathbb{R}} \frac{1}{2}\|\omega\|^2 + \frac{1}{\nu l}\Sigma_i \xi_i - \rho \tag{3}$$

subject to

$$(\omega \cdot \Phi(\boldsymbol{x}i)) \geq \rho - \xi i, \xi i \geq 0, \rho \neq 0. \tag{4}$$

Here, $\nu \in (0,1)$ is a parameter which determines upper bound on the fraction of outliers. Our decision function is given by

$$f(\boldsymbol{x}) = \operatorname{sgn}((\omega \cdot \Phi(\mathbf{x})) - \rho). \tag{5}$$

The trade-off between the penalization of slack variables and the regularization term $\|\omega\|^2$ is controlled by ν .

Using Lagrangian multipliers $\alpha_i, \beta_i \geq 0$ and set the relevant derivatives to zero, we obtain:

$$\omega = \sum_i \alpha_i \Phi(\boldsymbol{x}_i), \tag{6}$$

$$\alpha_i = \frac{1}{\nu l} - \beta_i \leq \frac{1}{\nu l}, \sum_i \alpha_i = 1. \tag{7}$$

According to the Lagrange multiplier and (2) we get the dual problem:

$$\min_{\alpha} \frac{1}{2} \sum_{ij} \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \quad \text{subject to} \quad 0 \leq \alpha_i \leq \frac{1}{\nu l}, \sum_{\alpha_i} = 1. \tag{8}$$

And the ρ in the decision function is given by:

$$\rho = (\omega \cdot \Phi(\boldsymbol{x}_i)) = \sum_j \alpha_j \, k(\boldsymbol{x}_j, \boldsymbol{x}_i). \tag{9}$$

As is shown above, the one-class SVM algorithm is similar to the original SVM and is a variant of it.

The following task is to solve the optimization problem of (8). The strategy is to break up task into the smallest optimization steps possible, and then we optimize over pairs of variables due to the constraint. We do the elementary optimizing step, which means that we fix the other variables and compute the formulas that the two variables satisfy to achieve the optimization goal. Then after initialization of ν and ρ , we do the overall optimization. We first select a variable and do one of two approaches of scanning KKT violators to consecutively recompute the offset ρ. An accuracy tolerance is often used in considering whether two quantities are equal.

3 Choice of kernel functions

The choice of kernel function can greatly influence the decision boundary and the novelty detection result. We consider four different kinds of kernel functions, including linear kernels, polynomial kernels, RBF kernels and sigmoid kernels. There are also new kinds of kernels such as Arc-cosine kernels [6]. In our experiments, we try some parameters in four kernels respectively and we discover that RBF kernels perform well among the four types of kernels. For this reason, the choice of kernel functions and inference of hyperparameters in this paper will generally be focused on RBF kernels. The functional form of RBF is:

$$k(\boldsymbol{x_1}, \boldsymbol{x_2}) = \exp\left(-\frac{\|\boldsymbol{x_1} - \boldsymbol{x_2}\|^2}{l^2}\right). \tag{10}$$

We will try different approaches to estimate the best hyperparameter l in the next section.

4 Methods of selecting hyperparameters in kernel functions

4.1 Median Heuristic

The median heuristic value for bandwidth parameter selection in RBF kernels is investigated [7,8]. Unlike in supervised kernel methods, a simple cross-validation approach for kernel parameter selection in unsupervised learning is not possible. When we are given a RBF kernel, one important step is to calculate the Gram matrix:

$$K = (k(\boldsymbol{x}_i, \boldsymbol{x}_j))_{1 \le i,j \le n}. \quad (11)$$

Gram matrix depends on the samples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_l$ and the hyperparameter l. It can be shown that when $l \to 0$ or $l \to +\infty$, the Gram matrix will lose relevant information in such extreme cases. So a reasonable approach to select the parameter is to compute:

$$l = \text{median}(\|x_i\text{-}\mathrm{x}_j\|_2), \quad (12)$$

which is considered to be in the "middle range". The median heuristic has been widely used in practice, but in some situations it does not lead to good performance.

4.2 Bayesian Learning of Kernel Embeddings

The model of Bayesian learning of kernel embeddings is investigated in [7]. It was designed primarily for unsupervised learning and is particular useful for kernel-based methods. It provides a method for learning bandwidth of RBF kernels. This approach consists in specifying a prior on the kernel mean embedding and a likelihood function linking it to the observations through the empirical estimator. Then we are allowed to infer the posterior distribution of kernel mean embedding. We can also derive a marginal likelihood function in which we can estimate the bandwidth in RBF kernels by finding the parameter that can maximize the marginal likelihood.

In some situations, the model of Bayesian learning of kernel embeddings has advantages over median heuristic. The Bayesian kernel learning can better reduce the type IIerror (i.e. probability that the test fails to reject the null hypothesis) than the median heuristic method. Besides, the Bayesian kernel learning can better reflect the distribution of the datasets when the data generating process is a mixture model.

4.3 GridSearch in Python

GridSearch is a method already available in Python. It is implemented by adjusting parameters in step length successively in a specific parameter range and finding the parameter that can have the best performance. It is a process of training and comparison. It is the method we mainly use in our experiments. However, this method costs a large amount of time and energy during the process of trying multiple different parameters and comparing them [9]. So a reasonable option is to choose a suitable range of parameters before the experiment.

5 Experiments

In the following four experiments, we utilize the sklearn.svm.OneClassSVM function to fit the models (Please refer to the Scikit-learn User Guide (Release 0.22.2) P1514-1515 for more information).

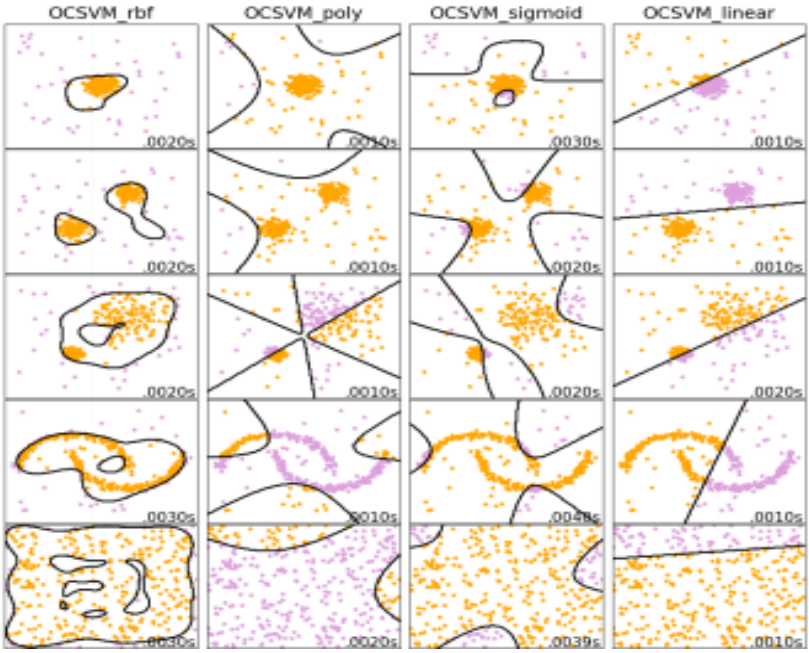

Fig. 1. Performance of four kernels on different datasets [Owner-drawn]

5.1 Rough Comparison of Four Kernels in Different datasets

In the first experiment, we use the Numpy package to generate four datasets with different shapes and distributions [10]. We set the four types of kernels with parameters of default values. We hope to provide a rough but intuitional comparison of how the kernels perform differently. We find that the RBF kernels perform the best among the four kernels. The other three kinds of kernels often lead to strange detection results or fail to reflect the characteristics of the data distribution (see Figure 1). For this reason, the following inference of parameters will focus on RBF kernels.

5.2 Performance of Median Heuristic

In the second experiment using median heuristic, the datasets are obtained from a real dataset of wine recognition and a function of Python that can randomly generate points in a specific distribution. In the first dataset, we calculate that the parameter l equals 1.016 by the median heuristic method. The decision boundary (darkred line) seems fairly good, since it draws a contours that well capture most of our points (see Figure

2). In the second dataset, we compute that the parameter l equals 5.254. The detection result turns out not to be good. There are two clusters in the dataset, but the decision boundary (darkred line) does not distinguish them. One possible explanation is that the median heuristic value lies in the range of distances of two points that are far from each other. So the median heuristic value is quite large in this situation and this value is not suitable for capturing points within a cluster. From the two examples, we can draw the conclusion that median heuristic can perform well sometimes, but in some situations, it is not a satisfying approach.

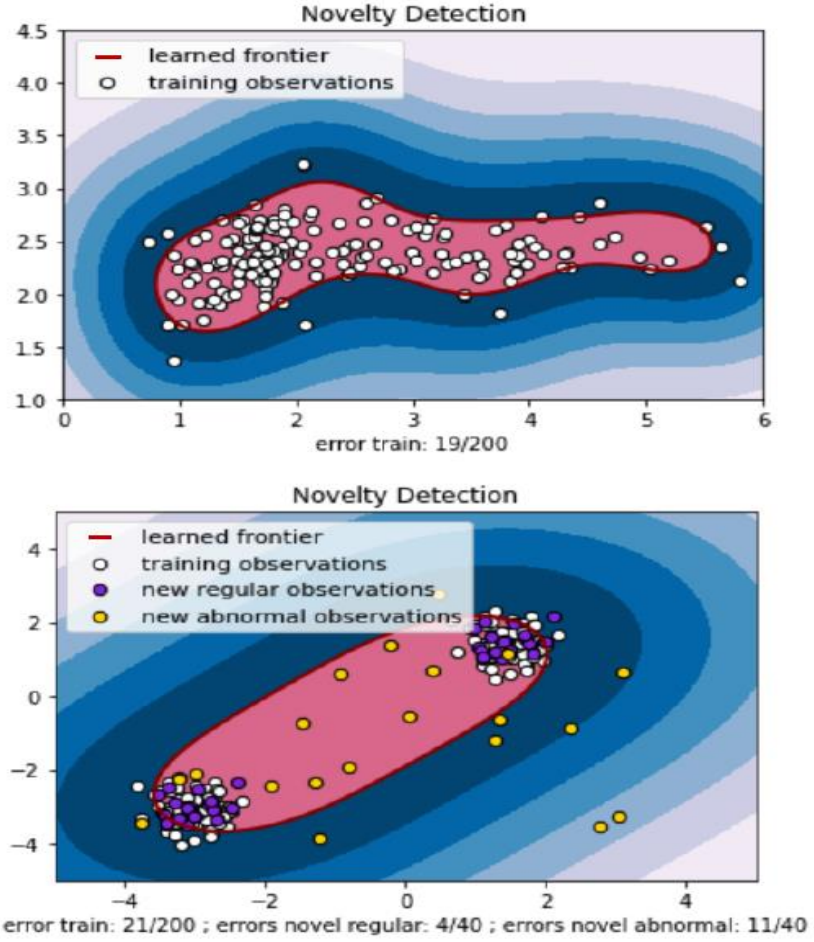

Fig. 2. Median heuristic on a real dataset and an artificial dataset [Owner-drawn]

5.3 Selecting the Parameter of RBF Kernels Using GridSearch

In the third experiment, we choose three kinds of datasets, including a small dataset with two clusters of the same covariance, a large dataset with two clusters of different covariance and a large data with two distant clusters of different covariance. We try to

use the GridSearch to estimate the best parameters for the RBF kernels in the three situations. The performance of RBF kernels using GridSearch is quite good (see Figure 3). We also calculate some statistical values including accuracy, percentage of false negative points, percentage of false positive points, precision, recall and F1-score to measure the performance of the kernels (see Table 1).

5.4 Comparison of Four Novelty Detection Approaches

In the fourth experiment, we do some further research. We have known that RBF kernels perform the best among the four kernels and we know how to select the parameters of RBF kernels. We want to compare the One-class SVM to other novelty detection approaches. The dataset is from the third one of the last subsection. The detection result and statistics show that One-class SVM performs the best among the four approaches (see Figure 4 and Table 2).

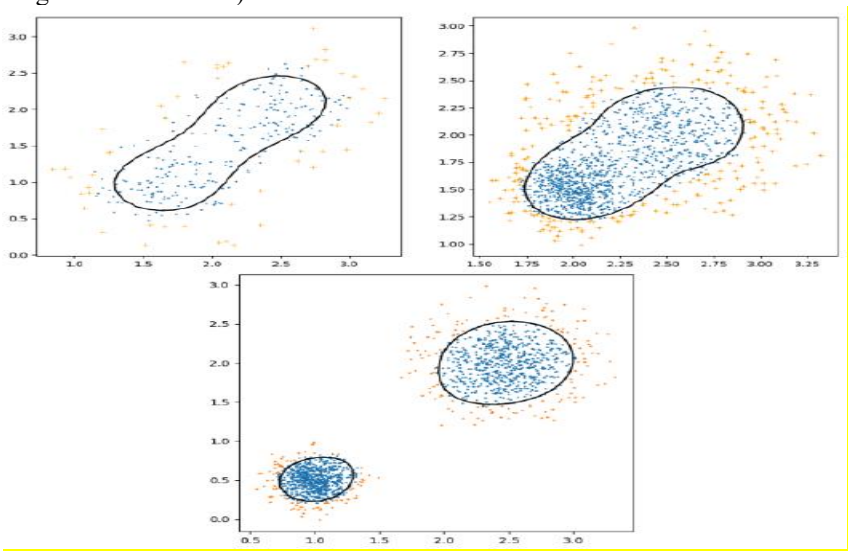

Fig. 3. Inference of parameters of RBF kernels in three datasets [Owner-drawn]

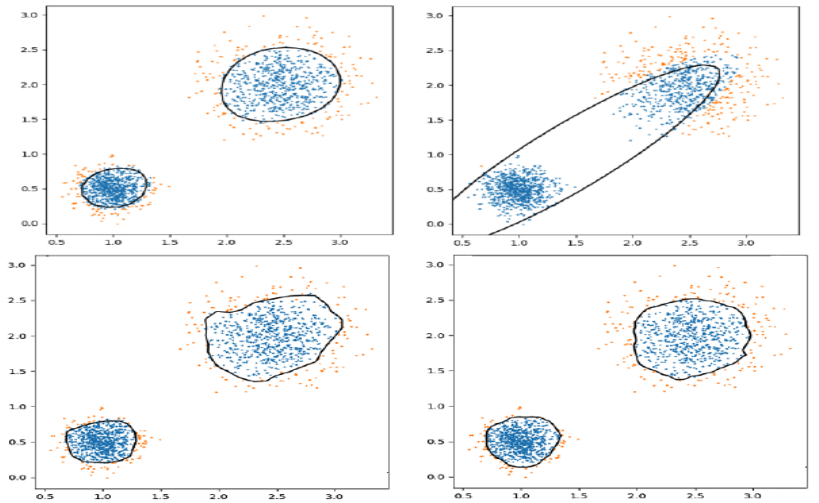

Fig. 4. Comparison of Four Classification Approaches [Owner-drawn]

Table 1. Selected parameters and relevant statistical values in three datasets [Owner-drawn]

Datasets	Parameters	Accuracy	False Positive	False Negative	Precision	Recall	F1-score
1	0.84	0.950	0.027	0.023	0.969	0.973	0.971
2	2.63	0.933	0.033	0.034	0.962	0.960	0.961
3	1.05	0.959	0.020	0.021	0.976	0.975	0.976

Table 2. Accuracy Performance of four classification approaches in one dataset [Owner-drawn]

Classification Approaches	Accuracy	False Positive	False Negative	Precision	Recall	F1-score
One-class SVM, Gamma=1.05	0.959	0.020	0.021	0.976	0.975	0.976
Elliptic Envelop	0.813	0.093	0.093	0.890	0.890	0.890
Local Outlier Factor	0.943	0.033	0.024	0.961	0.972	0.966
Isolation Forest	0.939	0.031	0.031	0.964	0.964	0.964

6 Conclusion

In this paper, we have learned about the algorithm of One-class SVM method. Then four choice of kernel functions and the selection of hyperparameters in the algorithm are discussed. Related experiments were also implemented, including comparing the performance of four kernel functions, using median heuristic to inference RBF kernel parameters, using GridSearch to find RBF kernel parameters and comparing four different classification approaches. We have been able to develop some understanding of how the kernels and hyperparameters can influence the performance of the algorithm. As for future work, the Bayesian kernel learning is supposed to be investigated further since we do not have deep understanding of this abstract model. Particularly we are interested in how much the Bayesian kernel learning is sensitive to the parameter τ. Besides, we wish to explore methods that can measure the performance of classification methods to a relatively accurate extent. It is very important in judging performance or comparing different choice of kernels and parameters, but in many situations, we can not fulfil these tasks in a fairly precise way. Some values of "precision" such as F1-score are calculated in the experiments, but we wonder if there are other satisfying methods.

References

1. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., and Duchersnay, E. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825-2830.
2. Haan, W. J. D. and Levin, A. T. (1996). A practitioner's guide to robust covariance matrix estimation. Technical report, National Bureau of Economic Research.
3. Liu, F. T., Ting, K. M., and Zhou, Z.-H. (2008). Isolation forest. In 2008 Eighth IEEE International Conference on Data Mining, pages 413-422. IEEE.
4. Breunig, M. M., Kriegel, H. P., Ng, R. T., and Sander, J. (2000). LOF: identifying density-based local outliers. In ACM sigmod record.
5. Bernard Schölkopf, John C. Platt, John Shawe-Taylor and Alex J. Smola. (2001). Estimating the support of a high-dimensional distribution. Neural computation, 13(7):1443-1471.
6. Cho Y. (2012). Kernel Methods for Deep Learning. [D]. University of California, San Diego.
7. Seth Flaxman, Dino Sejdinovic, John P. Cunningham, Sarah Filippi. (2016). Bayesian learning of kernel embeddings. arXiv preprint arXiv. 1603.02160.
8. Damien Garreau, Wittawat Jitkrittum, Motonobu Kanagawa. (2017). Large sample analysis of the median heuristic. arXiv preprint arXiv. 1707.07269.
9. Gretton, A., Sejdinovic, D., Strathmann, H., Balakrishnan, S., Pontil, M., Fukumizu, K., and Sriperumbudur, B. K. (2012). Optimal kernel choice for large-scale two-sample tests. In Advances in neural information processing systems, pages 1205-1213.
10. Developers, N. (2013). Numpy. NumPy Numpy. Scipy Developers, page 31.