Possibility of Using Overrule to Evaluate Overlap in Causal Inference

<What is the performance of Overrule in identifying overlap for different types of datasets?>

Shukun Cheng[1]

Supervisor(s): Jesse Krijthe[1]**, Rickard Karlsson**[1]

[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 25, 2023

Name of the student: Shukun Cheng
Final project course: CSE3000 Research Project
Thesis committee: Jesse Krijthe, Rickard Karlsson, Frans Oliehoek

Abstract

Causal inference is a widely recognized concept in various domains, including medicine, for estimating the effect of a medication on a certain disease. During this estimation, overlap is commonly used to eliminate the error caused by other features. However, finding the real overlap region in practice is challenging due to the limited sample size and unknown data distribution. Therefore, some machine-learning methods have been proposed to estimate the overlap region. One such method is Overrule, a Python package proposed by Oberst et al.[1], Overrule is based on rule-based classification and estimates the overlap region by interpreting it as several rules across the features. However, it is still unclear how Overrule performs under different circumstances. Thus, the primary objective of this project is to test the performance of Overrule with different datasets. To accomplish this, a series of tests are built and executed to evaluate the performance of Overrule in diverse scenarios.

1 Introduction

"Causal inference is the field dedicated to estimating causal effects of some interventions from real-world retrospective data" [2]. It can represent how each causal variable affects the outcome. At its most basic concept, estimating an effect of a causal variable is done by comparing two groups that have different values for the causal variable. For example, smoking may have casual inference on heart disease, to estimate this, a group of smokers needs to be compared with a group of people that do not smoke. However, if some other variables might also affect heart diseases, such as age and exercise frequency, it is necessary to reduce or eliminate the effects of other casual variables. Finding sufficient overlap is a significant step in this process.

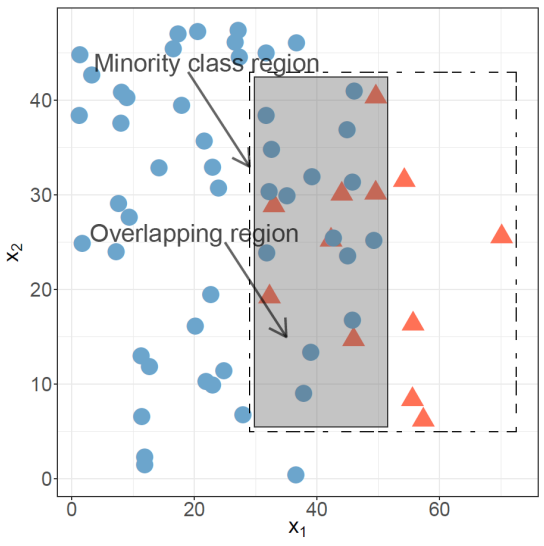


Figure 1: An illustration of an overlapped dataset [3]

Overlap of classes exists when classes share a common region in the data space [3]. Figure 1 shows the overlapping region between two classes. To find the overlap in the dataset, multiple methods can be used such as estimating the class conditional probabilities of all classes. However, with the limited amount of data, it becomes difficult to estimate the probability, especially in a high dimensional problem since the data become sparse. Thus, in this project, we are aiming to use a machine learning algorithm to evaluate overlap. A Python package called Overrule, which is based on a rule-based classification method, is the main focus of this project.

Although Overrule has been implemented by Oberst et al.[1], the possibility of using it in practice and the performance is still unknown. Therefore, this report aims to find the answer to the following research question: What is the performance of Overrule in identifying overlap for different types of datasets? To achieve this goal, several sub-questions need to be answered:

- How does Overrule identify overlap?
- How to find feasible hyperparameters of Overrule with a given dataset?
- Is Overrule sensitive to outlier data points?
- How does Overrule scale as the number of features and samples increases in the dataset?

In this report, the background information about this topic will be listed in Section 2, then in Section 3, some other methods that might evaluate overlap will be explained. Section 4 mainly shows the setup of the experiments, then the results will be shown and discussed in Section 5. After this, Section 6 will explain related issues of this research. In the end, the project will be concluded in Section 7.

2 Background

As mentioned in Section 1, the goal of this project is to test the performance of Overrule under different conditions. In this Section, some background information about this project will be expressed, such as overlap, rule-based classification, and Overrule.

2.1 Overlap

Section 1 has described that the overlap of classes happens when all classes share a common region in the data space [3]. However, this definition is still unclear about how to formulate the common region and use it as a standard to test the performance of Overrule. So, it is necessary to find another way to represent overlap. In this project, most of the testing dataset will be simulated by some distribution function, therefore, posterior probability is used as a baseline in order to compare with the overlapping result from Overrule.

Posterior probability $P(Y|X)$ is the probability of class Y given the data point X [4]. Usually, it is estimated by Bayes' theorem in 1, where $P(Y)$ is the prior probability of class Y, $P(X)$ is the probability of data point X, and $P(X|Y)$ is called class conditional probability. Since the simulated data used for this project has their distribution function, so the posterior probabilities of these data can be easily calculated.

$$P(Y|X) = \frac{P(Y)P(X|Y)}{P(X)} \tag{1}$$

The overlap in this project is defined as where the posterior probabilities of all classes are higher than a predetermined threshold value ϵ as shown in 2, which is set to 0.05 in most of the tests in this project.

$$Overlap = \{X; \forall Y : P(Y|X) > \epsilon\} \tag{2}$$

2.2 Rule-based Classification

In this project, rule-based classification is the main method that will be used. A rule-based classification is a process that extracts relevant IF-Then rules from training data and uses these rules to classify unknown data. This method can be used to evaluate the overlap of given datasets with the support of other methods. There are two different ways to extract rules from datasets, one is using sequential covering algorithms to extract rules directly from the data, which is also called rule induction [5], such as CN2, RIPPER, and Holte's 1R, and the other one takes the rules indirectly from other data mining methods such as decision tree, for example, C4.5rules.

2.3 Overrule

Overrule [1] is used as the base of this project, it is a Python package that uses a rule-based classification method to evaluate the overlap between treatment groups. As Figure 2 shows, this method consists of two main processes, estimating α-minimum-value set S^α of all classes and estimating the rule set B^ϵ using Boolean rules restricted to S^α.

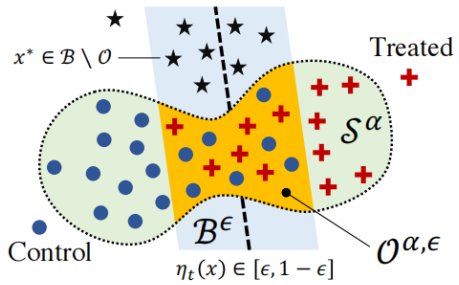


Figure 2: Estimate overlap $O^{\alpha,\epsilon}$ with α-MV set S^α and Boolean rule set B^ϵ [1]

Estimation of α-minimum-value set S^α

Let $x_1, x_2, ...x_m$ be random variables in a set $\mathcal{X}$ with distribution P. Let ℓ be a class of measurable subsets of $\mathcal{X}$ and $V(C)$ denote the volume of a set $C \in \ell$. An α-minimum-volume (α-MV) set S^α is then:

$$S^\alpha := \arg\min_C \{V(C); P(C) \geq \alpha, C \in \ell\} \quad 0 < \alpha \leq 1 \tag{3}$$

This means a set C is the minimum volume set when it contains at least a fraction α of the probability mass.

Based on the research from Schölkopf et al [6], it is difficult to calculate the real S^α, therefore, One-Class SVM is a feasible tool for calculating the estimation of α-MV set $\hat{S}$. It uses a technique called the kernel trick with function Φ to map the data into a higher-dimensional space, the function can be presented as follows

$$k(\mathbf{x}, \mathbf{y}) = (\Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})) \tag{4}$$

The most used kernel is radial basis function (RBF) kernel:

$$k(\mathbf{x}, \mathbf{y}) = e^{-||\mathbf{x}-\mathbf{y}||^2/c} \tag{5}$$

Then isolates them from the origin of the higher-dimensional space with maximum margin as normal SVM. This method will return a function f that determines if the given data point is in this class by evaluating which side of the hyperplane it falls on. Therefore, to separate the data point from the origin, the following function should be solved

$$\begin{aligned} &\min_{w \in F, \xi \in \mathbb{R}^m, \rho \in \mathbb{R}} \frac{1}{2}\|w\|^2 + \frac{1}{\nu m}\sum_i \xi_i - \rho \\ &\text{subject to} \quad (w \cdot \Phi(x_i)) \geq \rho - \xi_i, \xi_i \geq 0 \end{aligned} \tag{6}$$

The parameter ν is an upper bound on the fraction of outliers and a lower bound on the fraction of SVs. Then the decision function f can be expressed as:

$$f(x) = sgn((w \cdot \Phi(x)) - \rho) \tag{7}$$

Where 1 means the given data point belongs to this class, -1 means not. For each class, a One-Class SVM will be trained, then they can generate the estimated α-MV set $\hat{S}$. A data x is in $\hat{S}$ when there is at least one One-Class SVM returns 1.

Estimation of rule set B^ϵ

After estimating the α-MV set $\hat{S}$, the rough estimation of the overlap, $\tilde{B}$ can be found by taking the intersection of all $\hat{S}$. Then the estimation of the real overlap, which is represented as $\hat{B}$ can be calculated with the formula as follows

$$\begin{aligned} \hat{B} := \arg\min_C &\frac{1}{|\hat{S} \setminus \tilde{B}|} \sum_{i: x_i \in \hat{S} \setminus \tilde{B}} \mathbb{1}[x_i \in C] + R(C) \\ \text{subject to} &\sum_{i: x_i \in \hat{S} \cap \tilde{B}} \mathbb{1}[x_i \in C] \geq \beta |\hat{S} \cap \tilde{B}| \end{aligned} \tag{8}$$

Where $R(C)$ is the regularization term that controls complexity by placing penalties λ_0 on each clause in the rule and λ_1 on each condition in a clause [1]. This can be formulated as follows

$$R(C) = K\lambda_0 + \lambda_1 \sum_{k=1}^{K} p_k \tag{9}$$

3 Related Works

Overrule is the main method used in this project, it is based on rule-based classification. Moreover, several other methods that are based on machine learning techniques have also been proposed besides Overrule to evaluate overlap. Such as propensity score, nearest neighbors, novelty detection, and density estimation. In this Section, all of these methods will be explained.

3.1 Propensity score

Crump et al. [7] have proposed a method based on propensity score to estimate overlap. A propensity score is the probability of a sample being assigned to a particular class given a set of observed covariates. For evaluating overlap, the framework used is from Rosenbaum and Rubin [8]. For each sample i in the given dataset, W_i indicates the class that this sample belongs to, and $Y_i(W_i)$ denotes the outcome for sample i with class W_i. Then W_i and Y_i can be observed as follow:

$$Y_i = Y_i(W_i) = \begin{cases} Y_i(0), & W_i = 0 \\ Y_i(1), & W_i = 1 \end{cases} \tag{10}$$

For a dataset with a K-dimensional vector of variables or covariates, denoted by X_i, with support $\mathbb{X} \in \mathbb{R}^K$. Then propensity score can be expressed as:

$$e(x) = P(W_i | X_i = x) \tag{11}$$

To solve the overlap problem with propensity score, two assumptions should be satisfied for any sample in the estimated overlap region:
Assumption 1: $W_i \perp\!\!\!\perp \{Y_i(0), Y_i(1)\} | X_i$
Assumption 2: For some $c > 0$, and all $x \in \mathbb{X}, c \leq e(x) \leq 1 - c$

3.2 Nearest neighbors

The nearest neighbors algorithm is a machine-learning algorithm that classifies the class of a given sample based on its neighbors in the test set. There are two main categories for this algorithm: k nearest neighbors (kNN) and radius neighbors. kNN takes k nearest samples of the test sample, then checks the majority class in these samples and assigns it to the test sample. The other method, radius neighbors, takes all samples within the circle centered on the test sample, then assigns the majority class to the test sample. Both classifiers can be formulated as follows

$$Y_x = \begin{cases} 0, & N_0 > N_1 \\ 1, & N_0 \leq N_1 \end{cases} \tag{12}$$

Where N_i is the number of class i samples in neighbors of sample x. One of the shortages of this classification algorithm is that it may misclassify the samples when there is overlap occurs. But on the other hand, this shortage can be used to evaluate overlap.

3.3 Novelty detection

Novelty detection can be defined as finding the difference in some respect between the test data and training data, this may be seen as one-class classification [9]. This method is also used in Overrule as shown in formula 6. To estimate the overlap region with this algorithm, data from each class should train a separate one-class classification algorithm to get the estimation of the distribution region, then the intersection is taken as estimated overlap.

3.4 Density estimation

Density estimation is a method that estimates the distribution f of a class by assigning a kernel density to every sample in this class. This can be formulated as follow:

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K(\frac{x - X_i}{h}) \tag{13}$$

Where the kernel function K should satisfy $\int K(x)dx = 1$ and h is known as bandwidth. In practice, there are many functions that can be used as K, such as the most popular one, the Gaussian kernel. The following Figure shows the estimation of a normal distribution dataset with three different kernel functions.

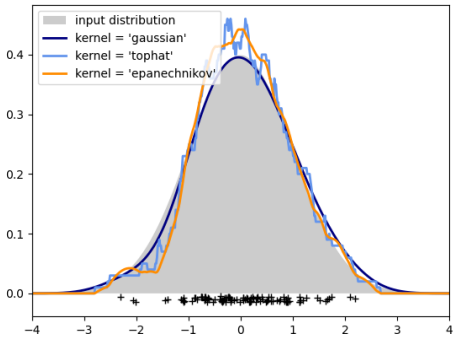


Figure 3: Density estimation of a normal distribution dataset with Gaussian, Tophat, and Epanechnikov

Same as novelty detection algorithm, the density function of each class should be estimated, and the overlap region can be found by checking the densities of all classes as formula 2.

4 Experimental Setup

In order to generate answers for the research question and sub-questions, experiments with different setups are required. In this section, the setups of these experiments will be explained first, such as how the method's performance is measured and how to simulate datasets with specific structures that each experiment needs. Then the results from these experiments will be shown.

4.1 General dataset simulation

When testing the performance of the method in a specific scenario, the effects of other factors should be eliminated. Therefore, simulated data is used for the experiments. In this project, `numpy.random` package is used for this, `numpy.random` is a standard Python package that can produce pseudo-random numbers, which can be defined to distribute under different probability densities. In this project, the most used distribution is a normal distribution [10]. There is an example of two datasets generated by a normal distribution shown in Figure 5.

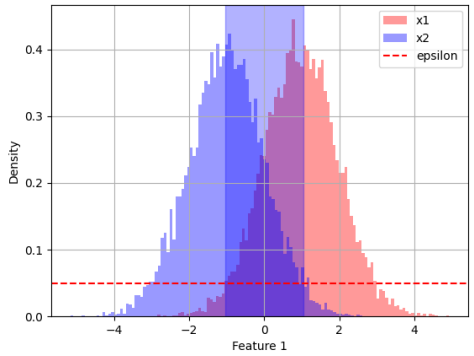


Figure 4: Density of two normal distribution classes $x1$ (μ=-1, σ^2=1, n=100000) and $x2$ (μ=1, σ^2=1, n=100000), blue region is the overlap

4.2 Matrix

Testing the performance of the method needs a baseline to compare with, where intersection over union (IoU) is used for this project. It is originally for comparing the similarity between two arbitrary shapes [11]. However, in this project, it is used for comparing the classification output for each sample in the dataset. It is calculated with a confusion matrix between the actual condition and the predicted condition as shown below:

		Prediction condition	
		Positive (PP)	**Negative (PN)**
Actual condition	**Positive (P)**	True Positive (TP)	False Negative (FN)
	Negative (N)	False Positive (FP)	True Negative (TN)

Table 1: Confusion matrix for classification problem

With this confusion matrix table, we can get the values of TP, FN, FP, and TN with true overlap classification output and the predicted one, the IoU value can be calculated as:

$$IoU = \frac{TP}{TP + FP + FN} \tag{14}$$

However, this formula leads to another question, how to generate the true overlap classification output for all samples? In this project, most of the test datasets are simulated datasets that are generated by a certain distribution function. As mentioned in subsection 2.1, formula 2 represents that overlap occurs when the possibilities of all classes are higher than a predefined value. Therefore, with given distribution functions, it is easy to obtain the true overlap classification.

4.3 Hyperparameters optimazation

As mentioned in Section 2.3, the Overrule package used for this project contains two main hyperparameters, α, and β. where α is the fraction of the probability mass or the minimum volume set, and β is the fixed fraction of the overlap set.

Oberst et al. [1] did not establish the direct relationship between these two hyperparameters (α and β) and the given condition, such as the size of the datasets, the number of features, and the predetermined threshold value. Consequently, it is necessary to identify a method for determining suitable hyperparameter values based on given conditions. Since both α and β have defined boundaries, grid search is a suitable tool to find feasible hyperparameter values. Grid search is a traditional method of hyperparameters optimization, which simply makes a complete search over a given subset of the hyperparameters space of the training algorithm [12].

Listing 1 Code example of Optuna

```
import optuna
import ...

def objective(trial):
    alpha = trial.suggest_float("alpha", 0.01, 1)
    beta = trial.suggest_float("beta", 0.01, 1)
    clf = RuleBasedOverlapEstimator(alpha,
    beta)
    clf.fix(x_train, y_train)

    return clf.score(x_test, y_test)

search_space = {
        'alpha': np.linspace(0, 1, 21),
        'beta': np.linspace(0, 1, 21)
    }
study = optuna.create_styudy(direction="maximize",
    sampler=optuna.samplers.GridSampler(search_space))
study.optimize(objective, n_trial=100)
```

Akiba et al.[13] has designed a Python package called Optuna. It is an open-source optimization software that tries to find feasible hyperparameters with the best performance by using an objective function that takes a set of hyperparameters as input and returns its validation score. Listing 1 shows an example of how Optuna works. In this example, the objective function creates a space of two hyperparameters, $alpha$ and $beta$ with both boundary $[0.01, 1]$. Optuna is able to use different optimization strategies and in this case, grid search is used. The Figure below shows a hyperparameter optimization process for Overrule package, more processes can be checked in Appendix A.

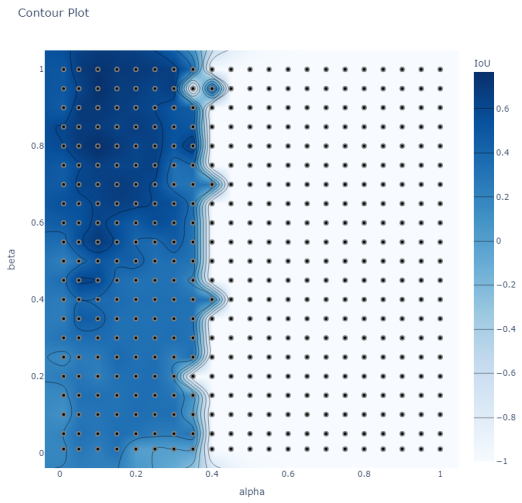


Figure 5: Optuna hyperparameters optimization with grid search for a dataset with 1 feature and 2 normal distributed classes, $x1$ (μ=0, σ^2=1, n=200) and $x2$ (μ=2.5, σ^2=1, n=200)

However, in a real experiment, the real overlap region is unknown, it is impossible to tune hyperparameters as in the process above. Therefore, the goal of this process is to find a set of hyperparameter values that can generate reasonable estimation in most cases.

4.4 Outliers

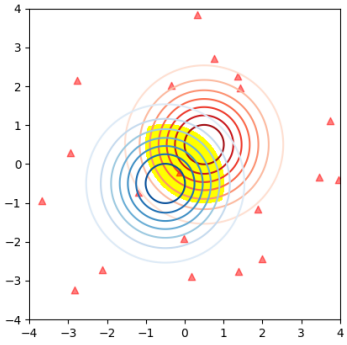


Figure 6: Outlier data (triangle marker) for class $x1$ (red) in a dataset with 2 features, the true overlap is colored in yellow

Hawkins [14] defined outlier as a data sample that deviates a lot from other samples. This occurs frequently in real datasets. In this project, simulated data with normal distribution is mostly used, so outlier data can also be added as a small group of data with a uniform distribution that randomly generates samples in the distribution region. Figure 6 above provides an example of an outlier in a dataset, where the red circle is the distribution of class 1 and the blue one is for class 2, the triangle markers stand for outlier data of class 1, which are far away from the main distribution region. During the experiment, the fraction of the outlier data starts from 1% of the amount of class data and increases to 10% in order to see the change in the performance.

4.5 Different numbers of samples and features

In order to test the performance of Overrule under different conditions, the number of samples and features are considered. During these tests, the number of samples will start from 10 to 30000 with exponential increments, datasets with 1 feature to 6 features will be tested. However, when comparing the performances of different numbers of features, the region of true overlap might be different and influence the outcome. To reduce this effect, for each dataset, the mean and variance are tuned to ensure for every dataset there is approximately 30% of overlap.

5 Results and Discussion

In this section, the results of the experiments during the research will be shown and discussed. First some basic results such as with 1d and 2d normal distribution datasets with default hyperparameters ($\alpha = 0.01$, $\beta = 0.95$). Then the performance under different hyperparameters will be tested. Later the results for different numbers of samples and features will be compared. After this, the performance under different fractions of outlier data will be investigated.

5.1 Basic test

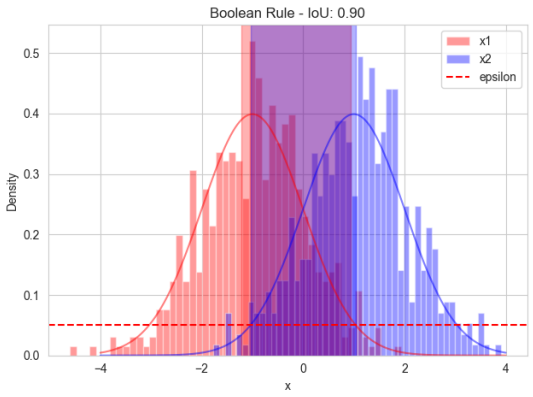


Figure 7: Overlap estimation of two normal distribution classes with 1 feature $x1$ (μ=-1, σ^2=1, n=500) and $x2$ (μ=1, σ^2=1, n=500) with threshold ϵ=0.05, where the blue region is true overlap and the red is estimated overlap

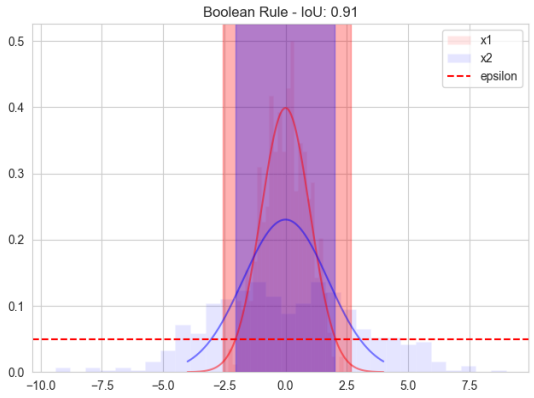


Figure 8: Overlap estimation of two normal distribution classes with 1 feature $x1$ (μ=0, σ^2=1, n=500) and $x2$ (μ=0, σ^2=3, n=500) with threshold ϵ=0.05, where the blue region is true overlap and the red is estimated overlap

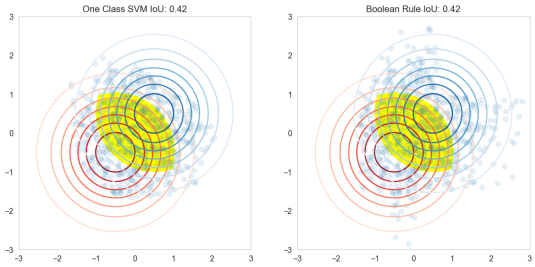
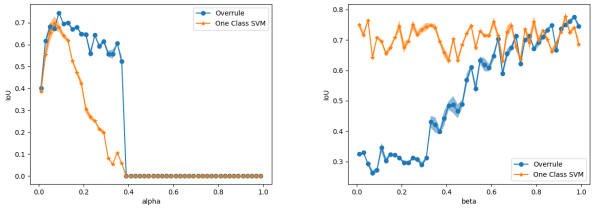


Figure 9: Overlap estimation of two normal distribution classes with 2 features $x1$ (μ=-0.5, σ^2=1, n=500) in red and $x2$ (μ=0.5, σ^2=1, n=500) in blue with threshold ϵ=0.05, where the yellow region is the true overlap region and blue points are the estimated overlap samples

Figure 7 shows the performance of Overrule with default hyperparameters, on evaluating overlap for two normal distributed datasets with 1 feature. In this test, the true overlap is defined as where both classes have densities higher than 0.05. The true overlap is visualized as a blue region in the figure, and the estimated overlap is shown as a red one, where the overlap of both boxes becomes purple. This result means that the default rule-based method overestimated the overlap. Figure 8 has a similar setup but only the distributions of the classes are different, in this experiment one class is fully covered by another class. Figure 9 is from a test with a dataset that has 2 features, as it shows that with default hyperparameters, estimated overlap $\hat{B}$ from Overrule is the same as $\tilde{B}$ from One-Class SVM. These tests indicate that with default hyperparameters, Overrule performs well in 1 feature, but decreases significantly when increasing the number of features. This means that hyperparameter tuning is necessary for Overrule.

5.2 Different hyperparameters

As mentioned in 4.3, both hyperparameters α and β do not have a clear relation with a given dataset. Also in real experiments the true overlap region is unknown, so tuning α and β with IoU is not possible. Therefore, in this subsection, Each hyperparameter will be tested separately to find a regular pattern to ensure the performance of Overrule stays at an optimum level for most of the conditions.

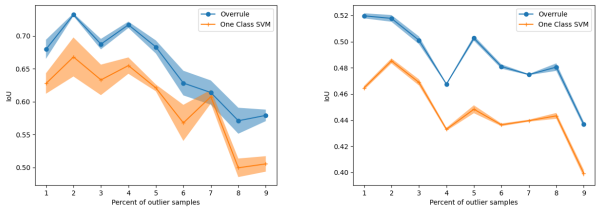


(a) varying α, β = 0.95, ϵ = 0.05 (b) varying β, α = 0.1, ϵ = 0.05

Figure 10: IoU of One-Class SVM and Overrule for dataset(30% overlap) with only one varying hyperparamer

In the two Figures 10a, 10b above, two hyperparameters are tested separately. When testing one of them, the other value is set to the default value, where α is 0.1 and β is 0.95. More results for different numbers of features can be found in Appendix B. From Figure 10a it is known that the overall trend for α is that IoU decreases with the increment of α, but with datasets that have 1 or 2 features, low α will also generate low IoU. Therefore, it is wise to choose a value between low α and the peaks, in this case, 0.1 is assigned to α for the later experiments. Figure 10b shows that β is not as sensitive as α, for most of the experiments the IoU stays at its peak when β is higher than a certain value, however, for some experiments the IoU drops when β reaches the maximum. Thus, 0.9 is assigned to β.

5.3 Outliers



(a) 1 feature (b) 2 features

Figure 11: IoU of One-Class SVM and Overrule in dataset (30% overlap) with different numbers of outlier samples

After completing experiments with the outliers, the results are shown in Figure 11 above. For both the 1 feature and 2 features datasets, it is clear that outlier data does not influence the performance until the fraction increases to 5%.

5.4 Different number of samples and features

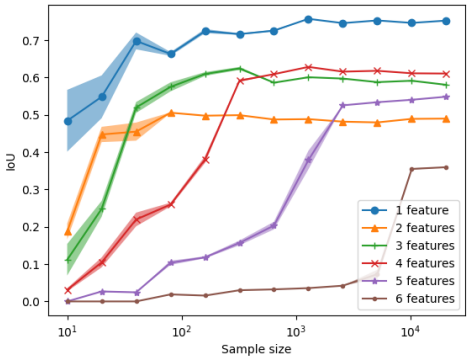


Figure 12: IoU of Overrule for 30% overlap datasets with different numbers of samples and dimensions, where $\alpha = 0.1$, $\beta = 0.9$, $\epsilon = 0.05$

Figure 12 shows the results of Overrule ($\alpha = 0.1$, $\beta = 0.9$) with different numbers of samples and features, when sample size increases, the performance of Overrule first increases and then stays at a stable level, which means that Overrule does not get affected by the number of samples once the number reaches the minimum requirement. With more features, the number of samples needed to reach a stable state will increase.

For different numbers of features, Overrule has decent performance with datasets that have 1, 3, 4, or 5 features. But for datasets with 2 or 6 features, it does not have good results. From Appendix 16b it is known that the influence of α value on the performance has a bell curve with a peak at 0.3, so $\alpha = 0.1$ will cause low IoU for 2 features case. For the dataset with 6 features, IoU drops a lot with the increments of α, which also result in the low IoU in Figure 12.

5.5 Iris dataset

After determining the feasible hyperparameter values through grid search, experiments with real datasets become valuable. In this subsection, the Iris dataset is used to evaluate the performance of Overrule as a practical case study. The Iris dataset is well-known and widely used in the field of machine learning. The figure below scatters 2 chosen classes in Iris with 2 features, it shows that these two classes share the common data space in the middle of the figure, where the overlap occurs.

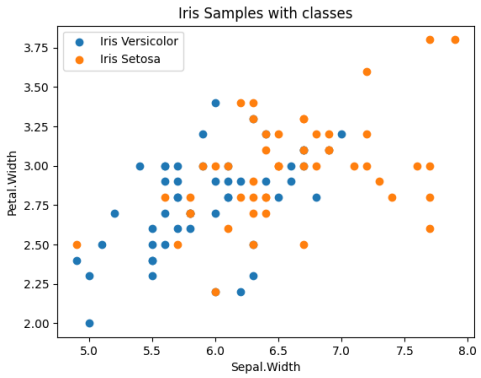


Figure 13: 2 classes in Iris dataset with Sepal.Width and Petal.Width

Overrule estimated the overlap region as follows. As Figure 14 shows that the estimated overlap from Overrule is closely aligned with the actual overlap region.

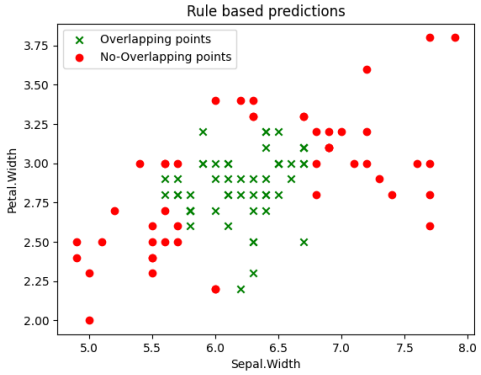


Figure 14: Overrule overlap estimation in Iris with two classes and two features

In Appendix C, additional experimental results are presented, showcasing the performance of different overlap estimation algorithms on the Iris dataset. The figures in the appendix provide comparative analyses of several algorithms, including Overrule, K nearest neighbors LOF, K nearest neighbors LRD, and radius neighbors. The results demonstrate that Overrule, K nearest neighbors LOF, K nearest neighbors LRD, and radius neighbors have good performance. On the other hand, propensity score, and kernel density estimation tend to overestimate the overlap region, they produce a larger overlap region than the actual region.

6 Responsible Research

One of the goals of this project is to investigate the Overrule package and check whether it can contribute to real production. By the end of the project, an open-source library for this project will be built and released publicly in Github[1]. Meanwhile, some issues also come up, such as the possibility of

[1]https://github.com/ShukunCheng/Rule-Based-Overlap-Estimator

reproducing the experiment results, and some ethical problems of misconduct.

6.1 Reproducibility

In this project, all the experiments are implemented in Python and released to the same repository as the main method. Therefore, anyone who installed all required packages is able to reproduce the experiment results locally.

6.2 Misconduct

Evaluating overlap is a common process in many fields, and some of them always come with some ethical issues such as in the medical field. In this project, the performance of Overrule package was only tested by simulated data, the performance on the real dataset is still unknown, also the consequences of misconduct also need to be determined. Therefore, Overrule is still not yet recommended to be used in some sensitive fields.

7 Conclusion and future work

In conclusion, several experiments have been set and tested in this project to answer the subquestions and main research question mentioned in Section 1. It becomes clear how Overrule evaluates overlap by using One-Class SVM and Boolean rule method to generate an interpretable rule set. Grid search is used to find the feasible choice of the values for hyperparameters α and β, with the experiments that have been done, $\alpha = 0.1$, $\beta = 0.9$ seems to be a feasible option for most of the cases. With this hyperparameter set, Overrule performs mostly well with different numbers of samples and features. When the number of features increases, Overrule requires more samples to have stable performance. Overrule is not sensitive to outlier data since in the experiments the performance starts dropping when the fraction of outliers is higher than 5%.

After this project, more work could be organized to gain more insight into Overrule package. Experiments that have been done were based on normal distribution data, it would be valuable to redo the experiments with datasets that have different densities or two separated overlap regions. Also, datasets with varying amounts of classes or different types of features could be tested to check the scalability of Overrule. Since all experiments of this project were using simulated data, the performance of Overrule in practice is unknown, testing with real data will make the result more convincing.

References

[1] Michael Oberst, Fredrik Johansson, Dennis Wei, Tian Gao, Gabriel Brat, David Sontag, and Kush Varshney. Characterization of overlap in observational studies. In *International Conference on Artificial Intelligence and Statistics*, pages 788–798. PMLR, 2020.

[2] Ehud Karavani, Peter Bak, and Yishai Shimoni. A discriminative approach for finding and characterizing positivity violations using decision trees. *arXiv preprint arXiv:1907.08127*, 2019.

[3] Pattaramon Vuttipittayamongkol, Eyad Elyan, and Andrei Petrovski. On the class overlap problem in imbalanced data classification. *Knowledge-based systems*, 212:106631, 2021.

[4] Vijay Kotu and Bala Deshpande. Chapter 4 - classification. In Vijay Kotu and Bala Deshpande, editors, *Data Science (Second Edition)*, pages 65–163. Morgan Kaufmann, second edition edition, 2019.

[5] Xiaoli Li and Bing Liu. Rule-based classification., 2014.

[6] 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.

[7] Richard K Crump, V Joseph Hotz, Guido W Imbens, and Oscar A Mitnik. Dealing with limited overlap in estimation of average treatment effects. *Biometrika*, 96(1):187–199, 2009.

[8] Paul R Rosenbaum and Donald B Rubin. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55, 1983.

[9] Marco AF Pimentel, David A Clifton, Lei Clifton, and Lionel Tarassenko. A review of novelty detection. *Signal processing*, 99:215–249, 2014.

[10] Eric W Weisstein. Normal distribution. *https://mathworld. wolfram. com/*, 2002.

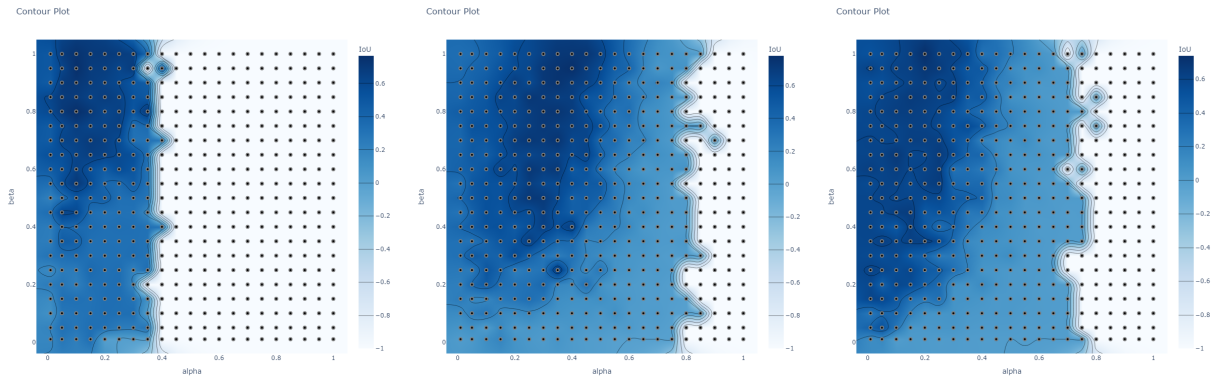
[11] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 658–666, 2019.

[12] Petro Liashchynskyi and Pavlo Liashchynskyi. Grid search, random search, genetic algorithm: a big comparison for nas. *arXiv preprint arXiv:1912.06059*, 2019.

[13] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.

[14] Douglas M Hawkins. *Identification of outliers*, volume 11. Springer, 1980.

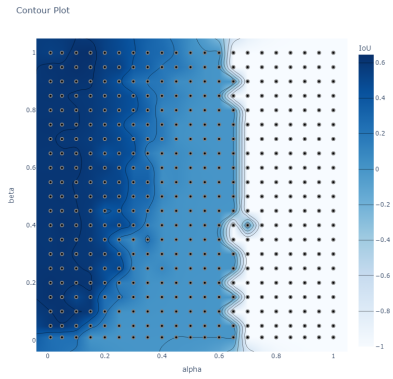
A Optuna hyperparameter optimization



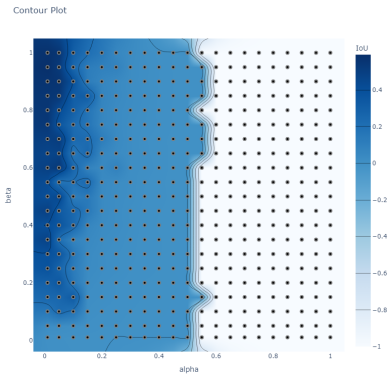
(a) 1 feature

(b) 2 features

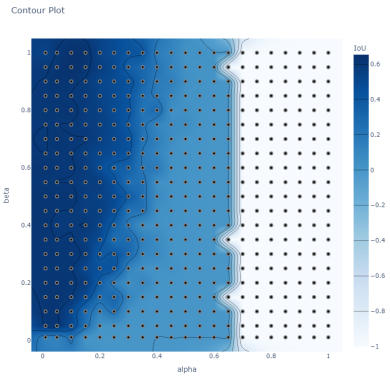
(c) 3 features



(d) 4 features



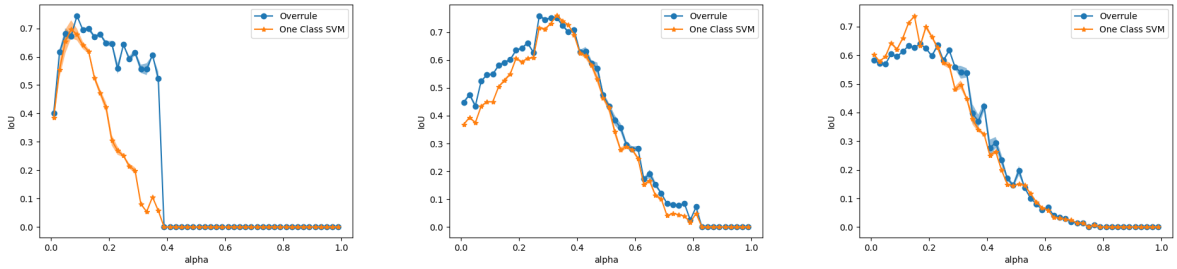
(e) 5 features



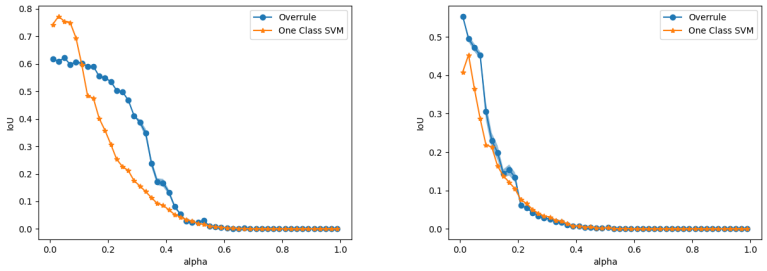
(f) 6 features

Figure 15: Optuna optimization result for datasets with different number of features

B Grid search for one hyperparameter

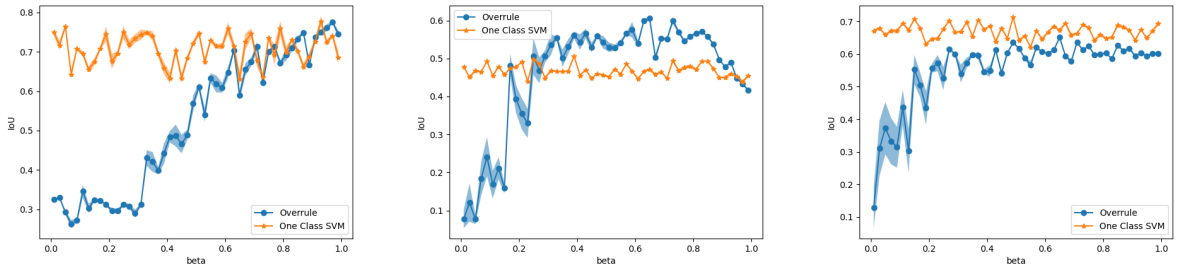


(a) 1 feature (b) 2 features (c) 3 features

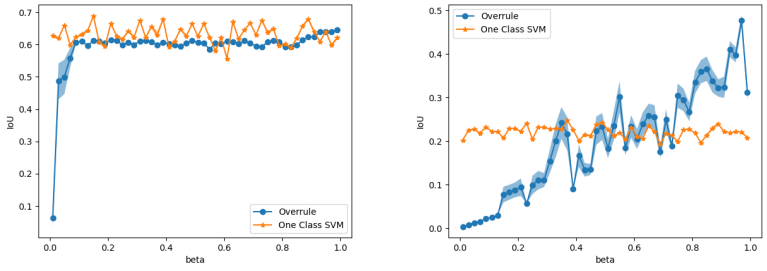


(d) 4 features (e) 5 features

Figure 16: IoU of One-Class SVM and Overrule for dataset(30% overlap) with 1 feature and different α, $\beta = 0.95$, $\epsilon = 0.05$



(a) 1 feature (b) 2 features (c) 3 features



(d) 4 features (e) 5 features

Figure 17: IoU of One-Class SVM and Overrule for dataset(30% overlap) with 1 feature and different β, $\alpha = 0.1$, $\epsilon = 0.05$

C Iris dataset comparison

This project used Overrule to evaluate overlap, meanwhile, as mentioned in Section 3, several other machine learning algorithms have been proposed for this topic. In this Appendix, the results of these different algorithms will be compared, the dataset used for this comparison is Iris dataset from sklearn package. Two classes and two features are chosen for this experiment. The chosen classes and features are shown as a figure below

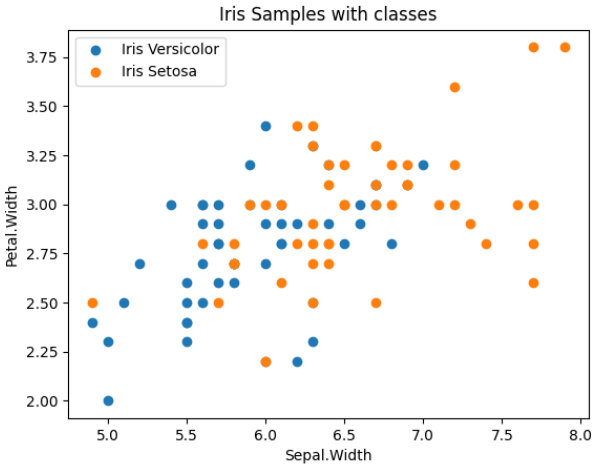
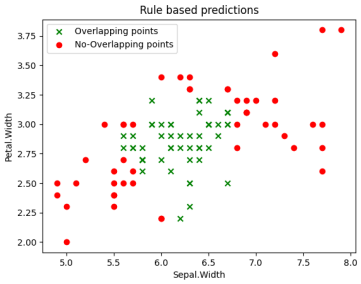
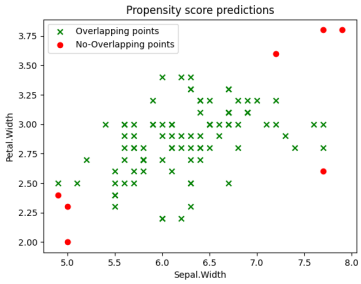


Figure 18: Iris dataset with

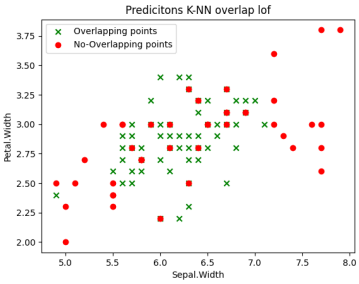
The algorithms that are included in this comparison are Overrule, propensity score, K nearest neighbors LOF, K nearest neighbors LRD, radius neighbors, and kernel density estimation. All results are shown in the figures below



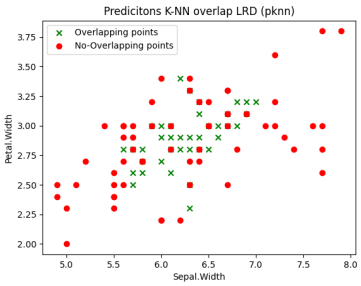
(a) Overrule



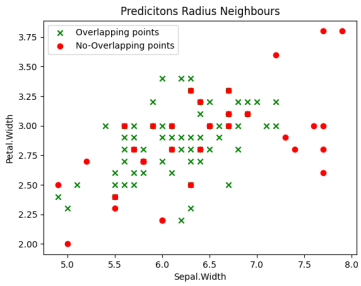
(b) propensity score



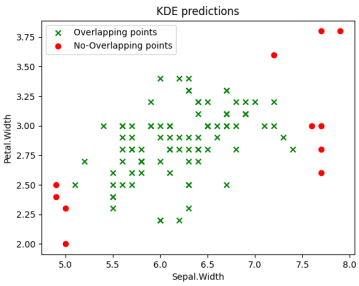
(c) K nearest neighbors LOF



(d) K nearest neighbors LRD



(e) radius neighbors



(f) Kernel density