

# Adaptive Prototype-Based Classification via Graph Theoretic and Topological Methods

by

Jordan Eckert

A dissertation submitted to the Graduate Faculty of  
Auburn University  
in partial fulfillment of the  
requirements for the Degree of  
Doctor of Philosophy

Auburn, Alabama  
August 9, 2025

Keywords: Graph-Based Classification, Class Cover Catch Digraphs, Hybrid Learning,  
Ensemble Graph Learning, Topological Data Analysis, Prototype Selection

Copyright 2025 by Jordan Eckert

Approved by

Elvan Ceyhan, Professor of Mathematics and Statistics  
Nedret Billor, Professor of Mathematics and Statistics  
Henry Schenck, Professor of Mathematics and Statistics and Rosemary Kopel Brown Chair  
Jingyi Zheng, Associate Professor of Mathematics and Statistics

## Abstract

In statistical learning, many modern methods represent data using graphs to capture structure and relationships. Among these, class cover catch digraphs (CCCDs) were originally introduced to address the class cover problem (CCP) and have since been applied to classification and clustering tasks. This dissertation addresses two distinct, yet complementary, challenges in statistical learning: (i) classification performance degradation under class imbalance and class overlap, and (ii) reducing data cardinality through a novel, principled prototype selection method. We propose modified CCCD variants that improve robustness and generalization in imbalanced and overlapped class settings while preserving the geometric intuition of the original CCCD framework. These contributions enhance the practical utility of CCCD classifiers. In addition, we introduce a topological data analysis (TDA)-based framework for selecting representative subsets (prototypes) from large datasets. We show that this approach preserves classification performance while substantially reducing data size. Such methods are crucial in resource-constrained environments where memory and computation are limited. Together, these contributions advance both algorithmic and geometric aspects of prototype learning and offer practical tools for scalable, interpretable, and efficient classification.

## Acknowledgments

There have been many along this journey that have helped me reach this point. For them, I'm eternally grateful. However, no one was as supportive and helpful to my academic journey than my parents. Over the years, their sacrifices and encouragements have flourished into this dissertation. Therefore, I would like to dedicate this work to my Mom and Dad. It's been a privilege and an honor being your son, and I hope to continue making you both proud.

As a statistician, I am honored to have had the opportunity to work with the brilliant professors at Auburn University who have guided me to a better understanding of mathematical and statistical thinking. In particular, I would like to thank Prof. Elvan Ceyhan, who I've been lucky enough to be both a student and advisee under. His support, as an professor, an advisor, and as a colleague, have been invaluable for me on this journey. He has constantly pushed me to achieve greater things, and for that I acknowledge how lucky I was to have a Ph.D. advisor who is both open-minded and positively critical. I should also mention the occasional funny comments on his part that have made the long, arduous process of creating this work enjoyable.

Finally, I would like to thank my friends at the Department of Math & Stats at Auburn for making these last few years of my life joyful and meaningful. There are too many of you to list individually, but know that all of you have a special place in my heart. It is indeed a remarkable gift to have your friends by your side when you need them the most. I can only hope that they have enjoyed my company as much as I enjoyed theirs. I will always be grateful for the late nights working problems at the whiteboards, weekends spent together, and I hope for a future that we never lose touch with each other.

War Damn Eagle

## Table of Contents

|                                                       |              |
|-------------------------------------------------------|--------------|
| Abstract . . . . .                                    | ii           |
| Acknowledgments . . . . .                             | iii          |
| List of Tables . . . . .                              | viii         |
| List of Figures . . . . .                             | xi           |
| List of Abbreviations . . . . .                       | xvii         |
| <br><b>I Preliminaries</b>                            | <br><b>1</b> |
| 1 Introduction . . . . .                              | 2            |
| 1.1 Motivation . . . . .                              | 2            |
| 1.2 Previous Work on Graph - Based Learners . . . . . | 3            |
| 1.3 Contributions . . . . .                           | 4            |
| 2 Preliminary Tools . . . . .                         | 6            |
| 2.1 Topology Background . . . . .                     | 6            |
| 2.1.1 Topological Data Analysis . . . . .             | 9            |
| 2.2 $k$ -Nearest Neighbors Algorithm . . . . .        | 12           |
| 2.3 Support Vector Machines . . . . .                 | 13           |
| 2.3.1 Linearly Separable Case . . . . .               | 14           |
| 2.3.2 Non-linearly Separable Case . . . . .           | 18           |
| 2.3.3 Kernel SVM . . . . .                            | 20           |
| 2.3.4 Support Vector Data Description . . . . .       | 22           |
| 2.4 Decision Trees . . . . .                          | 24           |
| 2.4.1 C4.5 and C5.0 Algorithms . . . . .              | 25           |
| 2.5 Class Cover Catch Digraphs . . . . .              | 26           |



|           |                                                                              |           |
|-----------|------------------------------------------------------------------------------|-----------|
| 2.5.1     | Classification with P-CCCDs . . . . .                                        | 28        |
| 2.5.2     | Classification with RW-CCCDs . . . . .                                       | 29        |
| 2.5.3     | Multi-Class Classification . . . . .                                         | 32        |
| 2.6       | Algorithmic Modifications for Complex Data Intrinsic Characteristics . . . . | 32        |
| 2.6.1     | Resampling Methods . . . . .                                                 | 33        |
| 2.6.2     | Ensemble Frameworks . . . . .                                                | 35        |
| 2.6.3     | Cost-Sensitive Methods . . . . .                                             | 38        |
| 2.6.4     | Hybrid Classification Methods . . . . .                                      | 39        |
| 2.7       | Metrics for Measuring Performance Under Data Intrinsic Characteristics . .   | 40        |
| 2.7.1     | Area Under the Curve . . . . .                                               | 41        |
| 2.7.2     | $F_1$ -Score . . . . .                                                       | 41        |
| 2.7.3     | Geometric Mean . . . . .                                                     | 42        |
| <b>II</b> | <b>Algorithmic Modifications of Graph-Based Learners</b>                     | <b>43</b> |
| 3         | Enhancing P-CCCD Performance in Class Overlap Settings . . . . .             | 44        |
| 3.1       | Modified P-CCCD with Cardinality-Based Scoring . . . . .                     | 44        |
| 3.1.1     | Consistency of MP-CCCD . . . . .                                             | 45        |
| 3.2       | Bagged P-CCCD . . . . .                                                      | 50        |
| 3.3       | Monte Carlo Simulations . . . . .                                            | 51        |
| 3.3.1     | Performance Under Class Overlap Settings . . . . .                           | 52        |
| 3.3.2     | Performance Under Class Overlap and Class Imbalance Settings . . .           | 57        |
| 3.4       | Real Data Experiments . . . . .                                              | 63        |
| 3.5       | Conclusions and Discussions . . . . .                                        | 70        |
| 4         | A Hybrid Approach to Classification with CCCDs . . . . .                     | 71        |
| 4.1       | Support Vector Class Cover Catch Digraph (SV-CCCD) . . . . .                 | 71        |
| 4.2       | Monte Carlo Simulations . . . . .                                            | 74        |
| 4.2.1     | Performance Under Class Overlap Settings . . . . .                           | 76        |

|            |                                                                           |            |
|------------|---------------------------------------------------------------------------|------------|
| 4.2.2      | Performance Under Various Class Imbalance and Overlap Settings . . .      | 80         |
| 4.3        | Real Data Experiments . . . . .                                           | 86         |
| 4.4        | Conclusions and Discussions . . . . .                                     | 92         |
| 5          | Adaptive Ensemble Classification with CCCDs: The ACDC Algorithm . . . . . | 94         |
| 5.1        | Statistical Complexity . . . . .                                          | 94         |
| 5.2        | FlexiBall Classifiers . . . . .                                           | 99         |
| 5.3        | AdaCover Digraph Classifier (ACDC) . . . . .                              | 104        |
| 5.4        | Monte Carlo Simulations . . . . .                                         | 107        |
| 5.4.1      | Performance Under Various Class Overlap Settings . . . . .                | 109        |
| 5.4.2      | Performance Under Various Class Imbalance and Overlap Settings . .        | 113        |
| 5.5        | Real Data Experiments . . . . .                                           | 120        |
| 5.6        | Conclusions and Discussions . . . . .                                     | 126        |
| <b>III</b> | <b>Topological Prototype Selection</b>                                    | <b>128</b> |
| 6          | Topological Data Analysis as a Tool for Prototype Selection . . . . .     | 129        |
| 6.1        | Topological Data Analysis and the Prototype Selection Problem . . . . .   | 129        |
| 6.2        | Topological Prototype Selector (TPS) . . . . .                            | 131        |
| 6.2.1      | Recovering a meaningful sub-complex . . . . .                             | 132        |
| 6.2.2      | Algorithm Details . . . . .                                               | 134        |
| 6.3        | Performance Evaluation on Synthetic Data . . . . .                        | 137        |
| 6.4        | Performance Evaluation on Real Data . . . . .                             | 142        |
| 6.5        | Conclusions and Discussions . . . . .                                     | 147        |
| <b>IV</b>  | <b>Conclusions</b>                                                        | <b>149</b> |
| 7          | Summary and Discussions . . . . .                                         | 150        |
| 7.1        | Future Work . . . . .                                                     | 153        |

|          |                                  |            |
|----------|----------------------------------|------------|
| <b>V</b> | <b>Appendix &amp; References</b> | <b>155</b> |
| 8        | Appendix . . . . .               | 156        |

## List of Tables

|     |                                                                                                                                                                                                                                                                                                                        |    |
|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 2.1 | Cost matrix for binary classification settings. . . . .                                                                                                                                                                                                                                                                | 39 |
| 3.1 | The description of other CCCD and MP-CCCD classifiers used for comparison in simulation studies. . . . .                                                                                                                                                                                                               | 52 |
| 3.2 | Performance difference of MP-CCCD versus BP-CCCD, P-CCCD, and RW-CCCD for balanced data across different scenarios of class overlap and evaluation metrics. Positive differences indicate MP-CCCD performed better. . . . .                                                                                            | 57 |
| 3.3 | Performance difference of MP-CCCD versus BP-CCCD, P-CCCD, and RW-CCCD for imbalanced data across different scenarios of class overlap and evaluation metrics. Positive differences indicate MP-CCCD performed better. . . . .                                                                                          | 61 |
| 3.4 | Aggregated mean and standard deviations for P-CCCD, RW-CCCD, BP-CCCD, and MP-CCCD runtimes across simulated data settings. . . . .                                                                                                                                                                                     | 62 |
| 3.5 | Data intrinsic characteristics for the selected UCI and KEEL repository datasets.                                                                                                                                                                                                                                      | 64 |
| 3.6 | Mean and standard deviation of MP-CCCD and CCCD models' AUC across selected real data. Values highlighted in red indicate the best performer of all models for a specific dataset. The results of the relevant $5 \times 2$ CV $F$ -test against MP-CCCD are presented in parenthesis when relevant. . . . .           | 67 |
| 3.7 | Mean and standard deviation of MP-CCCD and CCCD models' $F_1$ -scores across selected real data. Values highlighted in red indicate the best performer of all models for a specific dataset. The results of the relevant $5 \times 2$ CV $F$ -test against MP-CCCD are presented in parenthesis when relevant. . . . . | 68 |
| 3.8 | Mean and standard deviation of MP-CCCD and CCCD models' G-mean across selected real data. Values highlighted in red indicate the best performer of all models for a specific dataset. The results of the relevant $5 \times 2$ CV $F$ -test against MP-CCCD are presented in parenthesis when relevant. . . . .        | 69 |
| 4.1 | The description of other CCCD and strong classifier variants used for comparison against SV-CCCD in simulation studies. . . . .                                                                                                                                                                                        | 75 |
| 4.2 | Performance difference of SV-CCCD versus CCCD and other strong classifiers for balanced data across different scenarios of class overlap and evaluation metrics. Positive differences indicate SV-CCCD performed better. . . . .                                                                                       | 79 |

|     |                                                                                                                                                                                                                                                                                                                                                                                                                                          |     |
|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 4.3 | Performance difference of SV-CCCD versus CCCD and other strong classifiers for imbalanced data across different scenarios of class overlap and evaluation metrics. Positive differences indicate SV-CCCD performed better. . . . .                                                                                                                                                                                                       | 84  |
| 4.4 | Aggregated mean and standard deviations for SV-CCCD and other selected strong classifiers runtimes across simulated data settings. . . . .                                                                                                                                                                                                                                                                                               | 85  |
| 4.5 | Mean and standard deviation of SV-CCCD and other chosen models' AUCs across chosen real data. Values highlighted in red indicate the best performer of all models for a specific dataset. Values highlighted in blue indicate the best performer out of CCCD variant models. The results of the relevant 5x2 CV $F$ -test are presented in parenthesis below. A p-value of "NA" indicates the same performance across all folds. . . . . | 89  |
| 4.6 | Mean and standard deviation of SV-CCCD and other chosen models' $F_1$ -scores across chosen real data. Values highlighted in red indicate the best performer of all models for a specific dataset. Values highlighted in blue indicate the best performer out of CCCD variant models. The results of the relevant 5x2 CV $F$ -test are presented in parenthesis when relevant. . . . .                                                   | 90  |
| 4.7 | Mean and standard deviation of SV-CCCD and other chosen models' G-mean scores across chosen real data. Values highlighted in red indicate the best performer of all models for a specific dataset. Values highlighted in blue indicate the best performer out of CCCD variant models. The results of the relevant 5x2 CV $F$ -test are presented in parenthesis when relevant. . . . .                                                   | 91  |
| 4.8 | Comparison of original SV-CCCD, modified $\delta^*$ SV-CCCD, and SMOTE+ENN preprocessed SV-CCCD average and standard deviation for selected real datasets. Values in blue would be the highest performing CCCD variant. Values in red would be the highest performing of all models. . . . .                                                                                                                                             | 92  |
| 5.1 | The description of other CCCD and strong classifier variants used for comparison against ACDC in simulation studies. . . . .                                                                                                                                                                                                                                                                                                             | 108 |
| 5.2 | Performance difference of ACDC versus CCCD and other strong model variants for balanced data across different scenarios of class overlap and evaluation metrics. Positive differences indicate ACDC performed better. . . . .                                                                                                                                                                                                            | 112 |
| 5.3 | Performance difference of ACDC versus CCCD and other strong classifiers for imbalanced data across different scenarios of class overlap and evaluation metrics. Positive differences indicate ACDC performed better. . . . .                                                                                                                                                                                                             | 118 |
| 5.4 | Aggregated mean and standard deviations for ACDC and other selected strong classifiers runtimes across simulated data settings. . . . .                                                                                                                                                                                                                                                                                                  | 120 |

|     |                                                                                                                                                                                                                                                                                                                                                                               |     |
|-----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 5.5 | Mean and standard deviation of ACDC and chosen models' AUC across chosen real data. Values highlighted in red indicate the best performer of all models for a specific dataset. Values highlighted in blue indicate the best performer out of CCCD variant models. The results of the relevant 5x2 CV $F$ -test are presented in parenthesis when relevant. . . . .           | 123 |
| 5.6 | Mean and standard deviation of ACDC and other chosen models' $F_1$ scores across chosen real data. Values highlighted in red indicate the best performer of all models for a specific dataset. Values highlighted in blue indicate the best performer out of CCCD variant models. The results of the relevant 5x2 CV $F$ -test are presented in parenthesis. . . . .          | 124 |
| 5.7 | Mean and standard deviation of ACDC and other chosen models' G-means across chosen real data. Values highlighted in red indicate the best performer of all models for a specific dataset. Values highlighted in blue indicate the best performer out of CCCD variant models. The results of the relevant 5x2 CV $F$ -test are presented in parenthesis when relevant. . . . . | 125 |
| 6.1 | Descriptions of the nine synthetic datasets used for testing prototype selection performance of TPS. . . . .                                                                                                                                                                                                                                                                  | 137 |
| 6.2 | TPS performance comparison across the selected synthetic datasets. The best configurations for TPS recorded were the ones that achieved the highest accuracy.                                                                                                                                                                                                                 | 138 |
| 6.3 | Descriptions of the real datasets used for testing prototype selection performance of TPS. . . . .                                                                                                                                                                                                                                                                            | 142 |
| 6.4 | Performance of TPS across the selected real datasets. The best configurations for TPS recorded were the ones that achieved the highest accuracy. . . . .                                                                                                                                                                                                                      | 143 |
| 6.5 | TPS performance comparison against CNN+ENN and SPOT across the selected real datasets. . . . .                                                                                                                                                                                                                                                                                | 147 |

## List of Figures

|     |                                                                                                                                                                                                                                               |    |
|-----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 2.1 | Demonstration of homotopy equivalence of $X$ and $Y$ . One “leg” of $X$ is retracted, and the remaining is adjusted to get the shape $Y$ without tearing or gluing. . .                                                                       | 7  |
| 2.2 | Geometric representations of $q$ -simplices. Each corresponds to their respective tetrahedral dimensional analogues. . . . .                                                                                                                  | 8  |
| 2.3 | Comparison of Čech complex (left) with Rips complex (right) at the same radius. The simplicial Čech complex consists of a hole since all three balls do not overlap, where the Rips complex does not. . . . .                                 | 10 |
| 2.4 | Example of barcode and persistent homology visualizations. Left is the original data, middle is the corresponding persistent homology diagram (birth, death) pairs plotted, and right is the barcodes for specific filtration values. . . . . | 12 |
| 2.5 | Optimal separation hyperplane, with parallel hyperplanes. The distance between the hyperplanes is the margin with the points in highlighted boxes lying along the hyperplanes being the support vectors. . . . .                              | 16 |
| 2.6 | Two radii, one with $\tau = .0001$ for the dotted circle and $\tau = 1$ for the solid circle, for the metric covering ball centered at $x$ . The gray and black points represent the target and non-target class, respectively. . . . .       | 27 |
| 2.7 | A snapshot of $R_x(r)$ at two different radii for the covering ball $B_x$ centered at $x$ .                                                                                                                                                   | 31 |
| 3.1 | Surface plots of optimal AUC counts for each $(\tau, e)$ combination of MP-CCCD classifiers on balanced, overlapping simulations. As classes become more overlapping, average counts decrease. . . . .                                        | 54 |

|     |                                                                                                                                                                                                                                                                             |    |
|-----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 3.2 | Heatmaps of the average optimal $F_1$ -score counts for each $(\tau, e)$ combination of MP-CCCD classifiers on balanced, overlapping simulations. In high overlap settings, $F_1$ -scores likely down-weights cardinality, except for few data-dependent scenarios. . . . . | 55 |
| 3.3 | Heatmaps of the average optimal G-mean counts for each $(\tau, e)$ combination of MP-CCCD classifiers on balanced, overlapping simulations. $(\tau, e)$ combination performance is similar to that of $F_1$ -score. . . . .                                                 | 56 |
| 3.4 | Scatterplot of optimal $\tau$ (left) and optimal $e$ (right) selections for MP-CCCD as overlap increases under each level of imbalance under AUC metric. . . . .                                                                                                            | 58 |
| 3.5 | Scatterplot of optimal $\tau$ (left) and optimal $e$ (right) selections for MP-CCCD as overlap increases under each level of imbalance under $F_1$ -score metric. . . . .                                                                                                   | 59 |
| 3.6 | Scatterplot of optimal $\tau$ (left) and optimal $e$ (right) selections for MP-CCCD as overlap increases under each level of imbalance under G-mean metric. . . . .                                                                                                         | 59 |
| 3.7 | Distribution of BP-CCCD optimal $\tau^*$ counts for the combination of each metric, overlap, and imbalance ratio. Class overlap plays a bigger role in selection of $\tau^*$ than class imbalance. . . . .                                                                  | 60 |
| 4.1 | An example of the naïve $\delta$ -separable construction for $\delta = 2$ . . . . .                                                                                                                                                                                         | 72 |
| 4.2 | Construction of $\delta$ -separable regions using SVM soft margin boundaries when $\delta = 2$ . . . . .                                                                                                                                                                    | 74 |
| 4.3 | Surface plots of optimal AUC counts for each $(e, \gamma)$ combination of SV-CCCD classifiers on balanced, overlapping simulations. Kernel parameter $\gamma$ plays a bigger role in tuning for optimal AUC than $e$ . . . . .                                              | 76 |
| 4.4 | Surface plots of optimal $F_1$ -score counts for each $(e, \gamma)$ combination of SV-CCCD classifiers on balanced, overlapping simulations. Kernel parameter $\gamma$ plays a bigger role in tuning for optimal $F_1$ -score than $e$ . . . . .                            | 77 |



|     |                                                                                                                                                                                                                                                           |     |
|-----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 4.5 | Surface plots of optimal G-mean counts for each $(e, \gamma)$ combination of SV-CCCD classifiers on balanced, overlapping simulations. Kernel parameter $\gamma$ plays a bigger role in tuning for optimal G-mean than $e$ . . . . .                      | 78  |
| 4.6 | Average optimal AUC counts for $\gamma$ kernel parameter in SV-CCCD across different imbalance levels. . . . .                                                                                                                                            | 81  |
| 4.7 | Heatmaps of the average optimal $F_1$ -score counts for each $(e, \gamma)$ combination of SV-CCCD classifiers on imbalanced, overlapping simulations. In high overlap and imbalance settings, larger $\gamma$ are more often selected as optimal. . . . . | 82  |
| 4.8 | Heatmaps of the average optimal G-mean counts for each $(e, \gamma)$ combination of SV-CCCD classifiers on imbalanced, overlapping simulations. In high overlap and imbalance settings, larger $\gamma$ are more often selected as optimal. . . . .       | 83  |
| 5.1 | Example of how the classification decision rule for SBCs change depending on which point is selected. . . . .                                                                                                                                             | 100 |
| 5.2 | The maximum empirical Rademacher complexity with bootstrapped confidence intervals for P-CCCD, FlexiBalls of size $\sqrt{n}$ , and SBC across different $n$ values. . . . .                                                                               | 105 |
| 5.3 | Contour plots of optimal AUC counts for each $(k, \tau)$ combination of ACDC classifiers on balanced, overlapping simulations. As classes become more overlapped, ACDC favors larger $k$ and smaller $\tau$ . . . . .                                     | 109 |
| 5.4 | Contour plots of optimal $F_1$ -score counts for each $(k, \tau)$ combination of ACDC classifiers on balanced, overlapping simulations. As classes become more overlapped, ACDC favors larger $k$ and smaller $\tau$ . . . . .                            | 110 |
| 5.5 | Contour plots of optimal G-mean counts for each $(k, \tau)$ combination of ACDC classifiers on balanced, overlapping simulations. As classes become more overlapped, ACDC favors larger $k$ and smaller $\tau$ . . . . .                                  | 110 |

|     |                                                                                                                                                                                                              |     |
|-----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 5.6 | Normalized selection rates of $k$ and $\tau$ for optimal AUC on imbalanced, overlapping datasets. ACDC selects larger $k$ for all imbalance levels when class overlap is high.                               | 114 |
| 5.7 | Normalized selection rates of $k$ and $\tau$ for optimal $F_1$ -score on imbalanced, overlapping datasets. ACDC selects larger $k$ for all imbalance levels when class overlap is high. . . . .              | 115 |
| 5.8 | Normalized selection rates of $k$ and $\tau$ for optimal G-mean on imbalanced, overlapping datasets. ACDC selects larger $k$ for all imbalance levels when class overlap is high. . . . .                    | 116 |
| 6.1 | An example of a commutative diagram for a bifiltration of $\mathcal{K}$ across two parameters. Each $\mathcal{K}_{i,j}$ represents a sub-complex. . . . .                                                    | 131 |
| 6.2 | Graphical example of representing a bifiltration in a 2D grid. Radius and curvature scale parameters change the underlying sub-complex. . . . .                                                              | 133 |
| 6.3 | Accuracy difference (in percent) of baseline 1-NN against 1-NN after each configuration of TPS is applied for each synthetic dataset. . . . .                                                                | 139 |
| 6.4 | Dataset reduction (in percent) training data split after each configuration of TPS is applied for each synthetic dataset. . . . .                                                                            | 140 |
| 6.5 | Example of prototype selection for $q = 0.10$ (left) and $q = 0.25$ (right) under $H_0$ configurations on the noisy circles synthetic dataset. Lower $q$ values have a greater reduction percentage. . . . . | 141 |
| 6.6 | Accuracy difference (in percent) of baseline 1-NN against 1-NN after each configuration of TPS is applied for each real dataset. . . . .                                                                     | 144 |
| 6.7 | Dataset reduction (in percent) training data split after each configuration of TPS is applied for each of the selected real dataset. . . . .                                                                 | 145 |

|     |                                                                                                                                                                                                                                       |     |
|-----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 7.1 | Plot of interaction effects for ACDC- $\sqrt{N}$ classifier runtimes across different dimension, overlap shift parameter $\Delta$ , and imbalance ratio. Increasing dimension greatly increases runtimes. . . . .                     | 152 |
| 1   | Example of TPS prototype selection on well-separated blobs. The left shows the original dataset and 1-NN accuracy, while the right is the selected prototypes, data reduction percentage, and comparison 1-NN accuracy. . . . .       | 156 |
| 2   | Example of TPS prototype selection on class overlapping blobs. The left shows the original dataset and 1-NN accuracy, while the right is the selected prototypes, data reduction percentage, and comparison 1-NN accuracy. . . . .    | 157 |
| 3   | Example of TPS prototype selection on class overlapping clusters. The left shows the original dataset and 1-NN accuracy, while the right is the selected prototypes, data reduction percentage, and comparison 1-NN accuracy. . . . . | 157 |
| 4   | Example of TPS prototype selection on two moons. The left shows the original dataset and 1-NN accuracy, while the right is the selected prototypes, data reduction percentage, and comparison 1-NN accuracy. . . . .                  | 158 |
| 5   | Example of TPS prototype selection on noisy two moons. The left shows the original dataset and 1-NN accuracy, while the right is the selected prototypes, data reduction percentage, and comparison 1-NN accuracy. . . . .            | 158 |
| 6   | Example of TPS prototype selection on concentric circles. The left shows the original dataset and 1-NN accuracy, while the right is the selected prototypes, data reduction percentage, and comparison 1-NN accuracy. . . . .         | 159 |
| 7   | Example of TPS prototype selection on noisy concentric circles. The left shows the original dataset and 1-NN accuracy, while the right is the selected prototypes, data reduction percentage, and comparison 1-NN accuracy. . . . .   | 159 |

|   |                                                                                                                                                                                                                                    |     |
|---|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 8 | Example of TPS prototype selection on imbalanced classes. The left shows the original dataset and 1-NN accuracy, while the right is the selected prototypes, data reduction percentage, and comparison 1-NN accuracy. . . . .      | 160 |
| 9 | Example of TPS prototype selection on mixed multi-class blobs. The left shows the original dataset and 1-NN accuracy, while the right is the selected prototypes, data reduction percentage, and comparison 1-NN accuracy. . . . . | 160 |

## List of Abbreviations

$k$ -NN     $k$ -Nearest Neighbors

ACDC    AdaCover Digraph Classifier

AUC    Area Under the Receiver Operating Characteristics Curve

BP-CCCD    Bagged Pure-Class Cover Catch Digraphs

BPS    Bifiltration Prototype Selection Algorithm

$Ck$ -NN    Cost-Sensitive  $k$ -NN

CCCD    Class Cover Catch Digraphs

CCP    Class Cover Problem

CNN    Condensed Nearest Neighbor

CSVM    Cost-Sensitive Support Vector Machines

CV    Coefficient of Variation

CV    Cross-Validation

$Ek$ -NN    EasyEnsemble  $k$ -NN

EC4.5    EasyEnsemble C4.5

ENN    Edited Nearest Neighbor

ESVM    EasyEnsemble SVM

FN    False Negatives

FP      False Positives

FPR    False Positive Rate

G-Mean   Geometric Mean

KKT    Karush-Kuhn-Tucker

MDS    Minimum Dominating Sets

MP-CCCD   Modified Pure-Class Cover Catch Digraphs

OCC    One-Class Classification

PCD    Proximity Catch Digraphs

RBF    Radial Basis Function

ROC    Receiver Operating Characteristic

RW-CCCD   Random-Walk Class Cover Catch Digraph

$Sk$ -NN   SMOTE+ENN  $k$ -NN Algorithm

SAMME   Stagewise Additive Modeling using a Multi-class Exponential loss

SBC    Single-Ball Classifier

SC4.5   SMOTE+ENN C4.5 Algorithm

SMOTE   Synthetic Minority Oversampling Technique

SPOT   Selecting Prototypes using Optimal Transport Algorithm

SSVM   SMOTE+ENN SVM Algorithm

SV-CCCD   Support Vector Class Cover Catch Digraph

SVDD   Support Vector Data Description

|     |                                |
|-----|--------------------------------|
| SVM | Support Vector Machine         |
| TDA | Topological Data Analysis      |
| TN  | True Negatives                 |
| TP  | True Positives                 |
| TPS | Topological Prototype Selector |

# Part I

## Preliminaries



## Chapter 1

### Introduction

#### 1.1 Motivation

Much research has been done on machine learning methods based on set covering algorithms since these algorithms have shown potential for advanced learning. Formally, a *cover* is a region (a portion of space where every point can be reached from any other point within it) that encapsulates specific members of the data set or *class of interest* [19]. In this work, we approach statistical learning with methods that solve the *class cover problem* (CCP), where the goal is to find regions that cover classes of interest. This problem is closely related to the *support estimation problem* in statistics, where the support of a random variable is estimated in an arbitrary measurable space. Class covers serve as estimators of the support of a class-specific distribution [115]. In addition, by investigating the shape and structure of a cover, latent sub-classes of a class may be revealed which makes these covers effective tools of exploratory data analysis [99]. Classification is the most pertinent machine learning application for class covers, since any query point can be labeled as a member of that class if it falls within the cover [19]. The family of class covering methods we discuss in this thesis utilize subsets of data, called *prototype sets*, for classification. The *prototype selection problem* is critical in reducing data complexity, enhancing computational efficiency, and improving the interpretability of machine learning models [92].

Classifier performance is inherently tied to *data intrinsic characteristics*. *Class imbalance* and *class overlap* are widely recognized as major challenges in traditional classification algorithms [57, 64, 73]. Class imbalance occurs when one (or more) classes are represented by far more observations than the other class(es) in the data set [23, 72]. The *class imbalance problem* has become a topic of research in many areas such as clinical trials of patients with

rare diseases [81]; detecting fraudulent credit card transactions [96]; and archiving collegiate student performance where records often retain students with average or higher performance [107]. Class overlap occurs when samples of more than one class share a common region of the data space. Class overlap has long been recognized as one of the most challenging and pervasive problems in classification [141]. Empirical studies have demonstrated that misclassification errors predominantly concentrate near decision boundaries, where class overlap can be significant [69].

Although numerous methods address these issues individually, few studies approach the combined class overlap problem and class imbalance problem due to the increased complexity. In this work, we approach algorithmic modifications for graph-based methods to improve classification performance under conditions of both class imbalance and overlap. Specifically, we focus on *spherical proximity graphs* to construct families of digraphs called *class cover catch digraphs* (CCCDs). We also address the *prototype selection problem* using topological data analysis (TDA) to identify an informative subset based on the underlying class *topology*, or shape and connectivity of the data.

## 1.2 Previous Work on Graph - Based Learners

Early formulations of class covers estimated class support using collections of metric covering balls, with subsets selected through approximation algorithms [19]. Subsequent generalizations extended the geometric representation of covers to include alternative shapes. One such approach utilizes axis-aligned boxes within the framework of the *maximum redundancy problem*, which seeks to maximize coverage per unit while minimizing the total number of shapes required [116]. Other methods adopt rectilinear polygons to define covers and study the associated computational complexity [13]. Convex hulls have been employed to construct classifiers analogous to CCCD-based models [126]. Triangle-based covers derived from the Delaunay tessellation of the opposite class have been shown to yield analytical bounds on

the *minimum dominating sets* needed to define a valid cover [22]. Extensions to higher-dimensional settings using simplicial regions demonstrate that certain families of proximity catch digraphs (PCDs) can produce mathematically tractable minimum dominating sets [77].

Approximate minimum dominating sets of class cover catch digraphs (CCCDs), along with the associated covering ball radii, have been established as efficient classifiers [100]. New families of CCCDs have been proposed to address practical limitations. Random-walk CCCDs (RW-CCCDs) relax the strict exclusion of other class observations in the covering balls in exchange for increased generalization [33]. Hierarchically boosted versions of RW-CCCDs have been shown to be effective for fast facial detection [120]. Ceyhan and Manukyan (2019) further demonstrated that CCCDs are robust to class imbalance but remain vulnerable to class overlap [76]. Their work also benchmarked CCCD variants against traditional state-of-the-art classifiers across both synthetic and real-world datasets.

### 1.3 Contributions

CCCDs offer a powerful prototype-driven framework that can be adapted to address longstanding challenges in statistical learning, such as the class imbalance and class overlap problems. In this work, we propose a series of algorithmic modifications to CCCDs to address class imbalance and class overlap. Building on this principle, we explore

- (i) a novel cardinality-based scoring rule for a family of CCCDs,
- (ii) hybrid models that integrate CCCDs with state-of-the-art classifiers to better discriminate overlapping class regions, and
- (iii) a new ensemble-friendly CCCD variant for which we provide theoretical learnability results.

We evaluate the performance of these enhanced CCCD models on both synthetic and real-world datasets, and benchmark them against other established state-of-the-art classifiers. In

addition to CCCD-based improvements, we introduce a novel prototype selection strategy grounded in TDA.

TDA provides the tools to uncover the geometric and topological structure of data, offering insights into its intrinsic shape. We use persistent homology to identify observations that lie along the borderline between classes’ topological features and show how these points can serve as informative prototypes. This approach aims to retain the essential structure of the data while enabling significant reductions in data size, improving both interpretability and computational efficiencies.

The rest of the dissertation is broken down as follows. Chapter 2 details background information necessary for topological data analysis, classification, current methods for handling class imbalance and overlap, and evaluation metrics that will be used throughout the work. Chapter 3 discusses the use of cardinality-based scoring for the CCCD decision rule. Chapter 4 discusses a hybrid CCCD-based approach to quantifying and classifying under class overlap. Chapter 5 introduces a new family of CCCDs called FlexiBalls and discusses an adaptive ensemble classifier. Chapter 6 details the topological prototype selection algorithm. We close with discussions about our proposed methods and future research directions in Chapter 7.

## Chapter 2

### Preliminary Tools

In this chapter, we introduce tools from topology (namely topological spaces, simplices, simplicial complexes, homology, filtrations) which will be used in Chapter 6, and four families of classifiers ( $k$ -nearest neighbors, support vector machines, decision tree, and CCCDs) that will be used for comparison in later Chapters. Additionally, we introduce the current modification methods for classification under class imbalance and class overlap, as well as the evaluation metrics used for calculating classifier performance under these conditions.

#### 2.1 Topology Background

*Topology* is the mathematical study of shape, structure, and spatial properties that do not depend on exact distances or angles, but rather on *connectivity*. This makes topology naturally suited for exploring data where the underlying structure is unknown or poorly estimated [20]. In order to understand the role of topology in prototype selection, we must establish background information on topological spaces and their representation through simplicial complexes. We begin by establishing the notion of a *topological space*.

**Definition 2.1.1.** A *topological space* is any set equipped with a topology. A *topology* on a set  $\mathcal{X}$  is a collection  $\mathcal{T}$  of subsets of  $\mathcal{X}$  satisfying three axioms [87]:

- (i) both  $\emptyset$  and  $\mathcal{X}$  belong to  $\mathcal{T}$ ,
- (ii) arbitrary unions of elements in  $\mathcal{T}$  remain in  $\mathcal{T}$ , and
- (iii) finite intersections of elements in  $\mathcal{T}$  remain in  $\mathcal{T}$



Figure 2.1: Demonstration of homotopy equivalence of X and Y. One “leg” of X is retracted, and the remaining is adjusted to get the shape Y without tearing or gluing.

Topology focuses on properties preserved under *continuous deformations*. Notions of equivalence for topological spaces are formalized through continuous maps and their inverses. Two spaces are *homeomorphic* if there exists a continuous, bijective function from one space to another with a continuous inverse. Informally, both spaces can be transformed into the other without “tearing” or “being glued”. Topological spaces are *homotopy equivalent* if one space can be continuously deformed to have the same “shape” as another, but the converse is not necessarily true [87]. Homotopy equivalence is a weaker condition than homeomorphisms as it does not require a continuous inverse.

Figure 2.1 illustrates a homotopy equivalence between the letters X and Y. The transformation contracts and removes the left segment of the X, after which the remaining segment is bent to form the bifurcated structure of the Y. Rather than treating the letters X and Y as distinct objects, topology allows us to deduce properties of one from the other, as they belong to the same homotopy class and thus share equivalent *homotopical features*.

While topological spaces provide the theoretical foundation for continuous spaces, computational topology requires discrete representations called *simplices* [36]. A *simplex* provides discrete representations by encoding topological information through combinatorial structures [114].

**Definition 2.1.2.** A *q-simplex*,  $\sigma$ , is the generalization of a tetrahedral region of space to  $q$  dimensions. Given a  $q$ -simplex  $\sigma$ , a  $d$ -simplex  $\tau$  with  $0 \leq d \leq q$  and vertex sets  $\mathcal{V}(\tau) \subseteq \mathcal{V}(\sigma)$  is called a *d-face* of  $\sigma$  denoted by  $\tau \leq \sigma$ . We also call  $\sigma$  a *q-coface* of  $\tau$  denoted by  $\sigma \geq \tau$ .

Note, the nomenclature used in this section coincides with others used throughout. When using symbols with dual meanings, such as  $\tau$ ,  $\sigma$ , we will make it obvious to which we

are referring. The vertex set  $\mathcal{V}(\sigma)$  is the set of *vertices* of  $\sigma$  and the simplex  $\sigma$  is said to be *generated* by  $\mathcal{V}(\sigma)$ . The dimension of a simplex  $\sigma$  is one less than the cardinality of its vertex set, that is  $\dim(\sigma) = |\mathcal{V}(\sigma)| - 1$ . Geometrically, simplices correspond to either a vertex point (0-simplex), edges between vertices, (1-simplex), triangles (2-simplex), tetrahedra (3-simplex), and their higher-dimensional analogues as seen in Figure (2.2).

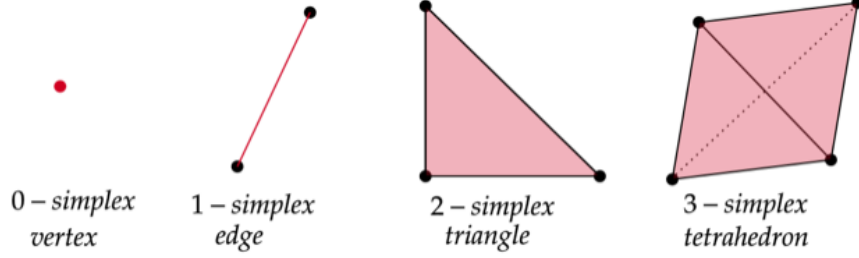


Figure 2.2: Geometric representations of  $q$ -simplices. Each corresponds to their respective tetrahedral dimensional analogues.

Simplices are assigned orientation and boundary operators represented by matrices [114, 36]. Collections of simplices that are glued together in a specific orientation are called *simplicial complexes*.

**Definition 2.1.3.** A simplicial complex  $\mathcal{K}$  in  $\mathbb{R}^n$  is a finite collection of simplices in  $\mathbb{R}^n$  such that

- (i) Given  $q$ -simplex  $\sigma \in \mathcal{K}$  and  $\tau \leq \sigma$  then  $\tau \in \mathcal{K}$ , and
- (ii) If  $\sigma_1, \sigma_2 \in \mathcal{K}$  then  $\sigma_1 \cap \sigma_2$  is either a face of both  $\sigma_1$  and  $\sigma_2$  or is empty.

*Homology* provides an algebraic framework for quantifying topological features using simplicial complexes. The *homology group*,  $H_k(\mathcal{X})$ , is calculated by defining boundary maps between the *chain* of different groups of formal linear combinations of  $k$ -simplices, finding the kernels and images of these maps, and then taking the quotient of the kernels by the images at each stage. For each dimension  $k$ ,  $H_k(\mathcal{X})$  characterizes the  $k$ -dimensional “holes” in the topological space  $\mathcal{X}$  [20, 36].  $H_0(\mathcal{X})$  measures connected components,  $H_1(\mathcal{X})$  measures

“loops” or 1-dimensional holes,  $H_2(\mathcal{X})$  measures “voids” or 2-dimensional holes, and so on for  $H_k(\mathcal{X})$  measuring  $k$ -dimensional analogues. One *topological invariant*, is the rank of  $H_k(\mathcal{X})$ , called the *Betti number* which measures the number of independent  $k$ -dimensional holes [114]. Topological data analysis (TDA) provides a principled framework for understanding the intrinsic geometric and topological structure of data using homology, and will play an important role in prototype selection.

### 2.1.1 Topological Data Analysis

Under the TDA framework, the topological space  $\mathcal{X}$  is taken to be a discrete set of data. Analysis of the topological structure of this data via homology occurs along multiple scales, motivating the idea of a *filtration* of the data [36, 114].

**Definition 2.1.4.** Let  $\mathcal{K}$  be a simplicial complex. A *filtration*,  $\mathcal{F}$ , on  $\mathcal{K}$  is a succession of increasing *sub-complexes* of  $\mathcal{K}$ ,  $\emptyset \subseteq \mathcal{K}_0 \subseteq \mathcal{K}_1 \subseteq \mathcal{K}_2 \cdots \subseteq \mathcal{K}_n = \mathcal{K}$ .

In TDA, the scales along a filtrations correspond to metric covering balls with increasing radii that form covers of the set of data,  $\mathcal{X}$ . Filtrations in TDA applications are dominated by two primary simplicial complex constructions, the *Čech complex* and the *Vietoris-Rips (Rips) complex* [114].

**Definition 2.1.5.** The *Čech complex*,  $\check{C}_\epsilon(\mathcal{X})$ , is a simplicial complex with  $q$ -simplices if and only if there exists  $\{x_i\}_{i=1}^{q+1} \in \mathcal{X}$  tuples of points such that  $\bigcap_{i=1}^{q+1} \overline{B_\epsilon(x_i)} \neq \emptyset$ . That is, there exists a  $q$ -simplex if the intersections of the closed  $\epsilon$ -distanced balls centered at each of the  $q + 1$  points is non-empty.

Čech complexes act as a *nerve complex*. Nerve complexes are abstract complexes that record the pattern of intersections between the points in  $\mathcal{X}$ . The *nerve theorem* states that that under conditions where Čech complexes form *good covers*, the simplicial complex is homotopy equivalent to  $\mathcal{X}$  [114]. We define good covers below.



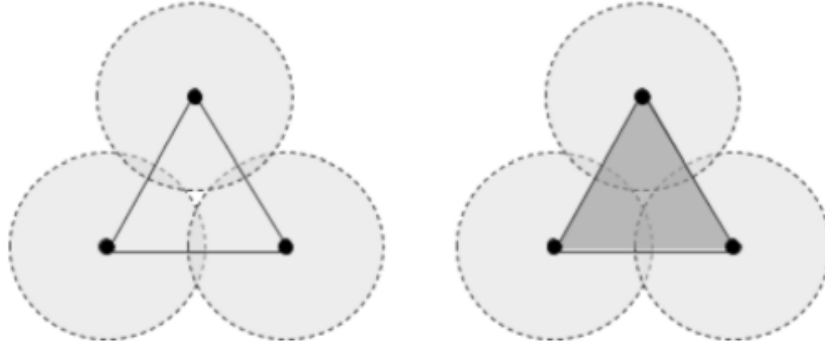


Figure 2.3: Comparison of Čech complex (left) with Rips complex (right) at the same radius. The simplicial Čech complex consists of a hole since all three balls do not overlap, where the Rips complex does not.

**Definition 2.1.6.** Let  $\mathcal{X}$  be a topological space. Let  $\mathcal{U}$  be a non-empty and finite collection of sets  $\mathcal{U} = \{U_i\}_{i \in I}$  such that  $\mathcal{U}$  covers  $\mathcal{X}$ . That is,  $\mathcal{X} = \bigcup_{i \in I} U_i$ . If all  $U_i \in \mathcal{U}$  and all finite, nonempty intersections of  $U_i$  are able to be shrunk to a single point we say  $\mathcal{U}$  is a *good cover*.

Rips complexes do not require that require all  $\epsilon$ -radius balls to intersect together to form a simplicial complex, just that pairwise overlap occurs.

**Definition 2.1.7.** The *Vietoris-Rips complex*,  $R_\epsilon(\mathcal{X})$ , is a simplicial complex with  $q$ -simplices if and only if there exists  $q + 1$  sets of points of  $\mathcal{X}$  such that  $d(x_i, x_j) \leq \epsilon$  for all  $x_i, x_j \in \mathcal{X}$  such that  $i \neq j$  for distance metric  $d(\cdot, \cdot)$ .

The Rips complex only needs to check the pairwise distances to determine if higher-dimensional simplices exist. If all edges exist, the higher-dimensional simplex is automatically included, making Rips complexes more computationally efficient to compute compared to Čech complexes. Rips complexes provide only approximate topological accuracy to the underlying  $\mathcal{X}$ . We provide a comparison of both simplicial complexes in Figure (2.3).

We can “trap” exact topological information from Čech complexes using Rips complexes. The *de Silva Theorem* states that  $R_\epsilon \hookrightarrow \check{C}_{\sqrt{2}\epsilon} \hookrightarrow R_{\sqrt{2}\epsilon}$ . Therefore, it is computationally feasible to determine the Rips complex at some scale that captures the essential topological features of the Čech complex at a related scale and vice-versa [114].

TDA relies on the notion of *persistent homology*, which extends classical homology to filtrations by tracking how homology changes across the different  $\epsilon_i$ -radius scales in the filtration. The collection of  $\{\epsilon_i\}$  scales are the *filtration value collection* of positive values  $0 < \epsilon_0 < \epsilon_1 < \dots < \epsilon_n$  and the filtration corresponds to the sub-complex  $\mathcal{K}_i$  generated at each  $\epsilon_i$ . That is sub-complex  $\mathcal{K}_i$  is generated from the metric balls with radius  $\epsilon_i$ . Each  $\epsilon_i$  corresponds to a critical point of the underlying manifold, as this where the topology changes under Morse theory [114, 36].

Given a filtration  $\{\mathcal{K}_i\}_{i=0}^n$ , the inclusion maps  $\mathcal{K}_i \hookrightarrow \mathcal{K}_j$  induce homomorphisms  $H_k(\mathcal{K}_i) \rightarrow H_k(\mathcal{K}_j)$  on homology groups. Essentially, each level change from  $i$  to  $j$  will correspond to either a “birth” or a “death” of a topological feature. This yields a sequence of vector spaces connected by linear maps called a *persistence module*. Persistence modules decompose into “birth” and “death” pairs, each representing the “lifetime” of a topological feature [114, 20, 60]. The length of the interval of a (birth, death) pair, is called its *persistence*. Long persistence correspond to underlying geometric features in the data, while short persistence is likely noise. Persistence can be graphically represented through *barcodes* [47]. We can also visualize the lifetimes of topological features indicating when it first appears and when it disappears in the filtration by plotting (birth, death) pairs on *persistence diagrams* [20]. We display both styles of visualizations on an example data set in Figure (2.4). A crucial property of persistent homology is its stability, as small perturbations in the input data produce small changes in barcodes and persistence diagrams.

Many applications, especially those in biological fields, require simultaneous consideration of filtrations across multiple scale parameters [140, 133]. For instance, in analyzing medical imaging, one might vary both distance between points on a manifold and the curvature of the manifold to get a more in-depth topological understanding [111, 2]. This motivates the study of *bifiltrations*, a two-parameter extension of a filtration. Many of the fundamental theorems for filtrations do not have analogous variants for bifiltrations. Unlike one-parameter persistence, multi-parameter persistence modules do not admit a complete

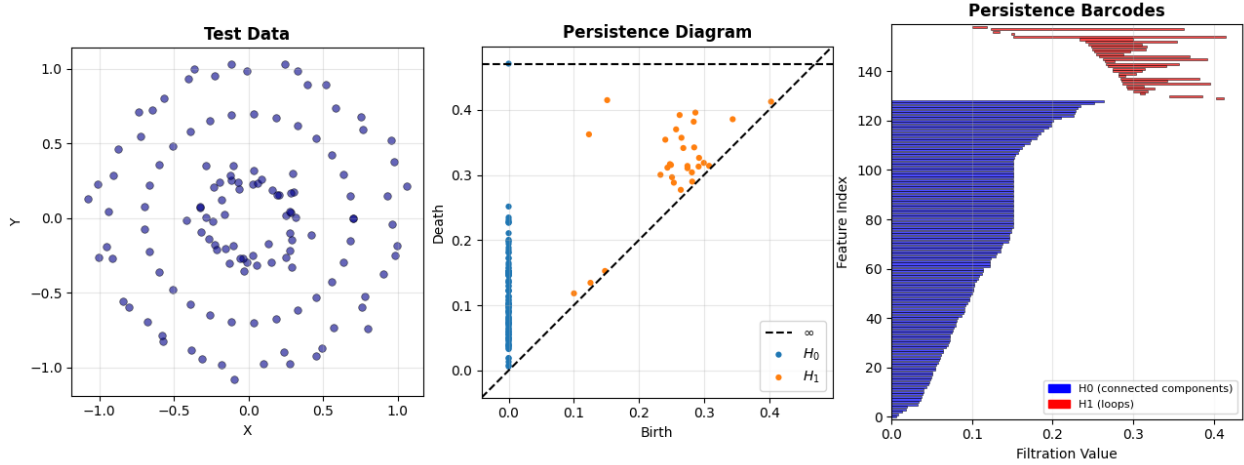


Figure 2.4: Example of barcode and persistent homology visualizations. Left is the original data, middle is the corresponding persistent homology diagram (birth, death) pairs plotted, and right is the barcodes for specific filtration values.

discrete invariant. The classification of finitely generated modules over multivariable polynomial rings is much more complicated than the corresponding result for single variable polynomial rings [114, 111]. We will discuss bifiltrations further in Chapter 6. Having introduced topological invariants, we now turn to classical state-of-the-art classifier families that will be used for comparison in Chapters 3-5.

## 2.2 $k$ -Nearest Neighbors Algorithm

For our work, we assume  $\mathcal{X} = \{\mathcal{X}_0, \dots, \mathcal{X}_K\} \subset \mathbb{R}^d$  is a set of data (called “points” or “observations”) from  $K$ -classes from the  $d$ -dimensional reals equipped with the usual Euclidean metric with class conditional distributions,  $F_{\mathcal{X}_i}$  for  $i = 0, 1, \dots, K$  and joint distribution  $F_{\mathcal{X}}$  respectively. Let  $s(\cdot)$  represent the support functional for a specific class.

Machine learning classification algorithms attempt to learn a mapping from the data onto a set of labels which represent each class; that is, attempt to learn the function  $f : \mathbb{R}^d \mapsto \mathcal{Y}$ . Models can become too complex and “memorize” the mapping of  $\mathcal{X}$  instead of learning the underlying patterns [50]. Typically, a subset of the original data called *training data* is used to provide an estimate for  $f$ , denoted  $\hat{f}$ . We call this estimate the *decision*

*rule*. In this way, we can validate a model’s *generalization ability* on previously withheld data called *testing data*.

In binary classification, there are only two labels in  $\mathcal{Y}$ . For convenience, statisticians often use  $\mathcal{Y} = \{0, 1\}$ , but in reality any two labels quantifies binary classification. Multi-class classification refers to classification with the cardinality of  $\mathcal{Y}$  being greater than 2. In all binary classification settings, the *target class* refers to  $\mathcal{X}_j$  and *non-target class*  $\mathcal{X}_{1-j}$  for either  $j = 0, 1$ . We expand this notation for multi-class classification to target class  $\mathcal{X}_j$  and non-target classes  $\mathcal{X}_{i \neq j}$  for  $i, j = 0, 1, \dots, K$ .

We begin with the  $k$ -nearest neighbors ( $k$ -NN) algorithm. Using a query point from the testing set,  $z$ , the decision rule is a majority vote of the classes from the  $k$  closest neighbors to  $z$ . That is,

$$\hat{f}(z) = \arg \max_{y_i \in \mathcal{Y}} \sum_{x_i \in N_k(z)} f(x_i) \quad (2.1)$$

where  $N_k(z)$  is the neighborhood of  $z$  defined by the  $k$  closest points to  $z$  in the training sample, and  $f(x_i)$  represents the class label of point  $x_i$ .

The decision rule boundaries under  $k$ -NN algorithms are often irregular, responding to local imbalance in the data [66, 46]. The draw of  $k$ -NN is the inherent flexibility and simple easy implementation.  $k$ -NN often gives good performance on “well-behaved” training and testing data, but is characterized by its high-variance and low-bias since different  $k$  values drastically change performance of the classifier [50]. Even in cases where the training and testing data are not well-behaved, the algorithm can be used in tandem with other more sophisticated classifiers to improve classification performance [124].

## 2.3 Support Vector Machines

Support vector machines (SVMs) are a form of sparse kernel decision learner that are a variant of discriminant classifiers whose decision rules are based on finding a multidimensional

surface in the “margin” between different classes to act as a boundary between classes. For SVMs, finding this separating hyperplane is done analytically as a convex optimization problem, which means that SVM will always return the same optimal hyperplane parameters regardless of initialization and termination criteria [132].

### 2.3.1 Linearly Separable Case

We first consider the case where data is *linearly separable*. Data is linearly separable if there exists a hyperplane such that all observations belonging to one class are on one side of the hyperplane, and all observations belonging to the other class are on the opposite side of it [50]. The *optimal separating hyperplane* is the one that maximizes the margin, and will generalize better than the other infinite possibilities [132, 50, 29]. Geometrically, infinitely many hyperplanes can exist and be used for classification. The rudimentary binary classification rule for query observation  $z$  becomes

$$f(\hat{z}) = \begin{cases} 0, & w^T \cdot z + b < 0 \\ 1, & w^T \cdot z + b > 0 \end{cases} \quad (2.2)$$

with vector weights  $w$  and biases  $b$ . We can find the signed distance from a given training point  $x_i \in \mathcal{X} \subset \mathbb{R}^d$  to the defined hyperplane, simplifying the decision rule to just  $f(x_i)(w^T \cdot x_i + b) \geq 0$ . The margin,  $M$ , is defined as the distances between the hyperplane and the closest  $x_i$ .

The optimization problem for the optimal hyperplane is

$$\begin{aligned} & \max_{w^T, b, ||w||=1} M \\ & \text{subject to } f(x_i)(w^T \cdot x_i + b) \geq M \text{ for all } i = 1, \dots, N \end{aligned} \quad (2.3)$$

where  $N$  is the cardinality of the training data. Linear separability implies that there is some guaranteed distance between the training observations and the optimal separating hyperplane [18].

The set of conditions in the optimization ensures that all points are at least a signed distance  $M$  from the boundary defined by the weights and biases. For any weight and bias vector satisfying the inequalities, any positively scaled multiple satisfies them too [50]. We can manipulate the  $\|w\| = 1$  constraint in the optimization problem by maximizing the margin subject to  $\frac{1}{\|w\|}f(x_i)(w^T \cdot x_i + b) \geq M$  for  $i = 1, \dots, N$ , or equivalently  $f(x_i)(w^T \cdot x_i + b) \geq M\|w\|$  for  $i = 1, \dots, N$ .

Consider two hyperplanes that run parallel to the optimal separating hyperplane that intersect the points that are closest to the optimal separating hyperplane. Let the hyperplane intersecting target class observations be  $H_1$  and the one intersecting non-target class observations be  $H_2$ . Then the distance from observations along  $H_1$  to the separating hyperplane should be the same as the distance from observations along  $H_2$  to the separating hyperplane. Let  $x_i^+$  be an observation lying along  $H_1$  and  $x_i^-$  along  $H_2$ . We can calculate the margin as,

$$\begin{aligned} M &= \frac{1}{2} \left[ \left\langle \frac{w}{\|w\|}, x_i^+ \right\rangle - \left\langle \frac{w}{\|w\|}, x_i^- \right\rangle \right] \\ &= \frac{1}{2\|w\|} [\langle w, x_i^+ \rangle - \langle w, x_i^- \rangle] \\ &= \frac{1}{\|w\|} \end{aligned} \tag{2.4}$$

where  $\langle a, b \rangle$  represents the inner product between vectors  $a$  and  $b$  [21]. This implies that there is an empty space, or margin, around the linear decision boundary of thickness  $1/\|w\|$ .

In turn, the optimization problem in Equation (2.3) is equivalent to

$$\begin{aligned} &\max_{w^T, b} \frac{1}{2} \|w\|^2 \\ &\text{subject to } f(x_i)(w^T \cdot x_i + b) \geq 1, \quad i = 1, \dots, N. \end{aligned} \tag{2.5}$$

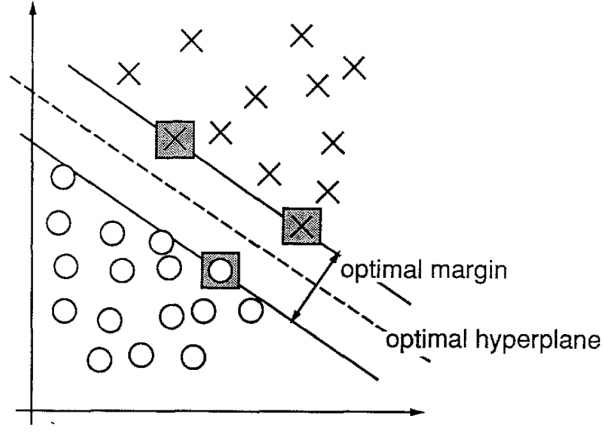


Figure 2.5: Optimal separation hyperplane, with parallel hyperplanes. The distance between the hyperplanes is the margin with the points in highlighted boxes lying along the hyperplanes being the support vectors.

Optimizing the geometric margin becomes a problem of minimizing the norm of the vector of weights in order to maximize the surrounding thickness [50]. Vapnik termed such hyperplanes as *canonical hyperplanes* [132]. The specific observations used for the optimization of the separating hyperplane that reside along  $H_1$  and  $H_2$  are known as *support vectors*, hence the name support vector machines.

Other points are discarded without altering the generalization performance of the classifier. In this way, SVM act as a prototype selection algorithm [21]. Figure (2.5), from [132], shows the geometric representation of the optimal margin and hyperplane for linearly separable data.

To simplify the optimization problem and facilitate the handling of conditions we transition to the dual form using the Lagrange formulation. The Lagrangian is given by,

$$L(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^N \alpha_i (f(x_i)(w^T \cdot x_i + b) - 1) \quad (2.6)$$

where  $\alpha_i$  are the Lagrange multipliers. Setting the derivative with respect to the primals  $w$  and  $b$  equal to zero and solving, we get

$$\begin{aligned}
\frac{\partial L(w, b, \alpha)}{\partial w} &= w - \sum_{i=1}^N \alpha_i f(x_i) x_i = 0 \\
\Rightarrow w &= \sum_{i=1}^N \alpha_i f(x_i) x_i, \text{ for all } i = 1, \dots, N
\end{aligned} \tag{2.7}$$

and

$$\begin{aligned}
\frac{\partial L(w, b, \alpha)}{\partial b} &= - \sum_{i=1}^N \alpha_i f(x_i) = 0 \\
\Rightarrow \sum_{i=1}^N \alpha_i f(x_i) &= 0, \text{ for all } i = 1, \dots, N.
\end{aligned} \tag{2.8}$$

Substituting the solutions back into Equation (2.6) gives the Wolfe dual form,

$$L(w, b, \alpha) = -\frac{1}{2} \sum_{i,j=1}^N \alpha_i \alpha_j f(x_i) f(x_j) \langle x_i, x_j \rangle + \sum_{i=1}^N \alpha_i. \tag{2.9}$$

The multipliers restrict the solution's search space to the set of feasible values under the constraint. The eventual solution maximizes the Wolfe dual under Karush-Kuhn-Tucker (KKT) conditions [5, 21]. The KKT conditions for all  $i = 1, \dots, N$  are

$$\begin{aligned}
-(f(x_i)(w^T \cdot x_i + b) - 1) &\leq 0 \\
\alpha_i &\geq 0 \\
\alpha_i(f(x_i)(w^T \cdot x_i + b) - 1) &= 0.
\end{aligned} \tag{2.10}$$



Thus, the Wolfe dual optimization becomes

$$\begin{aligned}
& \arg \max_{\alpha} \left[ -\frac{1}{2} \sum_{i,j=1}^N \alpha_i \alpha_j f(x_i) f(x_j) \langle x_i, x_j \rangle + \sum_{i=1}^N \alpha_i \right] \\
& \text{subject to } \sum_{i=1}^N \alpha_i f(x_i) = 0 \\
& \alpha_i \geq 0 \text{ for all } i = 1, \dots, N.
\end{aligned} \tag{2.11}$$

The dual constraints play a crucial role in ensuring optimality of the solution. At the optimal solution, the dual variables must be nonnegative, given that they are multiplied by a positive quantity. The presence of negative Lagrange multipliers would lead to a decrease in the value of the function, making them non-optimal. Support vectors have active or binding constraints, while other observations have multipliers of zero, and thus do not affect the solution. The optimal separating hyperplane produces a function  $z^T \hat{w} + \hat{b}$  such that the sign of the function determines the classification of a test observation  $z$ , that is  $\hat{f}(z) = \text{sign}\{z^T \hat{w} + \hat{b}\}$ .

### 2.3.2 Non-linearly Separable Case

In settings where complete separability of training data is not achievable, nonnegative slack variables, denoted as  $\xi_i \geq 0$ , are introduced into the SVM objective function. Slack allows errors in the classification process. Instead of perfect classification, SVM in the non-linearly separable case aims to identify a “soft margin” that classifies the majority of the data correctly which improves generalization. The problem in its primal form is a minimization of the objective function

$$\begin{aligned}
& \arg \min_{w, b, \xi} \left[ \frac{1}{2} \|w\|^2 + C \sum_{i=1}^N \xi_i \right] \\
& \text{subject to } f(x_i)(w^T \cdot x_i + b) \geq 1 - \xi_i \\
& \xi_i \geq 0, \text{ for all } i = 1, \dots, N.
\end{aligned} \tag{2.12}$$

where  $C$  is a prior selected *regularization term* that penalizes misclassification. Increasing  $C$  results in a tighter margin, putting greater emphasis on minimizing misclassification rate. Conversely, reducing  $C$  allows for more misclassifications due to an increase in margin size [132].

The solution is given by the saddle points of the Lagrangian primal function

$$\begin{aligned}
L(w, b, \xi, \alpha, \gamma) = & \frac{1}{2} \|w\|^2 + C \sum_{i=1}^N \xi_i \\
& - \sum_{i=1}^N \alpha_i [f(x_i)(w^T \cdot x_i + b) - (1 - \xi_i)] - \sum_{i=1}^N \gamma_i \xi_i
\end{aligned} \tag{2.13}$$

The Lagrangian is differentiated with respect to the primals and slack parameter. Setting the derivatives equal to zero and solving yields

$$\begin{aligned}
\frac{\partial L(w, b, \xi, \alpha, \gamma)}{\partial w} &= w - \sum_{i=1}^N \alpha_i f(x_i) x_i = 0 \\
\Rightarrow w &= \sum_{i=1}^N \alpha_i f(x_i) x_i \text{ for all } i = 1, \dots, N,
\end{aligned} \tag{2.14}$$

$$\begin{aligned}
\frac{\partial L(w, b, \xi, \alpha, \gamma)}{\partial b} &= -\sum_{i=1}^N \alpha_i f(x_i) = 0 \\
\Rightarrow \sum_{i=1}^N \alpha_i f(x_i) &= 0 \quad \text{for all } i = 1, \dots, N,
\end{aligned} \tag{2.15}$$

and

$$\begin{aligned}
\frac{\partial L(w, b, \xi, \alpha, \gamma)}{\partial \xi_i} &= C - \alpha_i - \gamma_i = 0 \\
\Rightarrow \alpha_i &= C - \gamma_i, \quad \text{for all } i = 1, \dots, N.
\end{aligned} \tag{2.16}$$

Optimization proceeds in the same way as in the linear separable case [132]. In Wolfe dual form, the soft margin SVM formulation is:

$$\begin{aligned}
&\arg \max_{\alpha} \left[ -\frac{1}{2} \sum_{i,j=1}^N \alpha_i \alpha_j f(x_i) f(x_j) \langle x_i, x_j \rangle + \sum_{i=1}^N \alpha_i \right] \\
&\text{subject to } \sum_{i=1}^N \alpha_i f(x_i) = 0, \\
&0 \leq \alpha_i \leq C, \quad \text{for all } i = 1, \dots, N.
\end{aligned} \tag{2.17}$$

Selection of optimal  $C$  is done using cross-validation techniques [21, 5, 50]. Support vectors, positioned on the margin imply  $\xi_i = 0$ , and the constraints hold with strict equalities while the residual slack is characterized by  $\alpha_i = C$ .

### 2.3.3 Kernel SVM

The classifier obtained from using a soft-margin SVM may not be robust for non-linearly separable data, even if the hyperplanes are optimally determined. To increase the separation between classes, the data is projected into a higher-dimensional space, called the *feature*

space, where the data is more linearly separable. The linear hyperplane in the feature space corresponds to a non-linear discriminant boundary when mapped back to the original input space.

Consider the mapping  $\Phi : \mathbb{R}^d \rightarrow \mathbb{R}^D$  where  $d \leq D$ . Classification of a new observation  $z$  with unknown label is done by first transforming to the feature space and then taking the sign of  $w^T \cdot \Phi(z) + b$ . As with soft-margin SVM, the vector  $w$  can be expressed as a linear combination of support vectors in the feature space [132].

The linearity of the inner-product implies that the classification function for new data depends solely on these inner products. Inner product calculations are done through kernel functions to save computational efficiency, denoted as  $K(u, v) = \langle \Phi(u), \Phi(v) \rangle$ . The key to the approach is to find a kernel function that can be evaluated efficiently. Kernels should be Hermitian and positive semi-definite matrices [5]. The kernel SVM classification dual becomes

$$\begin{aligned} & \arg \max_{\alpha} \left[ -\frac{1}{2} \sum_{i,j=1}^N \alpha_i \alpha_j f(x_i) f(x_j) K(x_i, x_j) + \sum_{i=1}^N \alpha_i \right] \\ & \text{subject to } \sum_{i=1}^N \alpha_i f(x_i) = 0, \\ & 0 \leq \alpha_i \leq C, \text{ for all } i = 1, \dots, N. \end{aligned} \tag{2.18}$$

Various kernels are employed in real-world applications, and there is currently no consensus on which kernel is universally superior for a specific application [50]. The most commonly used kernel is the Gaussian radial basis function (RBF) kernel

$$\beta(u, v) = \exp\{-||u - v||^2/2\gamma^2\} \tag{2.19}$$

where  $\gamma$  is a user-defined kernel parameter.

Even though *support vector data descriptions* are not in the same classification category as SVMs, our discussion of support vectors provides an opportunity to discuss them and their merits.

### 2.3.4 Support Vector Data Description

We briefly describe the problem of *data description* or *one-class classification* (OCC). The goal of OCC is to make a description of a set of objects and to detect which new objects resemble this set [86]. OCC delineates a boundary surrounding the target class data. This boundary aims to maximize the inclusion of observations from the target class while concurrently minimizing the probability of admitting non-target, or outlier, observations [59]. We describe the theoretical underpinnings of support vector data descriptions (SVDDs), but a more thorough discussion can be found in [128]. Let  $\mathcal{X}_j$  represent the target class for the one-class classification problem. SVDD defines a convex hypersphere characterized by center  $a \in \mathcal{X}_j$  and radius  $R > 0$  which gives a closed boundary around the target class data  $\mathcal{X}_j$  such that  $|\mathcal{X}_j| = n$ . We construct the hypersphere in such a way to minimize the volume while observations  $x_i \in \mathcal{X}_j$  are within the sphere. Analogous to SVM, a slack variable  $\xi_i$  is introduced which penalizes larger volumes for descriptions. The minimization problem is set up such that

$$\begin{aligned} \arg \min_R \text{Vol}(R, a) &= R + C \sum_{i=1}^n \xi_i \\ \text{subject to } ||x_i - a||^2 &\leq R + \xi_i, \\ \xi_i &\geq 0 \text{ for all } i = 1, \dots, n. \end{aligned} \tag{2.20}$$

The variable  $C$  controls the trade-off between penalizing a larger radius in exchange for some points being excluded from the description [128]. Optimization follows in a similar way to

SVMs. Rewriting the optimization problem in terms of Lagrange multipliers, gives us

$$\begin{aligned}
L(R, a, \xi, \alpha, \gamma) = R + C \sum_i \xi_i \\
- \sum_i \alpha_i [R + \xi_i - (\|x_i\|^2 - 2a \cdot x_i + \|a\|^2)] - \sum_i \gamma_i \xi_i.
\end{aligned} \tag{2.21}$$

$L$  should be minimized with respect to  $R, a, \xi$  and maximized with respect to the multipliers  $\alpha, \gamma$ . Taking partial derivatives, setting them equal to zero, and solving yields

$$\begin{aligned}
\frac{\partial L(R, a, \xi, \alpha, \gamma)}{\partial R} = 0 &\Rightarrow \sum_i \alpha_i = 1 \\
\frac{\partial L(R, a, \xi, \alpha, \gamma)}{\partial a} = 0 &\Rightarrow a = \sum_i \alpha_i x_i \\
\frac{\partial L(R, a, \xi, \alpha, \gamma)}{\partial \xi} = 0 &\Rightarrow \alpha_i = C - \gamma_i, \text{ for all } i = 1, \dots, n.
\end{aligned} \tag{2.22}$$

The KKT conditions and constraints of Lagrange multipliers implies that  $0 \leq \alpha_i \leq C$  for all  $i = 1, \dots, n$ . Substitution back into the original Lagrangian gives

$$L = \sum_{i=1}^n \alpha_i \langle x_i, x_i \rangle - \sum_{i,j=1}^n \alpha_i \alpha_j \langle x_i, x_j \rangle \text{ s.t. } 0 \leq \alpha_i \leq C \text{ for all } i = 1, \dots, n. \tag{2.23}$$

When  $x_i$  satisfies the inequality  $\|x_i - a\|^2 < R + \xi_i$ , the constraint is satisfied and  $\alpha_i = 0$ . Conversely, if  $\|x_i - a\|^2 = R + \xi_i$  then the constraint is enforced requiring  $\alpha_i > 0$ . Only observations  $x_i$  with  $\alpha_i > 0$  are needed in the description and these objects will therefore act as the *support vectors of the description*.

A query point is accepted in the description if the distance to the center of the sphere is smaller than  $R$ , since by definition  $R$  is the Euclidean distance from the center of the sphere to any of the support vectors on the boundary. Similarly to SVM, the inner products can be substituted with a kernel function  $K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$  for a more adaptable, though not necessarily convex, description [128]. An optimal kernel function would project

the target data onto a bounded, spherically shaped region within the feature space, with potential outlier observations outside this region.

SVDDs are commonly used in machine learning to estimate regions of class overlap. By constructing separate descriptions for each class and analyzing the intersection of their boundaries, SVDDs identify ambiguous regions where classification might be challenging due to overlapping class observations [76, 89]. We now continue our discussion on binary and multi-class classification models used for comparison in Chapters 3-5 with *decision trees*.

## 2.4 Decision Trees

Decision trees are undirected graph structures where any pair of vertices are linked by precisely one path along the edges [85]. Trees are composed of two elements: *nodes* and *branches* [17]. A more comprehensive review of alternative tree-based methods can be found in [28]. Each tree is initiated at the *root* or *decision node*. Nodes are typically *split* by some partitioning of the input space. Continuous numerical variables are partitioned into two disjoint spaces using inequalities, while discrete variables split by the factor levels. Nodes being split are called *parent nodes*, while the resultant nodes from the split are *children nodes*. The decision rule is a series of splits, with the final node encapsulating the classification decision on the *leaf node* [123]. Since each decision rule is a clearly defined list of partitions, decision trees are well-established to be *interpretable* machine learners [125].

Creating a tree is an inductive optimization problem with the objective being a partition that accurately predicts the target’s label while utilizing the fewest number of partitions possible. It is solved greedily through a variables’ *informativeness*, or the amount of homogeneous labels in each partition [28]. Typically, this homogeneity is measured by the *Gini Index*. That is, we pick splits that minimize,  $\sum_{k=1}^K (\hat{p}_{mk})(1 - \hat{p}_{mk})$  where  $K$  is the number of classes and  $\hat{p}_{mk}$  is the proportion of observations of class  $k$  at the specific node  $m$  [17, 50].

The determination of the optimal tree depth is an ongoing area of research [4]. A highly complex model may lack interpretability, while an overly simplistic model may struggle

to reasonably generalize. Current approaches *prune* a large tree by removing splits to an optimal size. Pruning can be categorized into two types: *pre-pruning*, which prevents the generation of nonsignificant branches, and *post-pruning*, which involves removing branches from a fully grown tree in a way that improves the overall classification accuracy when applied to a validation data set [123].

#### 2.4.1 C4.5 and C5.0 Algorithms

C4.5 is a decision tree algorithm that has gained popularity in the recent literature. C4.5 uses a normalized information gain rate based on system *entropy* instead of Gini Index [102]. Information gain is defined as the reduction of Shannon's entropy produced by partitioning a set with node split,  $a$ . In this way, the information gained is as follows

$$\text{Information Gain}(\mathcal{Y}, a) = H(\mathcal{Y}) - H(\mathcal{Y}|a). \quad (2.24)$$

where  $H(\mathcal{Y}|a)$  is the conditional Shannon's entropy based on node split  $a$ . The process is used to remove bias in the information gain ratio calculation where, for specific label  $y_i \in \mathcal{Y}$ , the split information is calculated by

$$\text{Split Information}(\mathcal{Y}) = - \sum_{i=1}^K \frac{N(y_i)}{N(\mathcal{Y})} \log_2 \left( \frac{N(y_i)}{N(\mathcal{Y})} \right) \quad (2.25)$$

with  $N(y_i)$  being the number of times label  $y_i$  occurs and  $N(\mathcal{Y})$  is the total number of labels possible in the event. The information gain rate is calculated by

$$\text{Information Gain Rate}(\mathcal{Y}, a) = \frac{\text{Information Gain}(\mathcal{Y}, a)}{\text{Split Information}(\mathcal{Y})}. \quad (2.26)$$

Entropy decreases with splitting as it implies the node is divided into its purest constituents. C4.5 cannot handle missing values in the dataset directly, and instead excludes them [123].



*Surrogate splitting* handles missing values indirectly by imputation using other available attributes.

C5.0 builds upon the foundation of C4.5 by addressing certain limitations and introducing advancements in handling numeric attributes, missing values, rule pruning, and scalability [103]. More detailed insights into C5.0 can be found in [62]. C5.0 enhances generalization by incorporating unequal misclassification costs during the tree-building process. The weights for misclassified values are dynamically updated at each iteration and utilized in subsequent classification rules. C5.0 uses *winnowing* for numerical inputs. Winnowing selects a small subset of potential split points, ensuring higher efficiency and accuracy compared to C4.5.

C5.0 trees tend to be smaller than their C4.5 counterparts due to its advanced pruning algorithm [102]. In terms of efficiency and scalability, C5.0 is generally considered superior to C4.5, featuring optimizations that accelerate tree construction and facilitate the handling of larger datasets with more efficient memory usage.

## 2.5 Class Cover Catch Digraphs

CCCDs are a digraph  $D = (\mathcal{V}, \mathcal{A})$  composed of a vertex set  $\mathcal{X}$  and an arc set  $\mathcal{A}$ . The vertex set is equal to the points from the target class being covered,  $\mathcal{V} = \mathcal{X}_j$ . The cover is constructed by considering a covering ball  $B_i = B(x_i, r_i)$  of radius  $r_i = r(x_i)$  with center of each located at point  $x_i \in \mathcal{X}_j$ . Families of CCCD are determined by the calculation of the radius of the covering balls. There are currently two major families of class cover catch digraphs, the *pure class cover catch digraphs* (P-CCCDs) and the *random-walk class cover catch digraphs* (RW-CCCDs). The arc set  $\mathcal{A}$  is defined by the relationship  $(x_i, x_j) \in \mathcal{A} \iff x_j \in B_i$ . That is, we build an arc from point  $x_i$  to  $x_j$  if and only if  $x_j$  is contained in the covering ball centered around  $x_i$ . In such a case, we say that  $x_i$  “catches”  $x_j$ . Arcs, instead of undirected edges, are necessary as the binary relation  $x_i \sim x_j$  is not symmetric, and preserves more information.

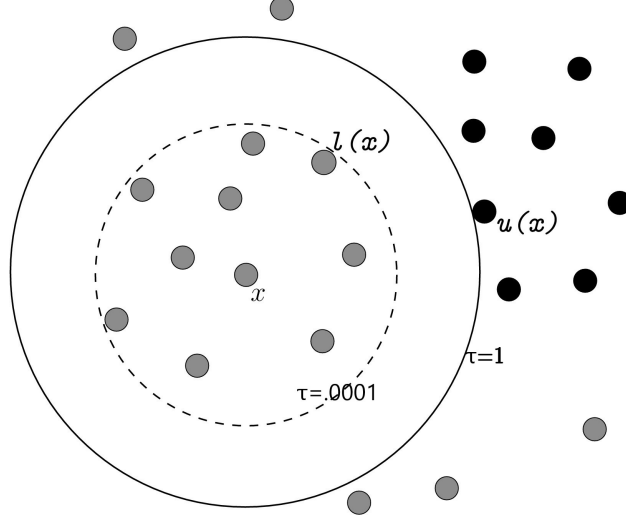


Figure 2.6: Two radii, one with  $\tau = .0001$  for the dotted circle and  $\tau = 1$  for the solid circle, for the metric covering ball centered at  $x$ . The gray and black points represent the target and non-target class, respectively.

As a prototype learning method, CCCD classifiers aims to find a subset of open covering balls  $B_x = \{z \in \mathbb{R}^d : d(x_i, z) < r(x_i)\}$ , denoted  $\mathcal{C}_{\mathcal{X}_j} \subseteq \mathcal{B}_{\mathcal{X}_j} = \{B_1, \dots, B_n\}$ , such that for some prototype set of the target class,  $\mathcal{Q}_{\mathcal{X}_j} \subseteq \mathcal{X}$ , we have  $\mathcal{Q}_{\mathcal{X}_j} \subseteq \bigcup_{B \in \mathcal{C}_{\mathcal{X}_j}} B$  and  $\mathcal{C}_{\mathcal{X}_j} = \bigcup_{B \in \mathcal{C}_{\mathcal{X}_j}} B$  is the cover of  $\mathcal{Q}_{\mathcal{X}_j}$ . Covering of non-target class  $\mathcal{X}_{1-j}$  reverses the roles of the classes.  $\mathcal{X}_{1-j}$  becomes the new target class with  $\mathcal{X}_j$  being the new non-target.

Define the *open neighborhood* of a vertex in a digraph  $s \in \mathcal{V}$  as  $N(s) = \{t \in \mathcal{V} : (s, t) \in A\}$ , that is, the set of vertices caught by  $s$ . A *dominating set* of a digraph  $D$  is defined as a subset of vertices  $S \subseteq \mathcal{V}$  such that the union of the closed neighborhoods defined by  $\overline{N(s)} = N(s) \cup \{s\}$  of elements  $S$  is the set of vertices of the digraph:  $\bigcup_{s \in S} \overline{N(s)} = \mathcal{V}$ . A dominating set with minimal cardinality is called a *minimum dominating set* (MDS), and this cardinality is called the *domination number*,  $\gamma(D)$ . Finding MDS is a known NP-hard optimization problem, so an approximation algorithm is used that is greedy in the number of out degrees for each vertex [25]. Full details for the greedy algorithm are found in [138].

### 2.5.1 Classification with P-CCCDs

To ensure that a cover is *pure*, i.e., without non-target points, for every observation  $x_i \in \mathcal{X}_j$  the radius  $r(x_i)$  of the open covering ball  $B_i$  must be bounded above by the distance between  $x_i$  and the nearest non-target point  $x'_i$ . Given hyperparameter  $\tau \in (0, 1]$ , we define the radius  $r(x_i)$  by

$$r(x_i) = (1 - \tau)d(x_i, l(x'_i)) + \tau d(x_i, u(x'_i)), \quad (2.27)$$

where

$$u(x'_i) = \arg \min_{x'_i \in \mathcal{X}_{1-j}} d(x_i, x'_i) \quad (2.28)$$

and

$$l(x'_i) = \arg \max_{x_i \in \mathcal{X}_j} \{d(x_i, x'_i) \text{ s.t. } d(x_i, x'_i) < d(x_i, u(x'_i))\}. \quad (2.29)$$

Any target class cover with such radii is guaranteed to be pure by the definition of Equation (2.27) as this definition necessarily means that all non-target points are kept out of the open covering balls centered at each  $x_i \in \mathcal{X}_j$ . Figure (2.6) gives an example of the effects of  $\tau$  on the radii of the covering ball [76]. Changing  $\tau$  changes the volume of the cover, affecting classification performance, but does not change the structure of the digraph [78, 76]. After covering all target and non-target points, the corresponding digraph  $D$  can be constructed.

Solving the CCP is equivalent to finding a MDS,  $S \subseteq \mathcal{V}$ . A *proper* cover covers all points in the target-class, hence by construction, the covers established by P-CCCDs are proper covers by setting  $\mathcal{Q}_{\mathcal{X}_j}$  to the MDS of the target class and  $\mathcal{Q}_{\mathcal{X}_{1-j}}$  to the MDS of the non-target class. Theoretical results on the computational complexity are detailed more thoroughly in [76].

After establishing both class covers, any new point,  $z$ , can be classified by the location of  $z$  with respect to the class that  $z$  is most similar [100, 76].

**Definition 2.5.1.** Mapping  $\rho : \Omega \times \Omega \rightarrow \mathbb{R}^+$  is a *dissimilarity measure* if necessarily for any two observations  $x, y \in \Omega \times \Omega$ ,  $0 \leq \rho(x, y) = \rho(y, x) < \infty$  and  $\rho(x, y) = 0$  if and only if  $x = y$ .

We define

$$\rho(z, x) = \frac{d(z, x)}{r(x)} \quad (2.30)$$

where  $\rho$  acts pseudo-dissimilarity measure, since it violates the symmetry axiom.  $\rho$  satisfies continuity conditions [98]. The estimated class of a given point  $z$  is determined by

$$\hat{f}(z) = \arg \min_{U \in \{\mathcal{X}_0, \mathcal{X}_1\}} \left[ \min_{x: B(x, r) \in \mathcal{C}_U} \rho(z, x) \right]. \quad (2.31)$$

If two points have the same size radius in terms of  $\rho$ , classification is done by arbitrary assignment. P-CCCDs were shown to be a *consistent* classifier under strictly separable class supports [100].

### 2.5.2 Classification with RW-CCCDs

The strategy to restrict the distance of the covering balls to the closest non-target class point as in P-CCCDs can tend to overfit the training data [32, 76]. Instead, an adaptive family of CCCD, called random walk class cover catch digraph (RW-CCCD), is used. RW-CCCD adjust the radii to allow certain neighboring non-target class points to lie within the cover, while some target class points may remain outside, as the optimal radius around these points may reduce to zero [33]. We define

$$\begin{aligned} R_{x_i}(r) &= R_{x_i}(r; \mathcal{X}_j, \mathcal{X}_{1-j}) \\ &= \frac{m}{n} |\{x_j \in \mathcal{X}_j : d(x_i, x_j) \leq r\}| - |\{x'_j \in \mathcal{X}_{1-j} : d(x_i, x'_j) \leq r\}| \end{aligned} \quad (2.32)$$

where  $|\{x_j \in \mathcal{X}_j : d(x_i, x_j) \leq r\}|$  is the number of target points in the current radius and  $|\{x'_j \in \mathcal{X}_{1-j} : d(x_i, x'_j) \leq r\}|$  is the number of non-target points in the current radius.

The name RW-CCCD is a result from the function  $R_{x_i}(r)$  being viewed as a random process resembling a one-dimensional random walk as demonstrated in Figure (2.7). The ratio  $m/n$  mitigates bias from class imbalance [33]. The radius is calculated by,

$$r_{x_i} = \arg \max_{r \in \{d(x_i, x_j) : x_j \in \mathcal{X}_j \cup \mathcal{X}_{1-j}\}} R_{x_i}(r) - P_{x_i}(r) \quad (2.33)$$

where  $P_{x_i}(r)$  is a penalty function so as to discourage covering balls with large radii, though in practice  $P_{x_i}(r) = 0$  is commonly used [33, 32]. RW-CCCDs yield closed covering balls,  $\overline{B}_{x_i} = \{x_j \in \mathbb{R}^d : d(x_i, x_j) \leq r(x_i)\}$ . Larger values of  $R_{x_i}(r)$  signify a higher local density of points belonging to the same class around  $x_i$  compared to the local density of points from other classes [120]. RW-CCCD have better generalization by allowing some points of  $\mathcal{X}_{1-j}$  inside  $\mathcal{C}_{\mathcal{X}_j}$  to mitigate overfitting [33, 76].

Finding the MDS for RW-CCCD is a modified extension of the CCP, as the composition of covering ball sets changes after adding a point to the dominating set. Each target class point  $x_i \in \mathcal{X}_j$  is given a score,  $T_{x_i}$  associated with  $R_{x_i}(r_{x_i})$ . The score function is given by

$$T_{x_i} = R_{x_i}(r_{x_i}) - \frac{r_{x_i} n_u}{2d_m(x_i)} \quad (2.34)$$

where  $n_u$  is the number of uncovered target class points in the current iteration, and  $d_m(x_i) = \max_{x_j \in \mathcal{X}_j} d(x_i, x_j)$  such that  $i \neq j$ . The term  $r_{x_i} n_u / (2d_m(x_i))$  penalizes scores corresponding to larger radii than for covering balls of smaller radii. This penalization is likely to choose dominating points with radius  $r = 0$ , which only dominate themselves and thus are not considered covered. Hence, RW-CCCD covers also can potentially be improper covers. An algorithm greedy on a score function is used to find the MDS. Despite the bias towards smaller radii, empirical studies showed that the lack of severity of the penalization means that RW-CCCD produces dominating sets with smaller sizes compared to P-CCCD in some

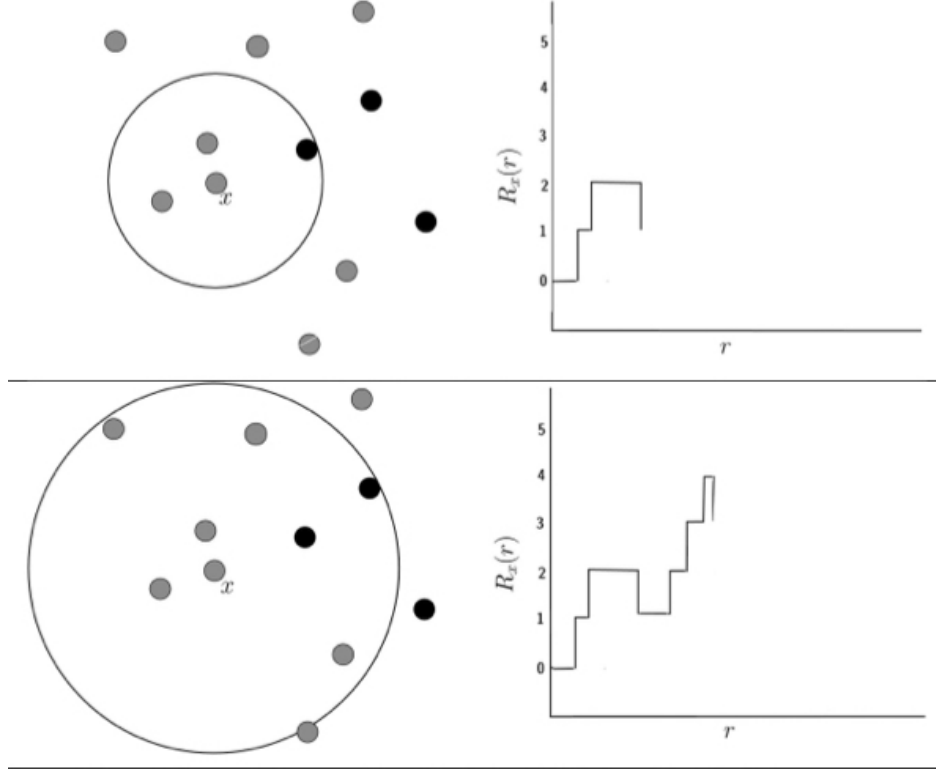


Figure 2.7: A snapshot of  $R_x(r)$  at two different radii for the covering ball  $B_x$  centered at  $x$ .

cases [76]. We reference [33, 76] for more details on the RW-CCCD greedy domination algorithm.

Finding an MDS of RW-CCCD is slower in computational time than P-CCCD counterparts. Classification in RW-CCCD is done by modifying Equation (2.31) to include the score function [32]. The class of a new point  $z$  is determined by

$$\hat{f}(z) = \arg \min_{U \in \{\mathcal{X}_j, \mathcal{X}_{1-j}\}} \left[ \min_{x: B(x, r) \in \mathcal{C}_U} \rho(z, x)^{T_x^e} \right]. \quad (2.35)$$

where  $\rho(z, x)$  is defined as in Equation (2.30) and  $e \in [0, 1]$ . Here,  $e$  acts as a hyperparameter to control at what level the score  $T_x$  is incorporated. When  $e = 0$ , the scores are not used in the classification decision, with  $e = 1$  meaning they are fully incorporated. Higher incorporation of scores improves performance of RW-CCCD classifiers with increasing target class size and decreasing dimensionality, as the radii more effectively capture local density

around prototype points in these settings [76]. If  $z \in \mathcal{C}_{\mathcal{X}_j} \cap \mathcal{C}_{\mathcal{X}_{1-j}}$ , the score acts as a good indicator to the likelihood of the covering ball containing more target class points, and thus is a good indicator for classifying the new point.

### 2.5.3 Multi-Class Classification

CCCDs are generalized for problems with  $K$ -classes by transformation of the multi-class classification problem into a series of binary class problems, or a *one-versus-rest approach*. In the one-versus-rest approach, to establish the set of covers  $\mathcal{C} = \{C_1, C_2, \dots, C_K\}$  associated with data  $\mathcal{X} = \{\mathcal{X}_1, \mathcal{X}_2, \dots, \mathcal{X}_K\}$ , we define  $\mathcal{X}_T = \mathcal{X}_i$  for one of  $i = 1, \dots, K$  and  $\mathcal{X}_{NT} = \bigcup_{i \neq j} \mathcal{X}_j \setminus \mathcal{X}_T$ . In this way, we take classes  $\mathcal{X}_T$  and  $\mathcal{X}_{NT}$  to be the target and non-target classes, respectively. The cover for the target class  $C_i$  is found in the same way as the two-class setting and then repeated until covers for all  $i = 1, \dots, K$  have been found.

With the baseline classifier families established, we now examine common algorithmic modifications designed to improve performance under challenging *data intrinsic characteristics* such as class imbalance and class overlap. Standard classifiers such as  $k$ -NN, SVM, and decision trees are built under the assumption of “well-behaved” data and thus often exhibit sensitivity to both class imbalance and overlap. This can lead to biased decision boundaries or reduced generalization [43, 46, 21, 11, 26]. Such modifications are therefore necessary to adjust their inductive biases, enhance robustness in ambiguous regions, and ensure fair representation of minority class instances.

## 2.6 Algorithmic Modifications for Complex Data Intrinsic Characteristics

The *class imbalance problem* refers to classification under imbalanced classes, and has been thoroughly studied [107, 6, 57, 66, 124]. *Class imbalance* refers to the disproportion in sizes between different class observations in data space, and is characterized either *globally* or confined *locally* to a region. Class imbalance, however, does not implicitly align with

classification complexity as even if the domain is highly imbalanced classification models can still have high performances [72].

Instead, the *class overlap problem* has been characterized as the biggest impacting data characteristic for the classification problem [136, 141, 67]. *Class overlap* exists in data if the different classes occupy the same region within the data space. Despite being labeled differently, these samples exhibit similar characteristics, making distinguishing their decision boundaries difficult [122]. The class overlap problem becomes much harder when combined with the class imbalance problem [109, 136]. Solving the classification problem for data with both imbalanced and overlapped intrinsic characteristics is an area of ongoing study [31, 134, 64, 141].

Algorithmic modifications are used to mitigate the effects of both class imbalance and overlap. Modifications are either

- (i) at the data level which involves a pre-processing of the data set being used in training, or
- (ii) at the algorithmic level where classifiers are modified to make the decision rule robust to the imbalances and / or overlap.

Many contemporary algorithms combine aspects of both types of modifications, with primary modifications being either resampling, ensemble learning, cost-sensitive methods, and hybrid classifiers or a combination of them [10, 27, 67, 31, 52, 57, 109, 52, 64]

### 2.6.1 Resampling Methods

Resampling methods either under- or over-sample as a way to balance the data set, and thus modify classifiers only at the *data level*. They are regarded as the simplest method to deal with class imbalance and class overlap problems [7, 6]. *Undersampling* is the process of removing or filtering points from the majority class to address data imbalance [7]. CC-CDs resemble undersampling techniques by markedly reducing the number of majority-class



observations. In both families of directed catch digraphs, covering balls that encompass the majority points tend to be larger than those of the minority class [76]. Given the larger membership of the majority class, these larger covering balls are more likely to encapsulate, and thus delete, neighboring majority points.

CCCDs differ from other undersampling methods by preserving local density information through radius retention. Larger radii denote a greater influence of a prototype over the domain. In this way, the score function for RW-CCCD serves as an additional cluster statistic, strategically utilized to yield classifiers with enhanced performance, greater data reduction, and less information loss.

However, certain scenarios may arise where retaining all points from both classes is essential, regardless of the variations in class sizes, resulting in a loss of information. *Over-sampling* increases the size of the minority class using artificial points to balance the class distributions. *Synthetic minority over-sampling technique* (SMOTE) has emerged as a seminal oversampling algorithm for addressing class imbalance in machine learning datasets [23, 44]. SMOTE operates by identifying minority class points and synthesizing new examples along the line segments between the identified point and k-nearest neighbors [23].

The number of generated synthetic observations is determined by a user-defined oversampling ratio. The additional synthetic points balance the two classes to facilitate a better understanding of the boundary region between classes [10, 83]. SMOTE has found widespread application in various domains where class imbalance poses a challenge, but there are limitations. SMOTE has the potential to introduce noise, especially when synthetic data are generated in regions of the input space that do not accurately represent the underlying distribution of the minority class [10]. The effect is more dramatic in higher dimensional data. Moreover, the synthetic data produced by SMOTE may potentially cause overfitting for some model techniques, particularly if the synthetic data is close to existing minority class data [52].

*Edited nearest neighbor* (ENN) seeks to refine datasets by identifying potentially misclassified or noisy observations. The algorithm determines  $k$ -nearest neighbors for each point using a specified distance metric and compares the class labels. Observations displaying *discordant* labels with the majority of their neighbors are deemed potential outliers and are thus removed from the dataset [139]. ENN has demonstrated effectiveness only in certain scenarios, however [7]. Often, SMOTE and ENN are combined to achieve class balance by creating synthetic points between data of the minority class and then using ENN to refining the dataset, thereby improving the classification performance [10].

Variations of resampling methods that target overlapping regions have been proposed to handle class overlap [142, 101, 110]. *Overlap-based methods* such as Borderline-SMOTE resample only the minority class observations near the borderline between classes [49]. *Hybrid sampling* combines different sampling techniques to optimize the selection of samples based on both time and space factors. Recent developments use evolutionary algorithms to identify majority class instances in overlapping regions and remove them [142].

## 2.6.2 Ensemble Frameworks

Ensemble learning has become a “catchall” term for methods that combine multiple models to make a decision. The information captured from a single classifier is less than the information from combined multiple models, and so the errors from a single classifier are corrected, leading to an overall performance increase. Ensemble methods themselves are not limited to a specific machine learning algorithm [108]. We can divide ensembling into two frameworks, the *independent framework* which includes *bagging*, *dagging*, and *stacking*, and the *dependent framework* which includes *boosting*.

### Independent Frameworks

As the name suggests, independent ensemble frameworks train models independently (and often in parallel), using a voting scheme to make the final prediction [61, 108]. *Bagging*,

---

**Algorithm 1** Classification with bagged classifier

---

**Input:** Query point,  $z$ , Classifier,  $\hat{f}$ , Training data,  $\mathcal{X}$ , Number of replicates  $L$

**Output:** Class labeling decision,  $\hat{f}_B(z)$

Initialize an empty ensemble of base classifiers:  $\hat{f}_l = []$

**for**  $l = 1$  to  $L$  **do**

    Bootstrap sample  $\mathcal{B}_l$  from  $\mathcal{X}$  such that  $\mathcal{B}_l \subseteq \mathcal{X}$ .

$\hat{f}_l = \hat{f}_l \cup \hat{f}(z|\mathcal{B}_l)$

    Record classification label  $\hat{f}_l(z)$

**end for**

$\hat{f}_B(z) \leftarrow$  aggregated decision across all  $\hat{f}_l(z)$

---

shorthand for “bootstrap aggregating”, uses bootstrap resampling to create the training data for the models. Bootstrap sampling introduces diversity to the training data and subsequent trained models. Theoretically, the distribution of bootstrapped samples should approximate the initial data distribution as the number of iterations increases [16, 40].

It is worth mentioning other popularized independent ensemble methods, *stacking* and *dagging*, even if our focus will be primarily on bagging. Stacking is another form of independent ensemble framework that uses different classifiers instead of resampled training data to introduce diversity, while dagging uses a smaller collection of disjoint samples from training data [129, 61].

A generic algorithm for bagging is presented in Algorithm (1). We can think of  $\hat{f}(z|\mathcal{B})$  to represent the classification decision rule of the query point  $z$  given a bootstrap sample from the original training data,  $\mathcal{B} \subseteq \mathcal{X}$ . Our goal is to build learning sets  $\{\mathcal{B}_l\}$  for  $l = 1, \dots, L$  number of independent replicates such that the sequence  $\{\hat{f}(z|\mathcal{B}_l)\}_{l=1,\dots,L}$  performs at least as well as, if not better than,  $\hat{f}(z|\mathcal{X})$ .

Aggregation is typically done by majority voting, meaning the final decision label is the mode label across the ensemble of all  $\hat{f}_l(z)$ . However, alternative aggregation methods, such as averaging, are available for methods that predict class labels using probabilities [16, 121]. Bagging is particularly effective in reducing variance and overfitting [16]. The stability of the underlying classifier determines performance of bagged classifier  $\hat{f}_B(z)$ ; meaning small changes in the learning set should produce only small changes in the classifier itself. Empirical

results show, with very minimal assumptions, that bagging is able to stabilize even highly variable models [121].

## Dependent Frameworks

A *weak classifier* in the context of the class imbalance problem refers to a model that performs slightly better than random guessing. Boosting is an ensemble method that emphasizes the creation of a *strong classifiers* by dependently and iteratively combining weak classifiers. Subsequent iterations are trained with more emphasis given to prior misclassified observations. The direct result is a new classifier that will be significantly different from its predecessors [113]. The final decision rule is the weighted aggregation of the predictions of all classifiers. Emphasis is given by incorporating a weighting schema either in the original classifier or in resampling of the training data. Boosting has shown remarkable performance in areas such as face detection and text classification [120, 14].

The main variants of boosting are *GradientBoost*, *XGBoost*, and *AdaBoost*. XGBoost is a distributive variant of GradientBoost. Training is based on the gradient of a loss function to help weighting in future iterations. Although both methods were originally developed for regression tasks, cross-entropy loss functions have allowed them to work for classification tasks as well [112].

Adaptive boosting, known as AdaBoost, is the most well-known classification boosting algorithm, and will be the primary focus of this work. For AdaBoost, we redefine class labels as  $\mathcal{Y} = \{-1, 1\}$  instead of the standard  $\{0, 1\}$ . The *base learner* can be theoretically any classifier, but in practice is typically a weak classifier. This may seem counterintuitive, however, weaker classifiers are able to create a more diverse ensemble and thus generalize better [113]. To determine the training data fed to the base learner in each round, AdaBoost manages a distribution,  $D$ , across the training examples. The  $l$ -th round distribution is indicated by  $D_l$  with the weights associated with the observation  $x_i \in \mathcal{X}$  denoted  $D_l(x_i)$ . Weighting acts as a measure of the importance of correctly classifying observation  $x_i$  on

---

**Algorithm 2** AdaBoost Ensemble Classifier

---

**Input:** Query point  $z$ , base learner  $\hat{f}$ , training data  $\mathcal{X} = \{(x_i, f(x_i))\}_{i=1}^N$  with classes  $\{-1, 1\}$ , number of rounds  $L$

**Output:** Final classifier  $\hat{f}(z)$

Initialize weights:  $D_1(x_i) \leftarrow 1/N$  for all  $i$

**for**  $l = 1$  to  $L$  **do**

Train  $\hat{f}_l$  using distribution  $D_l$

Compute error:  $\epsilon_l \leftarrow \Pr_{i \sim D_l}(\hat{f}_l(x_i) \neq f(x_i))$

Compute weight:  $\alpha_l \leftarrow \frac{1}{2} \ln \left( \frac{1-\epsilon_l}{\epsilon_l} \right)$

Update distribution:

$$D_{l+1}(x_i) \leftarrow \frac{D_l(x_i) \exp \left( -\alpha_l f(x_i) \hat{f}_l(x_i) \right)}{Z_l}$$

where  $Z_l$  normalizes  $D_{l+1}$

**end for**

Return:  $H(z) \leftarrow \text{sign} \left( \sum_{l=1}^L \alpha_l \hat{f}_l(z) \right)$

---

current boosting round  $l$ . Harder to classify observations receive successively higher weights in future iterations to ensure more focus during training. If the base learner does not use weighting to determine its decision rule,  $D_l$  is instead used in stochastically selecting training data so that misclassified points are more likely to appear in future training rounds. *EasyEnsemble* variants use balanced bootstrap samples to train AdaBoosted classifiers [70].

### 2.6.3 Cost-Sensitive Methods

Cost-sensitive learning methods assign a (not necessarily monetary) cost to each class [37]. In applications, it is common for the minority class to have a higher misclassification cost than the majority class [42]. We define binary labels  $\mathcal{Y} = \{+, -\}$  for each class, letting “+” represent the minority class while “−” represents the majority class. Then  $C(\cdot, \cdot)$  is the associated cost of classifying point. In the binary setting, the cost matrix is defined as in Table 2.1 [88].

The purpose of cost-sensitive learning is to build a model that minimizes the total cost. Since correct classification does not incur a cost, we can define total cost as

|              |          | Predicted Class |           |
|--------------|----------|-----------------|-----------|
|              |          | Positive        | Negative  |
| Actual Class | Positive | $C(+, +)$       | $C(+, -)$ |
|              | Negative | $C(-, +)$       | $C(-, -)$ |

Table 2.1: Cost matrix for binary classification settings.

$$\text{TotalCost} = C(+, -) \times FN + C(-, +) \times FP \quad (2.36)$$

where  $FN$  is the number of *false negatives* and  $FP$  is the number of *false positives*.

Overlap-sensitive cost functions are designed to explicitly account for different weighting of majority and minority classes in regions of class overlap [109]. A higher costs is assigned to misclassifications that occur in regions of low overlap where decision boundaries should, in theory, be more reliable. Conversely, lower costs are assigned in high overlap regions. These cost functions aim to reduce penalization for inherently hard-to-classify observations while encouraging accurate classification where classes are separated [118]. Additionally, incorporating overlap-sensitive weighting schemes into loss functions has shown promising results [117].

#### 2.6.4 Hybrid Classification Methods

For class overlap, projection of the original data using kernel functions to a higher dimensional feature space can result in more linearly separable classes. However, there is no certified guarantee for classes to be separable even in higher dimensions [67]. Instead, many classifiers use *hybrid classification* approaches, where the decision rule is different for regions of class overlap versus non-overlap. Identifying class overlap remains an active area of research in the development of hybrid classification methods. Common approaches to identify the region of overlap include previously discussed SVDD, probabilistic neural networks with soft-decision strategies, and or the use of fuzzy set logic [141, 127, 67, 11].

We now move to close this chapter with a brief description of different performance metrics for evaluating classifiers. Traditionally, in balanced settings, the predictive ability of the classification algorithm is typically measured by its *predictive accuracy* on the testing data. However, when classes are imbalanced, models can still record high accuracies by overestimating the majority class [54]. Comparisons for class imbalance require performance to be estimated in a way that accounts for the minority class representation.

## 2.7 Metrics for Measuring Performance Under Data Intrinsic Characteristics

No single unifying metric has been found for the evaluation of classifiers under difficult data intrinsic characteristics. For this reason, we have adopted using a collection of area under the curve,  $F_1$ -score, and geometric mean for the evaluation across all model performances. The combination provides a more comprehensive assessment of classifier performance by capturing different aspects of predictive quality. We use the following terminology in the definitions of the chosen performance metrics. The *true positive rate* (also known as *sensitivity* or *recall*) measures the proportion of actual positive cases that the model correctly identifies

$$\text{Sensitivity} = \frac{TP}{TP + FN} \quad (2.37)$$

where  $TP$  is the number of *true positives*, and  $FN$  is the number of *false negatives*. Similarly, the *false positive rate* (FPR) quantifies the proportion of actual negative cases that are incorrectly classified as positive

$$\text{FPR} = \frac{FP}{FP + TN} \quad (2.38)$$

where  $FP$  are the *false positives* and  $TN$  are the *true negatives*. *Precision* measures the proportion of predicted positive cases that are actually positive

$$\text{Precision} = \frac{TP}{TP + FP}. \quad (2.39)$$

The *specificity* of a model is the proportion of actual negative cases that are correctly classified

$$\text{Specificity} = \frac{TN}{TN + FP}. \quad (2.40)$$

### 2.7.1 Area Under the Curve

The *receiver operating characteristics* (ROC) curve was first used in signal detection theory to represent the tradeoff between the hit rates and false alarm rates [41]. The ROC curve is the plot of the sensitivity against the false positive rate across various probability values that determine how a classification model assigns data points to a positive or negative class, or *thresholds*. ROC curves help visualize a model's trade-offs between sensitivity and specificity. A curve close to the diagonal line, (i.e.,  $TP \approx FP$ ) indicates that the model is not very good at distinguishing between positive and negative cases. ROC curves are characterized by the area underneath the ROC curve (AUC) [54]. AUC acts as a measure of the model's overall performance across thresholds, with a higher AUC indicating better overall discrimination. An AUC of 1 indicates a perfect classification across thresholds by a model, while an AUC of 0.5 is equivalent to randomly guessing. If a model has an AUC less than 0.5, the model performed worse than random guessing.

AUC does not account for real-world costs of misclassification. Moreover, AUC is an *asymmetric* metric. Its value is dependent on how the minority and majority classes are defined. This means, in highly imbalanced datasets, AUC can be arbitrarily inflated depending on which class is designed the positive and negative classes [75].

### 2.7.2 $F_1$ -Score

The  $F_1$ -score is the harmonic mean of the sensitivity and precision. We define  $F_1$ -score as

$$F_1 = 2 \left( \frac{\text{Precision} \times \text{Sensitivity}}{\text{Precision} + \text{Sensitivity}} \right) = \frac{2TP}{2TP + FP + FN}. \quad (2.41)$$



By construction  $0 \leq F_1 \leq 1$ . The score is 1 only when  $FP = 0$  and  $FN = 0$  (i.e., precision and sensitivity are both 1), and it is 0 whenever  $TP = 0$  under the assumption that  $FP + FN > 0$  [68].  $F_1$ -score penalizes imbalance as extremely high precision with low sensitivity (or vice versa) yields a relatively low  $F_1$ .  $F_1$ -score depends only on  $TP$ ,  $FP$ , and  $FN$  (and not on true negatives,  $TN$ ). Thus,  $F_1$ -score for imbalanced classification is essentially a measure of positive-class performance only as it does not consider how the negative observations are classified. For this reason, like AUC,  $F_1$ -score is asymmetric.

### 2.7.3 Geometric Mean

The *geometric mean* metric (G-mean) is the geometric mean between sensitivity and specificity

$$\text{G-mean} = \sqrt{\text{Sensitivity} \times \text{Specificity}} = \sqrt{\frac{TP}{TP + FN} \times \frac{TN}{FP + TN}} \quad (2.42)$$

By construction  $0 \leq \text{G-mean} \leq 1$ . Due to the multiplicative nature, if one class has very low sensitivity,  $\text{G-mean} \rightarrow 0$  regardless of the other class's accuracy. As a result, optimizing G-mean tends to force the classifier to improve minority-class detection without sacrificing majority-class correctness as it achieves a high score only when both sensitivity and specificity are high [8]. The G-mean criterion for optimizing models corresponds to a Bayes-like objective that treats all classes symmetrically in the limit of large data [65]. The G-mean is symmetric if sensitivity and specificity are recalculated according to the class label swaps.

Having concluded our necessary background, we now move to the next section which contains our proposed algorithmic modifications to CCCD classifiers. In Chapters 3-5 we will compare our new variants on simulated and real datasets to other discussed models from this section. As noted, all comparisons are done using AUC,  $F_1$ -score, and G-mean to provide deeper insight into the behavior of the classification performance modified algorithms.

## Part II

# Algorithmic Modifications of Graph-Based Learners

## Chapter 3

### Enhancing P-CCCD Performance in Class Overlap Settings

RW-CCCD generally outperforms P-CCCD in handling complex or intertwined class boundaries due to its use of random walks, which incorporate connectivity and local geometric structure among same-class points [76]. In contrast, P-CCCD may be more sensitive to outliers and irregular point spacing, as it determines cover radii based solely on the opposing class, without accounting for intra-class structure. In this chapter, we explore two approaches aimed at improving the performance of P-CCCD under conditions of class overlap and imbalance

- (i) incorporating a cardinality-based scoring mechanism into the P-CCCD classification rule, which seeks to better capture local geometric information, and
- (ii) applying bagging to P-CCCD classifiers to enhance robustness and reduce variance.

#### 3.1 Modified P-CCCD with Cardinality-Based Scoring

For classification of new observation  $z$  with P-CCCD, we typically use the location of  $z$  and classify it with respect to the class that is “denser” in probability which is estimated by a smaller scaled dissimilarity metric [76, 100]. In Equation (2.31), the radius can be viewed as an indicator of the local density around the point  $x$ , thus a point with a bigger radius might suggest that the point  $z$  is more likely to be drawn from the same distribution as  $x$ . However, in practice this might not always be true. In order to more accurately indicate density of the covering balls we propose the following modification to Equation (2.31)

$$\hat{f}(z) = \arg \min_{U \in \{\mathcal{X}_0, \mathcal{X}_1\}} \left[ \min_{x: B(x, r) \in \mathcal{C}_U} \rho(z, x)^{|\overline{N(x)}|^e} \right] \quad (3.1)$$

where  $|\overline{N(x)}|$  is the number of target class points contained in the open covering ball at point  $x$ , and  $e \in [0, 1]$ . Parameter  $e$  acts as a way to control the level at which cardinality is incorporated into the classification similar to RW-CCCD. We denote these modified decision rule P-CCCDs classifiers as MP-CCCD.  $|\overline{N(x)}|$  acts similar to the score parameter in Equation (2.35) and  $e$  refines the indicator of density with respect to each class. In essence, covering balls with larger radii that do not catch many target class points are given less importance than balls that catch many points. When  $e = 0$ , the decision rule generalizes to P-CCCD. The extension to multi-class settings is the same as for P-CCCD discussed in Chapter 2.

### 3.1.1 Consistency of MP-CCCD

In this section, we present theoretical proofs leading up to the *consistency* of MP-CCCD. Our proofs follow as direct extensions from discussion in [100]. We assume that all classification errors are treated equally.

**Definition 3.1.1.** An estimator is *consistent* if  $L(\hat{f})$  converges to the Bayes optimal  $L^*$  [35, 63].

Let  $f$  represent the underlying true mapping of an observation  $z$  with unobserved class label of either  $\{0, 1\}$ . Let  $\hat{f}_{MP-CCCD}$  be the MP-CCCD classifier decision rule mapping given labeled training data, with  $L(\hat{f}_{MP-CCCD}) = P[\hat{f}_{MP-CCCD}(z|\mathcal{X}_j, \mathcal{X}_{1-j}) \neq f(z)]$  being the conditional probability of misclassification for MP-CCCD. In MP-CCCD, the choice of  $e$  is independent from the greedy calculation of the minimum dominating sets  $S_{\mathcal{X}_j}$  and  $S_{\mathcal{X}_{1-j}}$  and the respective radii, and thus does not affect the size or location of the cover. Therefore, our proposed modification should have the same decisions for cases of  $z \in \mathcal{C}_{\mathcal{X}_j} \setminus \mathcal{C}_{\mathcal{X}_{1-j}}$ , and  $z \in \mathcal{C}_{\mathcal{X}_{1-j}} \setminus \mathcal{C}_{\mathcal{X}_j}$  as P-CCCD, which is to classify  $z$  as a member of the class cover it is contained

solely in. The difference occurs when the observation lies in the overlap between two class covers or outside of the covers. We can rewrite the decision function as an indicator function,

$$\hat{f}_{MP-CCCD}(z|\mathcal{X}_j, \mathcal{X}_{1-j}) = I \left[ \min_{x'_j \in \hat{S}_{\mathcal{X}_{1-j}}} (d(x'_j, z)/r(x'_j))^{\overline{N(x'_j)}|^e} < \min_{x_j \in \hat{S}_{\mathcal{X}_j}} (d(x_j, z)/r(x_j))^{\overline{N(x_j)}|^e} \right]. \quad (3.2)$$

Calculating the *empirical* (also known as *resubstitution*) *error rate* is done by

$$\hat{L}^{(R)}(\hat{f}_{MP-CCCD}) = \frac{\left[ \sum_{x_j \in \mathcal{X}_j} I \left[ \hat{f}_{MP-CCCD}(x_j) \neq 0 \right] + \sum_{x'_j \in \mathcal{X}_{1-j}} I \left[ \hat{f}_{MP-CCCD}(x'_j) \neq 1 \right] \right]}{|\mathcal{X}_j| + |\mathcal{X}_{1-j}|} \quad (3.3)$$

By construction  $\hat{L}^{(R)}(\hat{f}_{MP-CCCD}) \geq 0$ . We first show that if the dissimilarity between the training datas is strictly positive, that the empirical error rate of the MP-CCCD classifier becomes zero.

**Theorem 3.1** (Empirical Error Rate of MP-CCCD). Let  $\mathcal{D} = \{(x_1, y_1), \dots, (x_n, y_n)\} \subset \mathbb{R}^d \times \mathcal{Y}$  be a training data with class labels  $\mathcal{Y} = \{0, 1\}$ . Define the class-specific subsets  $\mathcal{X}_0 = \{x_i \in \mathbb{R}^d \mid y_i = 0\}$  and  $\mathcal{X}_1 = \{x_i \in \mathbb{R}^d \mid y_i = 1\}$ . Suppose the between class dissimilarity is strictly positive, i.e.,

$$\min_{x \in \mathcal{X}_0, x' \in \mathcal{X}_1} \rho(x, x') > 0,$$

and that the MP-CCCD classifier constructs pure and proper class covers  $\mathcal{C}_0$  and  $\mathcal{C}_1$  using covering balls of radius defined by (2.27). Then the empirical error of MP-CCCD on  $\mathcal{D}$  is zero:

$$\hat{L}^{(R)}(\hat{f}_{MP-CCCD}) = 0.$$

*Proof.* Assume, for contradiction, that  $\hat{L}^{(R)}(\hat{f}_{MP-CCCD}) > 0$ . Then there exists a training point  $x_0 \in \mathcal{X}_0$  that is misclassified as class 1.

Let  $\mathcal{C}_0$  and  $\mathcal{C}_1$  be the class covers constructed around training points in  $\mathcal{X}_0$  and  $\mathcal{X}_1$ , respectively. Since these covers are assumed to be pure, we have  $\mathcal{C}_0 \cap \mathcal{X}_1 = \emptyset$  and  $\mathcal{C}_1 \cap \mathcal{X}_0 = \emptyset$ . Properness implies every point in  $\mathcal{X}_k$  is covered by some covering ball in  $\mathcal{C}_k$ .

The MP-CCCD classifier uses a scoring function that depends on the scaled distance to the nearest prototype (dominating point) in each class. Misclassification of  $x_0$  implies:

$$\min_{x_j \in \hat{S}_{\mathcal{X}_1}} \left( \frac{\rho(x_0, x_j)}{r(x_j)} \right)^{|\overline{N(x_j)}|^e} < \min_{x_i \in \hat{S}_{\mathcal{X}_0}} \left( \frac{\rho(x_0, x_i)}{r(x_i)} \right)^{|\overline{N(x_i)}|^e},$$

where  $\hat{S}_{\mathcal{X}_k}$  denotes the dominating set of the cover  $\mathcal{C}_k$ .

Let  $x_j^* \in \hat{S}_{\mathcal{X}_1}$  and  $x_i^* \in \hat{S}_{\mathcal{X}_0}$  be the minimizers of the respective expressions. Then:

$$\rho(x_0, x_j^*) < \rho(x_0, x_i^*).$$

But since  $x_0, x_i^* \in \mathcal{X}_0$  and  $x_j^* \in \mathcal{X}_1$ , this contradicts the assumption that the minimum inter-class dissimilarity is strictly positive. Thus, such a misclassification cannot occur.

Further, since  $\mathcal{C}_1$  is pure,  $x_0$  cannot be covered by any covering ball in  $\mathcal{C}_1$ , which contradicts the possibility that the classifier could assign  $x_0$  to class 1.

Therefore,  $\hat{L}^{(R)}(\hat{f}_{\text{MP-CCCD}}) = 0$ . □

We now move towards building our theoretical results on consistency. Before we begin the proof, we lay out formal definitions and other theoretical tools necessary to complete it.

**Definition 3.1.2.** (Continuity Condition) A dissimilarity measure  $\rho$  satisfies the continuity condition if for any  $z \in s(F)$  where  $s(F)$  is the support of distribution  $F$ , and  $\epsilon > 0$  we have

$$P_F[\arg \min_{x \in X} \rho(x, z) < \epsilon] > 0$$

For ease of notation, we define for set  $X$ ,  $\rho(X, z) = \arg \min_{x \in X} \rho(x, z)$ . It is necessary to show that our dissimilarity function satisfies the continuity condition before we can prove consistency [99, 35].

**Lemma 3.1.1** (Continuity of MP-CCCD Dissimilarity Score). Let  $X = \{x_1, \dots, x_n\} \subset \mathbb{R}^d$  be a finite set of reference points, and let  $z \in \mathbb{R}^d$ . Assume:

- $d(x, z)$  is an  $L_p$  distance for some  $p \geq 1$ ,
- $r(x) > 0$  is continuous for each  $x \in X$ ,
- For each  $x \in X$ ,  $|\overline{N}(x)| \in \mathbb{N}$  is fixed and  $e \in [0, 1]$ .

Define

$$\rho(X, z) = \min_{x \in X} \left[ \frac{d(x, z)}{r(x)} \right]^{|\overline{N}(x)|^e}.$$

Then the function  $\rho(X, z)$  is continuous on  $\mathbb{R}^d$ .

*Proof.* Fix the set  $X = \{x_1, \dots, x_n\} \subset \mathbb{R}^d$ . For each  $x_i \in X$ , define the function

$$f_i(z) = \left[ \frac{d(x_i, z)}{r(x_i)} \right]^{|\overline{N}(x_i)|^e}.$$

Since  $d(x_i, z)$  is continuous in  $z$ ,  $r(x_i) > 0$  is constant with respect to  $z$ , and the exponent  $|\overline{N}(x_i)|^e$  is by definition a constant  $\geq 1$ . It follows that each  $f_i(z)$  is continuous in  $z$ . The function  $\rho(X, z) = \min_{i=1, \dots, n} f_i(z)$  is the *pointwise minimum of finitely many continuous functions*, and hence is continuous.

Therefore,  $\rho(X, z)$  is continuous on  $\mathbb{R}^d$ . □

Having *strictly separable* class conditional distributions will be a necessary condition as we prove a weak consistency for *i.i.d.* training data [35]. We take this opportunity to define strict separability.

**Definition 3.1.3.** (Strictly Separable) class conditional distributions  $F_{\mathcal{X}_0}, F_{\mathcal{X}_1}$  are *strictly separable* if

$$\inf_{x_i \in s(F_{\mathcal{X}_0}), x_j \in s(F_{\mathcal{X}_1})} \rho(x_i, x_j) = \delta > 0.$$

We will show that for any strictly separable class conditional distributions random variable  $Z \sim F$ , that

$$\lim_{x_0 \in \hat{S}_j} \min (d(x_0, z)/r(x_0))^{\overline{N(x_0)}^e} \xrightarrow{a.s.} y_j < 1 \quad (3.4)$$

and

$$\lim_{x_0 \in \hat{S}_{1-j}} \min (d(x_0, z)/r(x_0))^{\overline{N(x_0)}^e} \xrightarrow{a.s.} y_{1-j} \geq 1 \quad (3.5)$$

for any  $j = 0, 1$ . We now proceed with the theorem and proof.

**Theorem 3.2** (Bayes Consistency of MP-CCCD). Let the data  $(x_i, y_i) \sim F = \pi_0 F_0 + (1 - \pi_0) F_1$  be drawn i.i.d. from a binary mixture distribution over  $\mathbb{R}^d \times \{0, 1\}$ , where each  $F_j$  has a compact support  $s(F_j) \subset \mathbb{R}^d$ , is continuous, and the supports are strictly separated:

$$\inf_{x_0 \in s(F_0), x_1 \in s(F_1)} \rho(x_0, x_1) > 0.$$

Let  $\hat{f}_{MP-CCCD}$  be the classification rule constructed from training data of size  $n$  using pure and proper class covers  $\mathcal{C}_0^{(n)}$  and  $\mathcal{C}_1^{(n)}$  formed by Equation (2.27), with dissimilarity function  $\rho$  being the MP-CCCD dissimilarity score satisfying continuity. Then

$$\lim_{n \rightarrow \infty} \hat{L}^{(R)}(\hat{f}_{MP-CCCD}) = L^*,$$

where  $L^*$  is the Bayes error rate.

*Proof.* Since each  $F_j$  is continuous and supported on a compact set, by the Glivenko–Cantelli theorem and uniform convergence of empirical distributions, the empirical distributions  $\hat{F}_j^{(n)}$  converge uniformly to  $F_j$  almost surely.

Because the sample points  $\mathcal{X}_j^{(n)}$  become dense in the compact support  $s(F_j)$  almost surely, and because the class cover  $\mathcal{C}_j^{(n)}$  is formed by continuous dissimilarity-based balls



around these points (with radii controlled by a continuous function of local distances), it follows that  $\mathcal{C}_j^{(n)}$  converges to  $s(F_j)$  in Hausdorff distance, almost surely.

Because the class-conditional supports are strictly separated and the covers are pure and proper by construction, for sufficiently large  $n$  there exists a margin  $\delta > 0$  such that each  $x \in \mathcal{X}_j^{(n)}$  is at distance at least  $\delta$  from any point in  $\mathcal{X}_{1-j}^{(n)}$ .

Then, for any test point  $z \in s(F_j)$ , the probability that its nearest dominating prototype from class  $j$  lies closer (under the scaled dissimilarity  $\rho(z, x)^{|\overline{N}(x)|^e}$ ) than any prototype from class  $1 - j$  converges to 1, since the class  $j$  prototypes become dense in  $s(F_j)$  and remain well-separated from the other class.

Finally, since the Bayes classifier assigns label  $j$  to any  $z$  with  $\eta(z) = \mathbb{P}(Y = j \mid X = z) > 1/2$ , and since  $\hat{f}_{MP-CCCD}$  matches the Bayes rule on the supports in the large-sample limit, we conclude

$$\hat{L}^{(R)}\left(\hat{f}_{MP-CCCD}\right) \rightarrow L^* \quad \text{as } n \rightarrow \infty.$$

□

Being *universal consistency* is a much more difficult proof as it requires consistency on every possible distribution. We conjecture that both P-CCCD and subsequently MP-CCCD are universal consistent, although arguments for this would have to be done through careful data-adaptive selection of prototype points which is left as future work.

### 3.2 Bagged P-CCCD

While bagging code has been implemented for P-CCCDs, no formal study to the authors' knowledge has been done on the performance effects. In addition to cardinality-based scoring, we formalize the effectiveness of this ensemble method for P-CCCD classifiers. We present Algorithm (3) for bagging with the P-CCCD classifier. The bagging algorithm framework does not consider P-CCCD dependency on non-target class observations to generate covers, as it does not guarantee non-target points in the bootstrap samples. For large enough

datasets, the probability of selection of at least one nontarget point will be high, even if classes are heavily imbalanced. However, alternative bootstrap sampling methods, such as stratified bootstrap can be used for smaller datasets.

---

**Algorithm 3** Bagged P-CCCD Classifier (BP-CCCD)

---

**Input:** Query point  $z$ , training data  $\mathcal{X} = \{(x_i, f(x_i))\}_{i=1}^N$  with  $K$  classes, number of replicates  $L$

**Output:** Aggregated label  $\hat{f}_B(z)$

**for**  $l = 1$  to  $L$  **do**

    Draw bootstrap sample  $\mathcal{B}_l \subseteq \mathcal{X}$

    Initialize  $\mathcal{Q} \leftarrow \emptyset$

**for**  $k = 1$  to  $K$  **do**

        Compute radii  $r_x$  for all  $x \in \mathcal{B}_l$  using (2.27)

        Construct class cover  $Q_{\mathcal{B}_k}$  and update  $\mathcal{Q} \leftarrow \mathcal{Q} \cup Q_{\mathcal{B}_k}$

**end for**

    Obtain class covers  $\mathcal{C} = \{C_1, \dots, C_K\}$  via greedy minimum dominating set

    Classify  $z$ :  $\hat{f}_l(z) \leftarrow$  decision rule in (2.31)

**end for**

Return  $\hat{f}_B(z) \leftarrow$  majority vote over  $\{\hat{f}_l(z)\}_{l=1}^L$

---

### 3.3 Monte Carlo Simulations

In this section, we present experimental results for the MP-CCCD and BP-CCCD classifiers on synthetic datasets, evaluated under varying levels of class overlap and imbalance. Performance is assessed using AUC,  $F_1$ -score, and G-mean for a more complete analysis of classifier performance. We compare these methods against P-CCCD and RW-CCCD classifiers as a baseline. A detailed overview of all CCCD-based classifiers considered in this study is provided in Table (3.1). Simulations were conducted in R. Code was written for the P-CCCD and MP-CCCD models, while the `cccd` package was used for RW-CCCD and BP-CCCD classification [104, 79].

CCCD classifiers require tuning for either  $\tau$ ,  $e$ , or in the case of MP-CCCD both  $\tau$  and  $e$ . To identify these optimal parameters, we conducted separate pilot studies aligned with the main simulation setting prior to the Monte Carlo simulation. Each pilot study consists of 200 replications during which we classify the test data using a grid search over  $\tau =$

| Method  | Description                                                                                                                                                          |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| P-CCCD  | P-CCCD with the optimal $\tau^*$ (in the pilot study) among $\tau = 0.0, 0.1, \dots, 1.0$ .                                                                          |
| RW-CCCD | RW-CCCD with the optimal $e^*$ (in the pilot study) among $e = 0.0, 0.1, \dots, 1.0$ .                                                                               |
| MP-CCCD | MP-CCCD with the optimal $e^*$ (in the pilot study) among $e = 0.0, 0.1, \dots, 1.0$ and optimal $\tau^*$ (in the pilot study) among $\tau = 0.0, 0.1, \dots, 1.0$ . |
| BP-CCCD | Bagged P-CCCD with optimal $\tau^*$ (in the pilot study) among $\tau = 0.0, 0.1, \dots, 1.0$ . The number of iterations were considered from $L = 2, \dots, 30$      |

Table 3.1: The description of other CCCD and MP-CCCD classifiers used for comparison in simulation studies.

$0.0, 0.1, \dots, 1.0$  and  $e = 0.0, 0.1, \dots, 1.0$ . We record the maximum evaluation metric along with the corresponding parameters, and update the frequency of each recorded parameter. The optimal parameter is denoted with  $*$ , i.e.,  $\tau^*$  for P-CCCD. We consider  $\tau = \epsilon$  (machine epsilon) as  $\tau = 0.0$  in the relevant CCCD settings.

### 3.3.1 Performance Under Class Overlap Settings

We first explore how each classifier performs under class overlap. Training data were simulated from the distribution  $F_{\mathcal{X}_0} = \text{Uniform}(0, 1)^d$  for Class 0 and  $F_{\mathcal{X}_1} = \text{Uniform}(\Delta, 1 + \Delta)^d$  for Class 1, with  $n_0 = n_1 = 500$  observations in each class. Throughout this work, we consider  $\Delta = 0.25, 0.5, 0.75$ , where  $\Delta$  serves as a shift parameter controlling the degree of overlap between the class distributions.

We define the overlapping region as  $E = s(F_{\mathcal{X}_0}) \cap s(F_{\mathcal{X}_1})$ . Since both classes are sampled from axis-aligned  $d$ -dimensional hypercubes, this intersection is given by  $E = (\Delta, 1)^d$ , which indicates that larger values of  $\Delta$  reduce class overlap. The overlap region  $E$  depends on both  $\Delta$  and the dimensionality  $d$ , however. As  $d$  increases, the volume of  $E$  increases, and points become increasingly sparse in  $\mathbb{R}^d$ , effectively reducing the practical overlap. To better

capture the interplay between  $\Delta$  and  $d$ , we define overlapping ratio  $\alpha$  as,

$$\alpha = \frac{\text{Vol}(s(F_{\mathcal{X}_0}) \cap s(F_{\mathcal{X}_1}))}{\text{Vol}(s(F_{\mathcal{X}_0}) \cup s(F_{\mathcal{X}_1}))} = \frac{(1 - \Delta)^d}{2 - (1 - \Delta)^d} \quad (3.6)$$

where  $\text{Vol}(\cdot)$  denotes the  $d$ -dimensional volume functional. By construction,  $\alpha \in [0, 1]$ , where  $\alpha = 0$  implies no overlap, and  $\alpha = 1$  indicates a complete overlap. Moreover, we note that the shift parameter  $\Delta$  can be expressed in terms of  $\alpha$  as:

$$\Delta = 1 - \left( \frac{2\alpha}{1 + \alpha} \right)^{1/d}. \quad (3.7)$$

We adopt this overlapping ratio  $\alpha$  throughout the remainder of this work to provide a measure of class overlap that is both dimension- and shift-aware for simulation studies.

We separate our derived  $\alpha$  for each  $(\Delta, d)$  combination into either “low overlap” for the first tertile of  $\alpha$ , “moderate overlap” for the second tertile, and “high overlap” for the remaining tertile. For test data generation, we continue simulations until the standard error of the chosen performance metric across all classifiers falls below 0.0005, ensuring stability in the Monte Carlo estimates.

We begin by examining the influence of class overlap on the selection of optimal parameters  $\tau$  and  $e$  within the MP-CCCD framework. Figure 3.1 presents surface plots of the average count of optimal configurations across the  $(\tau, e)$  parameter grid. Notably, average counts are substantially higher in low-overlap settings ( $\alpha \rightarrow 0$ ) compared to those with high overlap ( $\alpha \rightarrow 1$ ), suggesting that hyperparameter tuning becomes less critical when classes are well separated as more hyperparameter combination values share the optimal performance.

As  $\alpha$  decreases, the optimal value of  $\tau$  converges to  $\tau^* = 0$ , indicating a reduced reliance on large-volume covers. Despite this, the parameter  $e$  continues to exhibit multiple peaks across moderate to high-overlap scenarios, underscoring its importance in model tuning. We

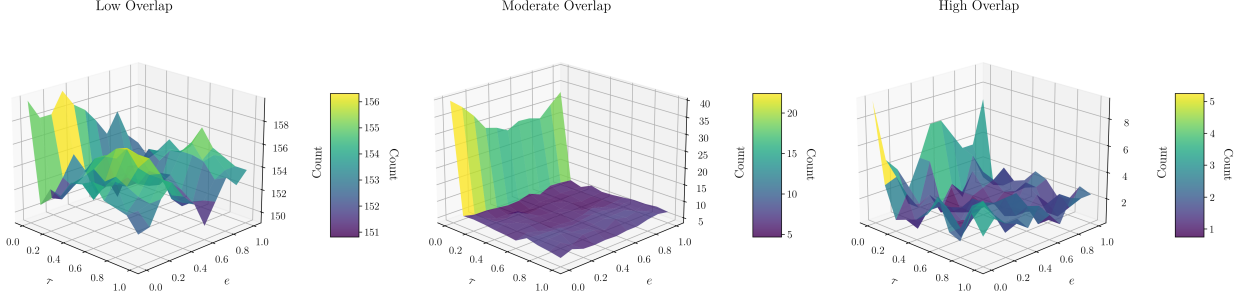


Figure 3.1: Surface plots of optimal AUC counts for each  $(\tau, e)$  combination of MP-CCCD classifiers on balanced, overlapping simulations. As classes become more overlapping, average counts decrease.

note that setting  $\tau^* = 0$  yields either optimal or near-optimal performance across all settings. P-CCCD requires the radii of the covering ball to incorporate similar geometric density information as MP-CCCD. Consequently, MP-CCCD maintains strong decision-making capabilities without requiring large covering volumes. Tuning  $\tau$  may often be unnecessary due to the additional geometric information from MP-CCCD.

We now consider how the  $F_1$ -score influences the selection of optimal parameters  $\tau$  and  $e$ . Class 1 is taken to be the “negative” class for calculation of  $F_1$ -score, although there should be no significant difference in the labels since classes are balanced. Figure (3.2) displays heatmaps of the average  $F_1$ -score optimality counts for each  $(\tau, e)$  combinations. Under this metric, we observe some similar trends to those seen with AUC. Average optimality counts decrease as  $\alpha$  increases, suggesting that a greater number of  $(\tau, e)$  combinations yield competitive performance in the most separable settings, and in moderate overlap settings, the optimal  $\tau$  parameter is  $\tau^* = 0.0$ . However, in high overlap settings higher values of  $\tau$  still occur frequently. Larger volume covers improve performance of Class 1 class observations. Additionally, borderline observations between classes become more common as class supports become less separable. High  $\tau$  values occur in tandem with lower values of  $e$ .  $F_1$ -score penalizes misclassifications more strictly than AUC, and lower  $e$  values may be favored to reduce over-prediction, ultimately improving precision.

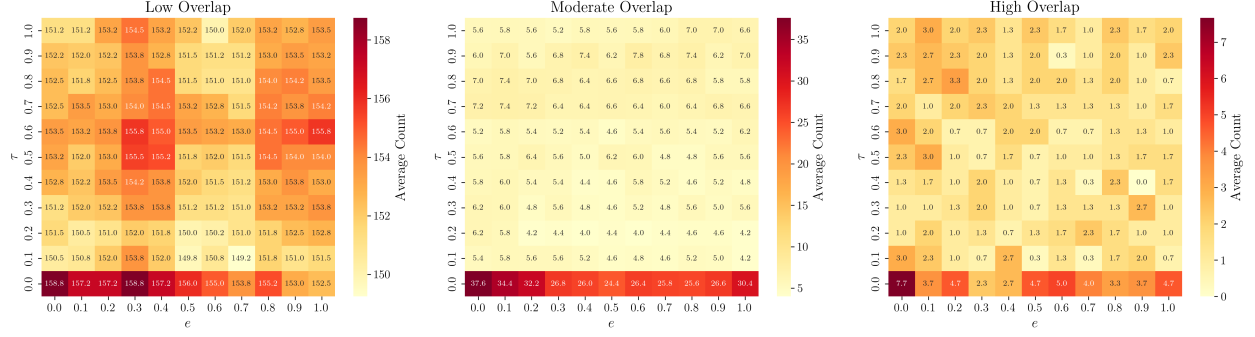


Figure 3.2: Heatmaps of the average optimal  $F_1$ -score counts for each  $(\tau, e)$  combination of MP-CCCD classifiers on balanced, overlapping simulations. In high overlap settings,  $F_1$ -scores likely down-weights cardinality, except for few data-dependent scenarios.

Under high overlap settings when  $\tau^* = 0$ , the values  $e = 0.0, 0.2, 0.5, 0.6, 1.0$  have the highest frequency. When cover ball volumes are the smallest possible, cardinality becomes less reliable for identifying Class 1 decision regions as the probability  $P(|N(x)| > 1) \rightarrow 0$ , reflecting the proximity of points to Class 0 observations. Lower  $e^*$  values, such as  $e^* = 0$ , reduce dependence on cardinality. Nevertheless, in certain data-dependent scenarios local structure can still temporarily favor using cardinality. One example is when a target class points lie within local non-target clusters. A greater reliance on cardinality (which corresponds to higher  $e$ ) may remain effective, particularly when the target resembles an outlier.

A similar effect is observed under the G-mean evaluation metric. Figure (3.3) displays heatmaps of the average optimality counts for each  $(\tau, e)$  combination. This furthers the evidence that there might exist specific data-dependent settings where cardinality offers no clear advantage in balanced, overlap settings when evaluated using metrics that account for precision.

The results on how bagging influences the selection of  $\tau^*$  for the BP-CCCD classifier are omitted for brevity, but indicated a consistent preference for larger  $\tau$  values across all settings. Under moderate and high overlapping, only  $\tau^* = 1$  was selected as optimal. In overlapping class scenarios, decision boundaries become more ambiguous. Bootstrap sampling inherently omits a portion of the training data, which can result in sparser or fragmented class coverage, particularly in high-dimensional spaces. Employing larger cover volumes helps mitigate these

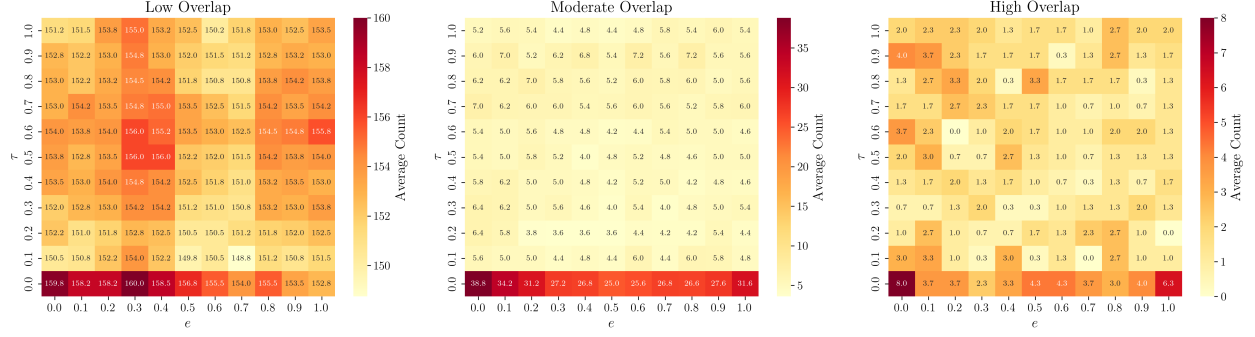


Figure 3.3: Heatmaps of the average optimal G-mean counts for each  $(\tau, e)$  combination of MP-CCCD classifiers on balanced, overlapping simulations.  $(\tau, e)$  combination performance is similar to that of  $F_1$ -score.

issues by ensuring broader class representation, even in the presence of “missing data” due to resampling.

We conclude this section by evaluating the performance of the selected MP-CCCD and BP-CCCD classifiers in comparison to other CCCD variants. Table (3.2) presents pairwise mean differences between MP-CCCD and each alternative CCCD classifier under the AUC,  $F_1$ -score, and G-mean evaluation metrics. Positive values indicate that MP-CCCD outperforms the competitor, while negative values indicate the opposite.

The results show that MP-CCCD marginally outperforms both P-CCCD and RW-CCCD across all settings. Additionally, BP-CCCD either ties or surpasses MP-CCCD (and thus P-CCCD and RW-CCCD) in every scenario. Performance differences increase under moderate and high class overlap, though none of the pairwise differences exceed 0.05. MP-CCCD and BP-CCCD are marginally more robust to class overlap than the compared baseline methods. Despite the lack of practical significance for the difference, we still advocate for the use of tuned MP-CCCD over tuned P-CCCD or RW-CCCD in all settings. While performance might be similar, in high overlap scenarios MP-CCCD is able to leverage cardinality in situations of difficult local structure better than P-CCCD and RW-CCCD counterparts.

| Overlap Category | Metric       | vs BP-CCCD | vs P-CCCD | vs RW-CCCD |
|------------------|--------------|------------|-----------|------------|
| Low Overlap      | AUC          | 0.000      | 0.002     | 0.003      |
|                  | $F_1$ -Score | 0.000      | 0.002     | 0.002      |
|                  | G-Mean       | 0.000      | 0.002     | 0.002      |
| Moderate Overlap | AUC          | -0.001     | 0.015     | 0.011      |
|                  | $F_1$ -Score | -0.001     | 0.016     | 0.011      |
|                  | G-Mean       | -0.001     | 0.016     | 0.011      |
| High Overlap     | AUC          | -0.005     | 0.019     | 0.018      |
|                  | $F_1$ -Score | 0.001      | 0.021     | 0.020      |
|                  | G-Mean       | -0.005     | 0.019     | 0.019      |

Table 3.2: Performance difference of MP-CCCD versus BP-CCCD, P-CCCD, and RW-CCCD for balanced data across different scenarios of class overlap and evaluation metrics. Positive differences indicate MP-CCCD performed better.

### 3.3.2 Performance Under Class Overlap and Class Imbalance Settings

To evaluate the performance of the new proposed CCCD classifiers under both class imbalance and class overlap, we simulate majority class observations  $\mathcal{X}_0$  from the distribution  $F_{\mathcal{X}_0} = \text{Uniform}(0, 1)^d$ , and minority class observations  $\mathcal{X}_1$  from  $F_{\mathcal{X}_1} = \text{Uniform}(\Delta, 1 + \Delta)^d$ , with  $n_0 = 500$  and  $n_1 = 500q$ , where  $q = 0.25, 0.50, 0.75$  defines the global class imbalance ratio.

The local imbalance ratio in the overlap region is  $q(E) = |\mathcal{X}_{1-j} \cap E|/|\mathcal{X}_j \cap E|$ , where  $E$  is the earlier defined class overlap region of the supports. Simulation settings were chosen so that local imbalance  $q(E)$  closely approximates the global imbalance  $q$ , that is,  $q(E) \approx q$ . Higher values of  $q$  correspond to more balanced class distributions both globally and locally. Test datasets are simulated until the standard error of each evaluation metric across all classifiers is below 0.0005 in the Monte Carlo simulations, ensuring reliable performance comparisons.

We begin by analyzing how class imbalance influences the selection of optimal parameters  $\tau$  and  $e$  in the MP-CCCD classifier. Figures (3.4), (3.5), and (3.6) illustrate how optimality counts vary with overlap under each  $q$  for AUC,  $F_1$ -score, and G-mean respectively. The relationships are estimated using smoothed functions to better capture underlying trends in



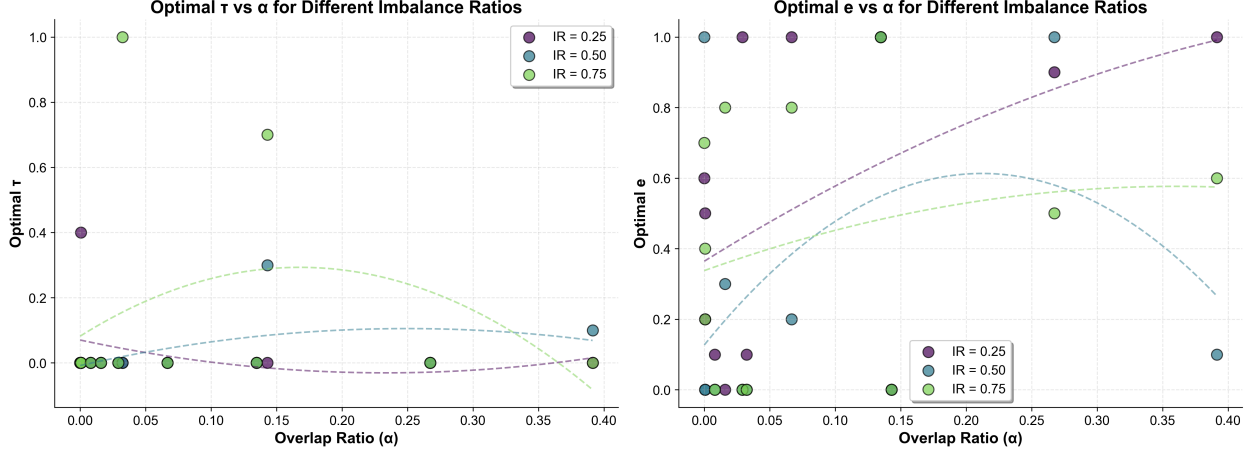


Figure 3.4: Scatterplot of optimal  $\tau$  (left) and optimal  $e$  (right) selections for MP-CCCD as overlap increases under each level of imbalance under AUC metric.

parameter selection across overlap and imbalance conditions. Notably, class imbalance does not significantly impact the choice of the optimal  $\tau^*$  for AUC and G-mean. The  $F_1$ -score shows a comparable trend under high and moderate imbalance but deviates somewhat under low imbalance.

In contrast, imbalance has a pronounced effect on the selection of the cardinality parameter  $e^*$ . For AUC, both low and high imbalance settings tend to favor larger values of  $e$ , indicating greater incorporation of cardinality information in the classification decision. Within the P-CCCD framework, the volume of the minority class cover tends to be smaller by construction [76]. MP-CCCD corrects this potential bias by incorporating cardinality scores, allowing for a more balanced consideration of local class structure despite the smaller cover volume. Higher values for  $e^*$  become more necessary as classes become more imbalanced. Under moderate imbalance, this relationship appears more quadratic but still reflects an overall increase in  $e^*$ .

For tuning parameter  $e^*$  under the  $F_1$ -score metric, we observe a negative quadratic relationship at low and high imbalances, but a positive quadratic effect at moderate imbalance. These results contrast with the tuning results from the balanced setting where higher counts were observed at low and high  $e$  values. Borderline points in imbalanced settings are

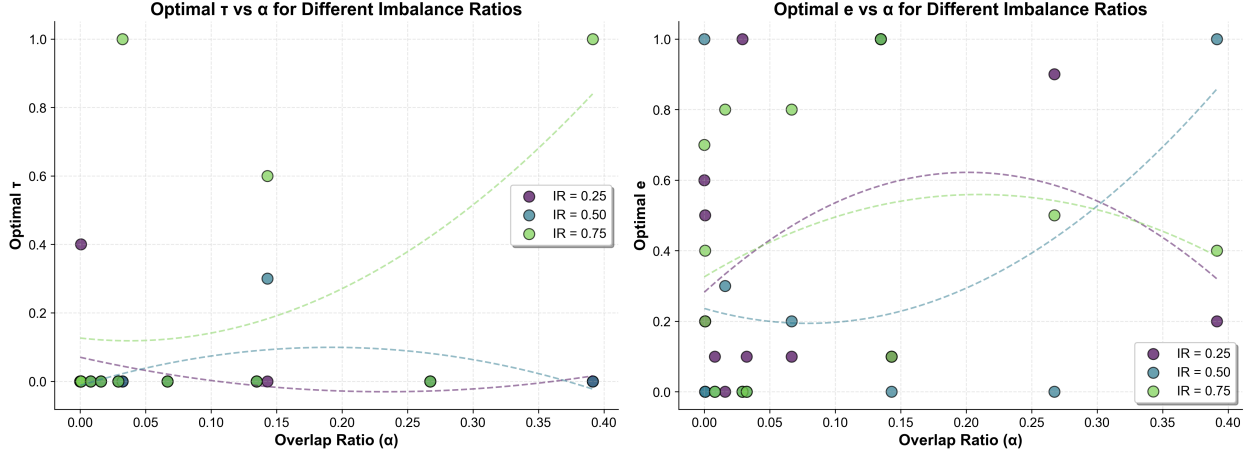


Figure 3.5: Scatterplot of optimal  $\tau$  (left) and optimal  $e$  (right) selections for MP-CCCD as overlap increases under each level of imbalance under  $F_1$ -score metric.

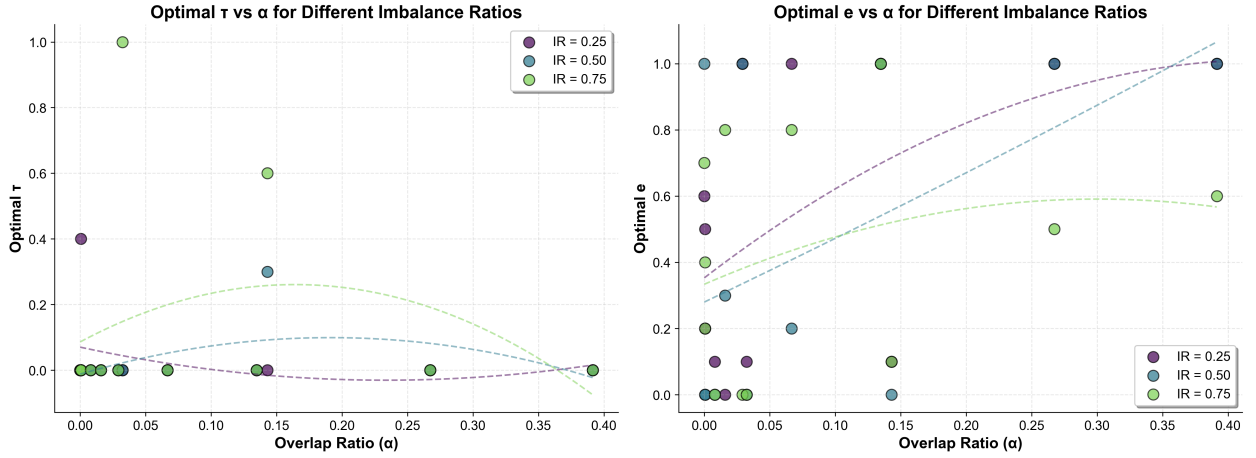


Figure 3.6: Scatterplot of optimal  $\tau$  (left) and optimal  $e$  (right) selections for MP-CCCD as overlap increases under each level of imbalance under G-mean metric.

likely to be majority class points, and thus a complete disregard or inclusion of cardinality potentially either under or over predicts minority class observations respectively. However, we see that for G-mean the selection of  $e^*$  patterns mirrors that of AUC.  $F_1$ -scores are more sensitive to the exact distribution of false positives and false negatives as G-mean does not care about precision. This might indicate that higher values of  $e$  lead to overall greater discrimination performance, but less precision.

For BP-CCCD, the influence of class imbalance on the selection of the  $\tau$  parameter is evident primarily when classes are more separable. Figure (3.7) illustrates the distribution

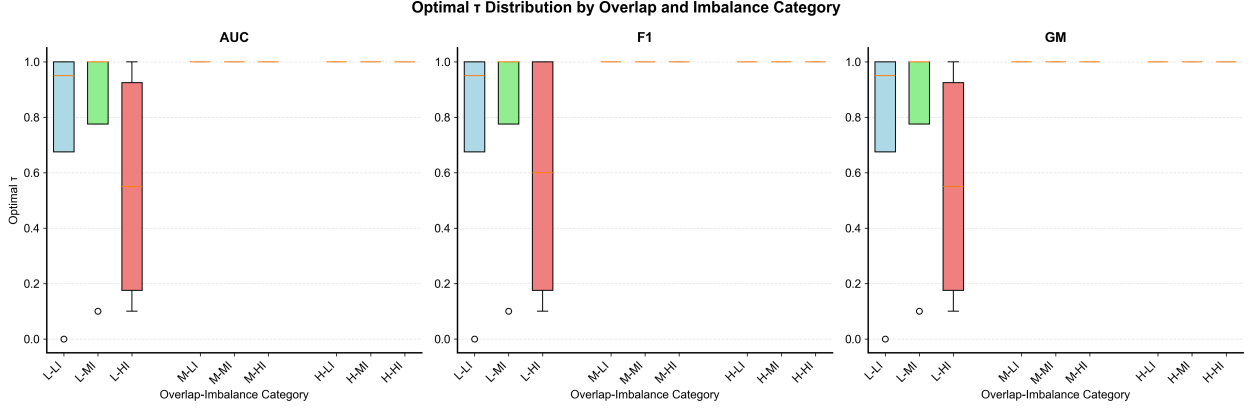


Figure 3.7: Distribution of BP-CCCD optimal  $\tau^*$  counts for the combination of each metric, overlap, and imbalance ratio. Class overlap plays a bigger role in selection of  $\tau^*$  than class imbalance.

of optimal  $\tau^*$  values across varying combinations of class overlap, imbalance, and evaluation metrics. In scenarios with well-separated classes, increasing class imbalance substantially broadens the range of  $\tau^*$  values, indicating that imbalance affects  $\tau$  selection in these settings. This effect is likely driven by variability in minority class coverage across the different bootstrap ensembles. However, as class overlap transitions from low to moderate, the optimal  $\tau^*$  is always selected to be  $\tau^* = 1.0$  regardless of imbalance level. This suggests that class overlap exerts a stronger influence on  $\tau^*$  selection than imbalance for BP-CCCD.

We now evaluate the performance of MP-CCCD against other CCCD classifiers. Table (5.3) summarizes the mean performance differences between MP-CCCD and BP-CCCD, P-CCCD, or RW-CCCD across all combinations of imbalance, overlap, and metrics. In most scenarios, BP-CCCD performs as well as or better than MP-CCCD across all metrics. Although the mean differences consistently favor BP-CCCD, the magnitude of these differences is generally small.

MP-CCCD marginally outperforms both P-CCCD and RW-CCCD, with positive mean differences across all settings. Only under conditions of high class overlap and high imbalance for the  $F_1$ -score metric are the differences greater than 0.05. This might indicate that MP-CCCD has better precision than the other two CCCD variants in challenging settings characterized by substantial overlap and imbalance.

| Imbalance          | Overlap          | Metric       | vs BP-CCCD | vs P-CCCD    | vs RW-CCCD   |
|--------------------|------------------|--------------|------------|--------------|--------------|
| Low Imbalance      | Low Overlap      | AUC          | 0.000      | 0.002        | 0.002        |
|                    |                  | $F_1$ -Score | 0.000      | 0.002        | 0.002        |
|                    |                  | G-Mean       | 0.000      | 0.002        | 0.002        |
|                    | Moderate Overlap | AUC          | -0.001     | 0.017        | 0.012        |
|                    |                  | $F_1$ -Score | -0.001     | 0.020        | 0.014        |
|                    |                  | G-Mean       | -0.001     | 0.017        | 0.011        |
|                    | High Overlap     | AUC          | -0.005     | 0.021        | 0.019        |
|                    |                  | $F_1$ -Score | -0.004     | 0.025        | 0.015        |
|                    |                  | G-Mean       | -0.005     | 0.020        | 0.018        |
| Moderate Imbalance | Low Overlap      | AUC          | 0.000      | 0.002        | 0.003        |
|                    |                  | $F_1$ -Score | 0.000      | 0.003        | 0.004        |
|                    |                  | G-Mean       | 0.000      | 0.002        | 0.003        |
|                    | Moderate Overlap | AUC          | -0.003     | 0.015        | 0.009        |
|                    |                  | $F_1$ -Score | -0.005     | 0.022        | 0.019        |
|                    |                  | G-Mean       | -0.003     | 0.016        | 0.009        |
|                    | High Overlap     | AUC          | -0.004     | 0.022        | 0.017        |
|                    |                  | $F_1$ -Score | -0.008     | 0.032        | 0.028        |
|                    |                  | G-Mean       | -0.001     | 0.024        | 0.016        |
| High Imbalance     | Low Overlap      | AUC          | -0.001     | 0.002        | 0.007        |
|                    |                  | $F_1$ -Score | -0.001     | 0.004        | 0.007        |
|                    |                  | G-Mean       | -0.001     | 0.002        | 0.007        |
|                    | Moderate Overlap | AUC          | -0.005     | 0.017        | 0.008        |
|                    |                  | $F_1$ -Score | -0.011     | 0.031        | 0.041        |
|                    |                  | G-Mean       | -0.004     | 0.018        | 0.008        |
|                    | High Overlap     | AUC          | -0.003     | 0.024        | 0.014        |
|                    |                  | $F_1$ -Score | -0.020     | <b>0.055</b> | <b>0.100</b> |
|                    |                  | G-Mean       | 0.005      | 0.031        | 0.007        |

Table 3.3: Performance difference of MP-CCCD versus BP-CCCD, P-CCCD, and RW-CCCD for imbalanced data across different scenarios of class overlap and evaluation metrics. Positive differences indicate MP-CCCD performed better.

While both MP-CCCD and BP-CCCD generally outperform RW-CCCD and P-CCCD, particularly under severe class overlap and imbalance, there is a tradeoff in computational efficiency. MP-CCCD requires an additional loop over selected prototypes to count the cardinalities of each covering ball and then subsequent storage for these values increasing computational overhead. This makes its worst-case computational complexity for training time the order of  $\mathcal{O}(N^3d)$ . Bagging also adds computation due to the need to calculate  $L$  models across the multiple ensemble iterations, though this process can be parallelized to mitigate runtime.

To assess practical significance, we benchmarked the computational time of each CCCD classifier similar settings to the MC simulation study. We chose a combinations of  $d = 2, 10$ , overlap parameter  $\Delta = 0.25, 0.75$ , and imbalance ratios  $q = 0.25, 0.75$  for the time trial settings. For each configuration, 200 iterations were used. The mean runtime and standard deviations for each classifier were recorded. Table (3.4) summarizes the aggregated results across all settings. A detailed analysis of the effects of each variable on computational time is provided later in the conclusions. Computations were completed with resources provided by the Auburn University Easley Cluster.

| Index   | Mean $\pm$ SD       |
|---------|---------------------|
| P-CCCD  | $0.1681 \pm 0.1964$ |
| RW-CCCD | $0.1435 \pm 0.1363$ |
| BP-CCCD | $2.3978 \pm 3.5584$ |
| MP-CCCD | $3.4688 \pm 5.2414$ |

Table 3.4: Aggregated mean and standard deviations for P-CCCD, RW-CCCD, BP-CCCD, and MP-CCCD runtimes across simulated data settings.

We conclude this section by discussing the performance of MP-CCCD and BP-CCCD in high-dimensional settings. We generated majority class observations  $\mathcal{X}_0$  from the distribution  $F_{\mathcal{X}_0} = \text{Uniform}(0, 1)^{500}$  and minority class observations  $\mathcal{X}_1$  from  $F_{\mathcal{X}_1} = \text{Uniform}(0.25, 1.25)^{500}$ , with sample sizes  $n_0 = 250$  and  $n_1 = 250q = 250(0.25) \approx 63$ . This configuration was selected to represent the most challenging combination of shift parameter  $\Delta$  and imbalance ratio  $q$

from the Monte Carlo settings. Test datasets were simulated until the standard error of the selected evaluation metric fell below 0.0005. Both MP-CCCD and BP-CCCD converged rapidly to their highest achievable metric values within the initial few iterations, demonstrating their robustness against the curse of dimensionality. Given the quick convergence and stable performance, we omit detailed results of these high-dimensional simulations for brevity.

We provide a quick summary of the Monte Carlo simulation results,

- (i) MP-CCCD classifiers favor lower  $\tau^*$  values due to their ability to approximate local density information without the volume of the cover. High values for  $e^*$  are selected as classes become more imbalanced.
- (ii) Both MP-CCCD and BP-CCCD perform better than P-CCCD and RW-CCCD in all settings. However, practical significance is only seen when both class imbalance and class overlap are high.
- (iii) MP-CCCD has average runtimes much slower than other CCCD variants.

### 3.4 Real Data Experiments

In this section, we compare the performance of the MP-CCCD and BP-CCCD classifier against other CCCD classifiers on datasets from the UC Irvine (UCI) Machine Learning and KEEL repositories [38, 1]. The datasets chosen were the Sonar, Ionosphere, Segment0 (Segment), Page-Blocks0 (Pageblocks), Vowel0 (Vowel), Shuttle0vs4 (Shuttle), Yeast4, Yeast1289vs7, Yeast5, Yeast6, and Abalone19 (Abalone) datasets.

In real datasets where the support of classes are neither defined nor available, it is necessary to estimate the supports of the respective classes. Analysis of the data intrinsic characteristics was performed in [76], but are detailed here for completeness. SVDD was used to estimate the class supports using a Gaussian kernel. The selection of the kernel parameter  $\gamma$  changes the class description, therefore a range of  $\gamma = 2, 3, \dots, 10$  was considered to better

| Data         | q      | N    | d  |    | $\gamma = 2$ | $\gamma = 3$ | $\gamma = 4$ | $\gamma = 5$ | $\gamma = 6$ | $\gamma = 7$ | $\gamma = 8$ | $\gamma = 9$ | $\gamma = 10$ |
|--------------|--------|------|----|----|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|---------------|
| Ionosphere   | 1.78   | 351  | 35 | OR | 25%          | 36%          | 66%          | 69%          | 66%          | 79%          | 61%          | 76%          | 81%           |
|              |        |      |    | IR | 90.00        | 62.50        | 8.70         | 6.20         | 5.44         | 3.67         | 5.02         | 3.98         | 3.31          |
| Sonar        | 1.14   | 208  | 61 | OR | 4%           | 19%          | 23%          | 25%          | 26%          | 26%          | 27%          | 28%          | 28%           |
|              |        |      |    | IR | 1.22         | 1.04         | 0.96         | 0.93         | 0.96         | 0.97         | 0.97         | 0.90         | 0.90          |
| Yeast6       | 41.40  | 1484 | 9  | OR | 30%          | 46%          | 42%          | 31%          | 30%          | 30%          | 10%          | 13%          | 13%           |
|              |        |      |    | IR | 64.14        | 21.03        | 27.59        | 76.33        | 73.66        | 73.66        | 38.25        | 63.00        | 63.00         |
| Yeast5       | 32.70  | 1484 | 9  | OR | 6%           | 3%           | 0%           | 0%           | 0%           | 0%           | 6%           | 7%           | 0.1%          |
|              |        |      |    | IR | NA           | NA           | NA           | NA           | NA           | NA           | NA           | NA           | NA            |
| Yeast4       | 28.10  | 1484 | 9  | OR | 45%          | 22%          | 39%          | 37%          | 26%          | 26%          | 26%          | 26%          | 31%           |
|              |        |      |    | IR | 18.97        | 99.75        | 18.50        | 18.24        | 392.00       | 390.00       | 391.00       | 390.00       | 41.18         |
| Yeast1289vs7 | 30.70  | 1484 | 9  | OR | 45%          | 44%          | 69%          | 43%          | 30%          | 29%          | 29%          | 29%          | 29%           |
|              |        |      |    | IR | 24.47        | 23.76        | 24.34        | 23.00        | 47.33        | 45.83        | 45.83        | 45.66        | 45.50         |
| Vowel        | 9.98   | 988  | 14 | OR | 0%           | 0%           | 0%           | 0%           | 0%           | 0%           | 0%           | 0%           | 0%            |
|              |        |      |    | IR | NA           | NA           | NA           | NA           | NA           | NA           | NA           | NA           | NA            |
| Shuttle      | 13.87  | 1829 | 10 | OR | 0%           | 0%           | 0%           | 0%           | 0%           | 0%           | 0%           | 0%           | 0%            |
|              |        |      |    | IR | NA           | NA           | NA           | NA           | NA           | NA           | NA           | NA           | NA            |
| Abalone      | 129.40 | 4174 | 9  | OR | 25%          | 20%          | 15%          | 14%          | 13%          | 12%          | 12%          | 11%          | 11%           |
|              |        |      |    | IR | 104.30       | 104.30       | 163.75       | 197.33       | 278.00       | 262.50       | 253.00       | 244.00       | 234.00        |
| Segment      | 6.02   | 2308 | 20 | OR | 0%           | 0%           | 0%           | 0%           | 0%           | 0%           | 0%           | 0%           | 0%            |
|              |        |      |    | IR | NA           | NA           | NA           | NA           | NA           | NA           | NA           | NA           | NA            |
| Pageblocks   | 8.79   | 5472 | 11 | OR | 0.6%         | 0.6%         | 0.6%         | 0.8%         | 0.9%         | 1%           | 1%           | 1%           | 1%            |
|              |        |      |    | IR | 0.47         | 0.22         | 0.22         | 0.25         | 0.40         | 0.39         | 0.43         | 0.54         | 0.63          |

Table 3.5: Data intrinsic characteristics for the selected UCI and KEEL repository datasets.

understand the class descriptions. Table (3.5) details the real datasets intrinsic characteristics. Parameter  $q$  is the global imbalance ratio for the data set,  $N$  is the total cardinality,  $d$  is the number of dimensions in the data set, and “IR” stands for the local imbalance ratio in the overlapping region determined by the SVDD using Gaussian kernel with parameter  $\gamma$ . “OR” stands for the overlapping ratio which is the percentage of points from both classes residing in the overlapping region between the class descriptions. A value of IR=“NA” indicates that one of the classes has no members in the intersections of SVDD regions of classes and thus no local imbalance. Ionosphere, Sonar, Yeast6, Yeast4, Yeast1289vs7, and Abalone all achieve an OR above 25% for at least one of  $\gamma = 2, 3, \dots, 10$ . Of these, Ionosphere and Sonar have relatively low global class imbalance, while Sonar specifically has a relatively low local class imbalance across all  $\gamma$ . Other datasets, such as Yeast5, are globally imbalanced but have linearly separable classes.

The classifiers compared were the same as those in Table (3.1). Each dataset was split into five repetitions with 2 folds per repetition. For each repetition, evaluation metrics were found using fold 1 as the training data and fold 2 as the testing set. Then the roles were

reversed for the next iteration of the repetition for a total of 10 different models. All model comparisons are made using the combined  $5 \times 2$  cross-validation (CV)  $F$ -test. The combined  $5 \times 2$  CV  $F$ -test acts as an omnibus test for the ten possible  $t$ -tests with a null hypothesis that the expected performance difference between the two classifiers is zero across the data distribution [3]. Presented  $5 \times 2$  CV  $F$ -test results represent comparison of the selected model against the MP-CCCD model. Tables (3.6), (3.7), and (3.8) report the  $5 \times 2$  cross-validation results for AUC,  $F_1$ -score, and G-mean, respectively. Each cell presents the mean and standard deviation with the combined  $5 \times 2$  CV  $F$ -test p-value against MP-CCCD in parentheses.

MP-CCCD achieves the highest mean AUC on Ionosphere, Sonar, Vowel, Shuttle, and Segment. On Ionosphere, its advantage over RW-CCCD ( $p = 0.043$ ) and BP-CCCD ( $p = 0.014$ ) is statistically significant, but not for P-CCCD ( $p = 0.055$ ). Across all datasets where MP-CCCD was not the highest performer, none of the other model differences were significant at a level of significance of 0.05.

For  $F_1$ -score, MP-CCCD is the highest performing model across all datasets. It was significantly higher than RW-CCCD for Ionosphere ( $p = 0.024$ ), Vowel ( $p < 0.001$ ), Segment ( $p = 0.003$ ), and PageBlocks ( $p = 0.003$ ). It was only significantly higher than P-CCCD for Ionosphere ( $p < 0.001$ ) and Vowel ( $p < 0.001$ ). Under the G-mean metric, MP-CCCD is the highest on Ionosphere, Sonar, Yeast6, Vowel, Shuttle, Segment, and PageBlocks. The only significant difference was versus RW-CCCD on Yeast6 ( $p = 0.019$ ). Notably, MP-CCCD’s exceptional  $F_1$  performance on Abalone contrasts with its low G-mean, which supports findings in the simulation study that MP-CCCD favors minority class prediction at the expense of majority class classification.

BP-CCCD never attains the sole highest mean on any real dataset under any metric, as it tied for highest performing with MP-CCCD and RW-CCCD on  $F_1$ -score for Abalone. However, BP-CCCD had a score similar to random guessing for AUC and G-mean on the Abalone dataset, but a corresponding high  $F_1$ -score. BP-CCCD might tend to bias selection



towards the minority class by over estimating it. Yeast6, Yeast4, Yeast1289vs7, and Abalone all had particularly poor performances, with these datasets representing the highest with global and local imbalances among the selected. Given the stark contrast to the synthetic settings, we believe that BP-CCCD either

- (i) requires additional tuning (i.e., increasing the number of bagging iterations above 30) due to the complex structures of the real data class supports, or
- (ii) BP-CCCD’s strength lies in settings with clear, well-defined class structures rather than noisy, heterogeneous data.

For completeness we extended the tuning of BP-CCCD so that  $L = 2, \dots, 60$ , and reran the BP-CCCD model on the Yeast6, Yeast4, Yeast1289vs7, and Abalone datasets. There was no change in BP-CCCD performance under the extended parameter range. Given the conflict with the simulation results, its likely then that BP-CCCD works well only on well-behaved class supports and overfits on noisy, heterogeneous data.

We now provide a quick summary of the real data results,

- (i) BP-CCCD works well on simulated data, but performs poorly when used on real datasets.
- (ii) MP-CCCD and RW-CCCD were the two highest performing CCCD variants on real datasets for AUC and G-Mean.
- (iii) MP-CCCD was the highest performing on  $F_1$ -score, indicating that MP-CCCD is much better and minority class classification than other CCCD variants.

| Model                | Ionosphere        | Sonar             | Yeast6            | Yeast5            | Yeast4            | Yeast1289vs7      | Vowel             | Shuttle           | Abalone19         | Segment           | PageBlocks        |
|----------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|
| MP-CCCD              | 0.946 $\pm$ 0.025 | 0.820 $\pm$ 0.046 | 0.766 $\pm$ 0.039 | 0.853 $\pm$ 0.073 | 0.653 $\pm$ 0.030 | 0.638 $\pm$ 0.048 | 0.969 $\pm$ 0.035 | 0.991 $\pm$ 0.013 | 0.517 $\pm$ 0.021 | 0.963 $\pm$ 0.008 | 0.869 $\pm$ 0.009 |
| RW-CCCD              | 0.904 $\pm$ 0.021 | 0.730 $\pm$ 0.055 | 0.834 $\pm$ 0.054 | 0.858 $\pm$ 0.077 | 0.771 $\pm$ 0.056 | 0.657 $\pm$ 0.067 | 0.870 $\pm$ 0.051 | 0.947 $\pm$ 0.105 | 0.622 $\pm$ 0.049 | 0.91 $\pm$ 0.025  | 0.875 $\pm$ 0.019 |
| (p-value vs MP-CCCD) | (0.043)           | (0.247)           | (0.313)           | (0.831)           | (0.113)           | (0.172)           | (0.049)           | (0.564)           | (0.074)           | (0.088)           | (0.690)           |
| P-CCCD               | 0.926 $\pm$ 0.028 | 0.795 $\pm$ 0.052 | 0.755 $\pm$ 0.054 | 0.792 $\pm$ 0.095 | 0.598 $\pm$ 0.051 | 0.564 $\pm$ 0.027 | 0.965 $\pm$ 0.031 | 0.988 $\pm$ 0.016 | 0.539 $\pm$ 0.037 | 0.957 $\pm$ 0.010 | 0.869 $\pm$ 0.009 |
| (p-value vs MP-CCCD) | (0.055)           | (0.144)           | (0.524)           | (0.071)           | (0.249)           | (0.177)           | (0.713)           | (0.483)           | (0.611)           | (0.287)           | (0.535)           |
| BP-CCCD              | 0.886 $\pm$ 0.028 | 0.722 $\pm$ 0.038 | 0.556 $\pm$ 0.045 | 0.752 $\pm$ 0.043 | 0.502 $\pm$ 0.006 | 0.500 $\pm$ 0.000 | 0.621 $\pm$ 0.027 | 0.989 $\pm$ 0.014 | 0.500 $\pm$ 0.000 | 0.865 $\pm$ 0.026 | 0.502 $\pm$ 0.014 |
| (p-value vs MP-CCCD) | (0.014)           | (0.176)           | (0.002)           | (0.370)           | (0.003)           | (0.028)           | (<0.001)          | (0.582)           | (0.114)           | (0.001)           | (<0.001)          |

Table 3.6: Mean and standard deviation of MP-CCCD and CCCD models' AUC across selected real data. Values highlighted in red indicate the best performer of all models for a specific dataset. The results of the relevant  $5 \times 2$  CV  $F$ -test against MP-CCCD are presented in parenthesis when relevant.

| Model                | Ionosphere    | Sonar         | Yeast6        | Yeast5        | Yeast4        | Yeast1289vs7  | Vowel         | Shuttle       | Abalone19     | Segment       | PageBlocks    |
|----------------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|
| MP-CCCD              | 0.960 ± 0.017 | 0.810 ± 0.048 | 0.582 ± 0.072 | 0.693 ± 0.076 | 0.358 ± 0.055 | 0.303 ± 0.102 | 0.956 ± 0.044 | 0.983 ± 0.016 | 0.996 ± 0.000 | 0.931 ± 0.013 | 0.788 ± 0.010 |
| RW-CCCD              | 0.924 ± 0.016 | 0.726 ± 0.055 | 0.450 ± 0.065 | 0.574 ± 0.088 | 0.334 ± 0.045 | 0.177 ± 0.048 | 0.628 ± 0.067 | 0.834 ± 0.308 | 0.996 ± 0.000 | 0.706 ± 0.035 | 0.676 ± 0.199 |
| (p-value vs MP-CCCD) | (0.024)       | (0.333)       | (0.170)       | (0.384)       | (0.783)       | (0.252)       | (<0.001)      | (0.524)       | (0.535)       | (0.003)       | (0.003)       |
| P-CCCD               | 0.947 ± 0.017 | 0.777 ± 0.057 | 0.556 ± 0.089 | 0.600 ± 0.132 | 0.250 ± 0.101 | 0.165 ± 0.073 | 0.374 ± 0.484 | 0.978 ± 0.020 | 0.994 ± 0.001 | 0.927 ± 0.012 | 0.786 ± 0.011 |
| (p-value vs MP-CCCD) | (<0.001)      | (0.027)       | (0.396)       | (0.076)       | (0.338)       | (0.208)       | (<0.001)      | (0.664)       | (0.054)       | (0.281)       | (0.535)       |
| BP-CCCD              | 0.940 ± 0.014 | 0.661 ± 0.045 | 0.182 ± 0.136 | 0.590 ± 0.074 | 0.006 ± 0.020 | 0.000 ± 0.000 | 0.144 ± 0.192 | 0.976 ± 0.016 | 0.996 ± 0.000 | 0.809 ± 0.025 | 0.726 ± 0.017 |
| (p-value vs MP-CCCD) | (0.039)       | (0.052)       | (0.003)       | (0.439)       | (0.002)       | (0.024)       | (<0.001)      | (0.772)       | (0.535)       | (0.001)       | (0.008)       |

Table 3.7: Mean and standard deviation of MP-CCCD and CCCD models'  $F_1$ -scores across selected real data. Values highlighted in red indicate the best performer of all models for a specific dataset. The results of the relevant  $5 \times 2$  CV  $F$ -test against MP-CCCD are presented in parenthesis when relevant.

| Model                | Ionosphere    | Sonar         | Yeast6        | Yeast5        | Yeast4        | Yeast1289vs7  | Vowel         | Shuttle       | Abalone19     | Segment       | PageBlocks    |
|----------------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|
| MP-CCCD              | 0.945 ± 0.025 | 0.824 ± 0.039 | 0.730 ± 0.055 | 0.697 ± 0.096 | 0.573 ± 0.058 | 0.526 ± 0.110 | 0.968 ± 0.036 | 0.991 ± 0.013 | 0.135 ± 0.146 | 0.963 ± 0.008 | 0.863 ± 0.011 |
| RW-CCCD              | 0.904 ± 0.021 | 0.724 ± 0.057 | 0.393 ± 0.165 | 0.845 ± 0.098 | 0.752 ± 0.070 | 0.599 ± 0.117 | 0.866 ± 0.055 | 0.938 ± 0.131 | 0.609 ± 0.059 | 0.908 ± 0.025 | 0.677 ± 0.200 |
| (p-value vs MP-CCCD) | (0.048)       | (0.140)       | (0.019)       | (0.389)       | (0.076)       | (0.071)       | (0.057)       | (0.556)       | (0.004)       | (0.076)       | (0.001)       |
| P-CCCD               | 0.925 ± 0.028 | 0.793 ± 0.053 | 0.715 ± 0.078 | 0.758 ± 0.129 | 0.450 ± 0.116 | 0.403 ± 0.093 | 0.964 ± 0.032 | 0.988 ± 0.016 | 0.272 ± 0.229 | 0.956 ± 0.011 | 0.862 ± 0.011 |
| (p-value vs MP-CCCD) | (0.062)       | (0.059)       | (0.530)       | (0.317)       | (0.359)       | (0.548)       | (0.754)       | (0.485)       | (0.579)       | (0.284)       | (0.535)       |
| BP-CCCD              | 0.888 ± 0.029 | 0.696 ± 0.042 | 0.293 ± 0.134 | 0.692 ± 0.068 | 0.039 ± 0.083 | 0.000 ± 0.000 | 0.458 ± 0.041 | 0.986 ± 0.013 | 0.000 ± 0.000 | 0.858 ± 0.026 | 0.797 ± 0.017 |
| (p-value vs MP-CCCD) | (0.017)       | (0.058)       | (0.008)       | (0.809)       | (0.002)       | (0.003)       | (<0.001)      | (0.610)       | (0.043)       | (0.002)       | (0.004)       |

Table 3.8: Mean and standard deviation of MP-CCCD and CCCD models' G-mean across selected real data. Values highlighted in red indicate the best performer of all models for a specific dataset. The results of the relevant  $5 \times 2$  CV  $F$ -test against MP-CCCD are presented in parenthesis when relevant.

### 3.5 Conclusions and Discussions

This chapter discusses enhancements to P-CCCD classifiers to improve performance under challenging conditions of class overlap and imbalance. The modified P-CCCD (MP-CCCD) incorporates a cardinality-based scoring mechanism that weights classification decisions by the number of points within each covering ball, controlled. Additionally, we explore bagged P-CCCD (BP-CCCD) classifiers to enhance robustness and reduce variance, particularly beneficial in high-overlap scenarios with balanced data characteristics. MP-CCCD is proven to be consistent under strictly separable class conditional distributions. MP-CCCD requires approximately  $20\times$  more computation time than P-CCCD due to additional cardinality calculations, while BP-CCCD requires about  $14\times$  more due to ensemble iterations. It should be noted, however, that the cost for BP-CCCD can be potentially lowered to the same as P-CCCD by computing each bagged model in parallel. We close the chapter with our recommendations for when to use MP-CCCD and BP-CCCD models.

- (i) We recommend MP-CCCD over RW-CCCD and P-CCCD in all settings, despite its increased computational time. Incorporating local geometric information through cardinality-based scoring provides a more flexible and generalizable approach to classification, especially in severe overlapping and imbalanced scenarios. MP-CCCD was never outperformed in simulation settings by RW-CCCD or P-CCCD.
- (ii) BP-CCCD shows advantages over MP-CCCD, P-CCCD, and RW-CCCD in balanced, lower dimensional scenarios where supports are well-behaved. However, its poor performance under real datasets that are noisy and heterogeneous makes it difficult to recommend BP-CCCD. This might be potentially why no formal literature on BP-CCCD has ever been written despite the code for BP-CCCD.

## Chapter 4

### A Hybrid Approach to Classification with CCCDs

In this chapter, we discuss a hybrid classification model using CCCDs. We first identify potential regions of soft- and hard-overlapping for classes using support vector concepts, and use a combination of SVM and RW-CCCD classifier rules depending on which region the data is in. We chose the selection of RW-CCCD classifiers due to the generally improved classification performance in extreme overlapped settings comparative to other CCCDs and strong classifiers [76].

#### 4.1 Support Vector Class Cover Catch Digraph (SV-CCCD)

In this section, we detail our construction for the hybrid classifier. We define the following to help construct our regions of overlap

**Definition 4.1.1** ( $\delta$ -separable). Sets  $A$  and  $B$  are said to be  $\delta$ -separable if and only if  $\inf\{d(x, y) | x \in A, y \in B\} \geq \delta$ .

Our first task is to partition our data into  $\delta$ -separable estimates. We first consider a naïve implementation and discuss its relative merits.

Fix  $\delta > 0$ . The partition proceeds in the following way, calculate each class' center of mass and draw a line between them. A perpendicular bisecting hyperplane, denoted  $P$ , is constructed through the midpoint of the line. We define two parallel hyperplanes  $P_1$  and  $P_2$  on each side of  $P$  such that  $d(P_1, P_2) = \delta$ . The region between  $P_1$  and  $P_2$  by definition partitions the feature space into two sets that are  $\delta$ -separable.

We label the  $\delta$ -separable collection of observations as *soft-overlapping*, while the observations between the two hyperplanes are *hard-overlapping*. Figure (4.1) provides a visual

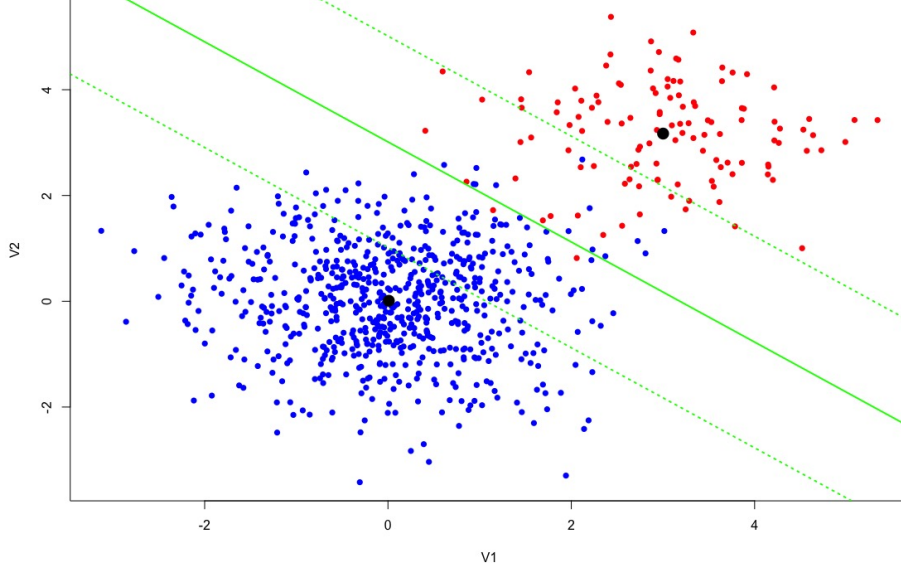


Figure 4.1: An example of the naïve  $\delta$ -separable construction for  $\delta = 2$ .

example of the naïve construction for a random dataset when  $\delta = 2$ . The black dots represent the centers of mass for the respective classes, with the solid green line representing the naïve hyperplane  $P$ . The dotted green lines are  $P_1$  and  $P_2$ . For the naïve strategy, employing separating hyperplanes requires calculating the mean of both classes which is done in  $\mathcal{O}(n + m)$  time. Assigning each point to either soft or hard overlap requires another loop which is again  $\mathcal{O}(n + m)$  time.

The naïve construction does not prioritize prior classification performance, and as such is unlikely to lead to the best possible performing hybrid classifier. Additionally, extending the construction to multi-class settings would be quite difficult as the traditional one-versus-rest approach could potentially bias the center of the union of non-target classes. Therefore, we need a way to generate a decision boundary that can be flexible even in multi-class settings.

We partition the space  $\mathbb{R}^d$  using a soft-margin SVM with Gaussian kernel defined in Equation (2.19) on the training data. The decision boundary becomes the decision hyperplane,  $P$ . We construct  $P_1$  and  $P_2$  such that the distance between them is  $\delta$ , where  $\delta$  is user defined. Using the newly constructed  $P$ , points in the hard-overlapping region are classified

---

**Algorithm 4** Support Vector–Class Cover Catch Digraph (SV-CCCD) Classifier

---

**Input:** Training data  $\mathcal{X}$ , query point  $z$ , bandwidth  $\delta$ , soft margin  $\lambda$ , kernel  $\beta$

**Output:** Predicted label  $y \in \mathcal{Y}$

1: Train Gaussian SVM on  $\mathcal{X}$  to obtain weights vector and bias  $(\vec{w}, b)$

2: Partition  $\mathcal{X}$  into:

$$\mathcal{X}_{\text{hard}} \leftarrow \{x_i \in \mathcal{X} : |\vec{w}^\top \beta(x_i) - b| < \delta/2\}$$

$$\mathcal{X}_{\text{soft}} \leftarrow \mathcal{X} \setminus \mathcal{X}_{\text{hard}}$$

3: Train  $C^{\text{RW}}$  on  $\mathcal{X}_{\text{hard}}$  (RW-CCCD classifier)

4: Train  $C^{\text{SVM}}$  on  $\mathcal{X}_{\text{soft}}$  (Gaussian SVM)

5: **if**  $|\vec{w}^\top \beta(z) - b| < \delta/2$  **then**

6:      $y \leftarrow C^{\text{RW}}(z)$

7: **else**

8:      $y \leftarrow C^{\text{SVM}}(z)$

9: **end if**

10: **return**  $y$

---

using RW-CCCD, while soft-overlapping points are classified using the constructed SVM boundary.

We refer to the resulting hybrid algorithm as support vector class cover catch digraph (SV-CCCD) algorithm. SV-CCCD is detailed more thoroughly in Algorithm (4). An added benefit to this method is the flexibility for SVM to incorporate class weights to adjust the decision boundary for imbalanced data. With class imbalance, fewer support vectors are chosen. However, RW-CCCDs still reduce the training data set more than SVMs, making them appealing options for classification in hard-overlapping regions where local imbalances might be large. While  $P$  may not achieve perfect separation of the training data by class, it is anticipated to yield a more accurate estimation of the support regions compared to the naïve method as  $P$  now prioritizes prior classification performance. If the SVM perfectly predicts the data, then the two classes are already  $\delta$ -separable and the algorithm defaults entirely to the SVM decision rule without incorporating RW-CCCD.

We focus initially on Gaussian kernels for simplicity, but acknowledge that kernel selection plays an important role in the hyperplane (and subsequent overlap region) construction. Figure (4.2) is a visual representation of the construction of the overlapping regions using SVM soft margin boundaries when  $\delta = 2$ .



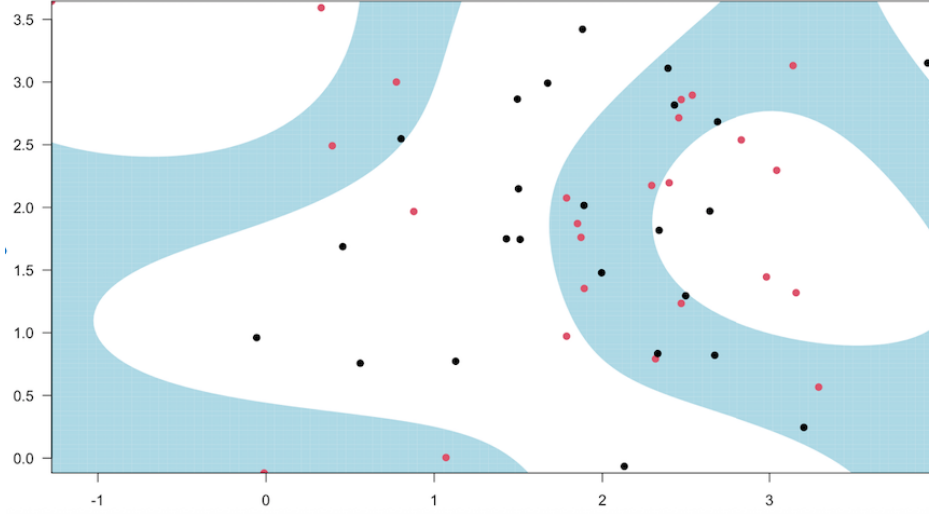


Figure 4.2: Construction of  $\delta$ -separable regions using SVM soft margin boundaries when  $\delta = 2$ .

SVM kernels allow for flexible decision boundary hyperplanes in multi-class settings. Prior knowledge of class support boundaries and shape are important in the multi-class setting to ensure that a correct kernel decision is made for the target class. The non-linear nature of kernel SVM boundaries creates decision boundaries that have complex geometric shapes able to handle multi-class classification in the feature space [132]. We advocate still, however, for a one-versus-rest approach. During pairwise comparisons, kernel functions can be changed to better handle complex geometric class supports.

## 4.2 Monte Carlo Simulations

In this section, we present similar experimental results for synthetic datasets used in Chapter 3 under different combinations of class overlap and imbalance settings for each of the three evaluation metrics. However, we will extend the comparison of our hybrid classifier to other strong classifiers. Table (4.1) provides a more detailed description of each classifier. Simulations were performed in R with self-written code for P-CCCD, MP-CCCD, and SV-CCCD classification; `cccd` for the RW-CCCD classification; `RWeka` for all C4.5 classifications;

DMwR2 for all k-NN classifications; e1071 for all SVM classifications; and c50 for the C5.0 classification [82, 53, 130, 97, 103, 104].

| Method   | Description                                                                                                                                                                                      |
|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| P-CCCD   | P-CCCD with the optimal $\tau^*$ (in the pilot study) among $\tau = 0.0, 0.1, \dots, 1.0$ .                                                                                                      |
| RW-CCCD  | RW-CCCD with the optimal $e^*$ (in the pilot study) among $e = 0.0, 0.1, \dots, 1.0$ .                                                                                                           |
| MP-CCCD  | MP-CCCD with the optimal $e^*$ (in the pilot study) among $e = 0.0, 0.1, \dots, 1.0$ and optimal $\tau^*$ (in the pilot study) among $\tau = 0, 0.1, \dots, 1.0$ .                               |
| SV-CCCD  | SV-CCCD with the optimal $e^*$ (in the pilot study) among $e = 0.0, 0.1, \dots, 1.0$ .                                                                                                           |
| $Sk$ -NN | $k$ -NN using a combination of SMOTE ( $t = 2$ and $k = 5$ ) and ENN ( $K = 3$ ) with optimal $k^*$ (in pilot study) among $k = 1, 2, \dots, 30$ .                                               |
| SSVM     | SVM using a combination of SMOTE ( $t = 2$ and $k = 5$ ) and ENN ( $K = 3$ ) with optimal $\gamma^*$ (in pilot study) among $\gamma = 0.1, 0.2, \dots, 3.9, 4.0$ .                               |
| SC4.5    | C4.5 using a combination of SMOTE ( $t = 2$ and $k = 5$ ) and ENN ( $K = 3$ ) with Laplace smoothing and no pruning.                                                                             |
| $Ek$ -NN | $k$ -NN using a combination of undersampling ( $t = 4$ ) and Adaboost ( $L = 10$ ) with optimal $k^*$ (in pilot study) among $k = 1, 2, \dots, 30$ .                                             |
| ESVM     | SVM using a combination of undersampling ( $t = 4$ ) and Adaboost ( $L = 10$ ) for support vector machines with optimal $\gamma^*$ (in pilot study) among $\gamma = 0.1, 0.2, \dots, 3.9, 4.0$ . |
| EC4.5    | C4.5 decision trees using a combination of undersampling ( $t = 4$ ) and Adaboost ( $L = 10$ ) with optimal $k^*$ (in pilot study) possible trees among $k = 1, 2, \dots, 30$ trees.             |
| $Ck$ -NN | A cost-sensitive version of $k$ -NN with optimal $k^*$ (in pilot study) among $k = 1, 2, \dots, 30$ .                                                                                            |
| CSVM     | A cost-sensitive version of SVM with optimal $\gamma^*$ (in pilot study) among $\gamma = 0.1, 0.2, \dots, 3.9, 4.0$ .                                                                            |
| C5.0     | C5.0 decision tree with no global pruning.                                                                                                                                                       |

Table 4.1: The description of other CCCD and strong classifier variants used for comparison against SV-CCCD in simulation studies.

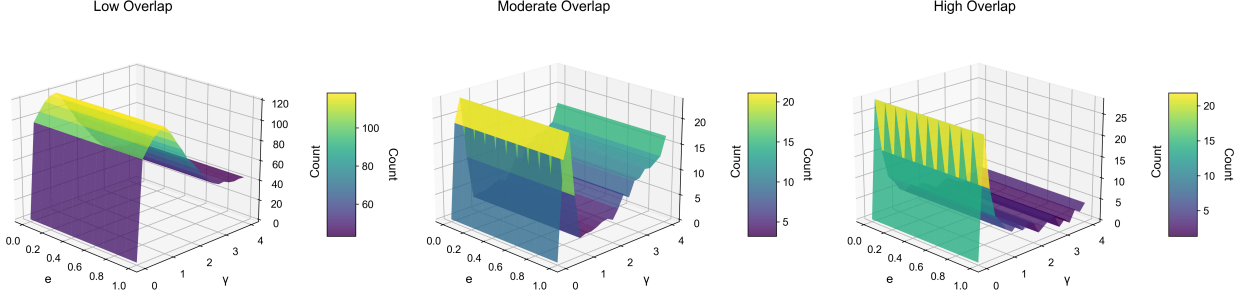


Figure 4.3: Surface plots of optimal AUC counts for each  $(e, \gamma)$  combination of SV-CCCD classifiers on balanced, overlapping simulations. Kernel parameter  $\gamma$  plays a bigger role in tuning for optimal AUC than  $e$ .

We define  $\delta^*$  for SV-CCCD to be the furthest misclassified point on each side of the SVM decision boundary. In the event that no points are misclassified  $\delta^* = 0$ , and the decision rule defaults to the trained SVM boundary.

#### 4.2.1 Performance Under Class Overlap Settings

To understand the effects of class overlap on SV-CCCD classification, we generated target class observations from  $F_{\mathcal{X}_0} = \text{Uniform}(0, 1)^d$  distribution and non-target class observations from  $F_{\mathcal{X}_1} = \text{Uniform}(\Delta, 1 + \Delta)^d$  distribution with  $n_0, n_1 = 500$  observations. We simulate test datasets until the standard error for the evaluation metric of all classifiers is below 0.0005 for the Monte Carlo simulations. We measure class overlap the same as in Chapter 3 using Equation 3.6.

We first look at how selection of optimal  $e^*$  and  $\gamma^*$  change across different overlap levels and evaluation metrics. Figure (4.3) displays the optimal AUC counts 3D surface plot for each  $(e, \gamma)$  combination across all three class overlap levels. The RW-CCCD score parameter  $e$ , is consistent across low-overlap regions, where most of the data falls into soft-overlapping regions. Interestingly, in regions of higher overlap, we see that all values of tested  $e$  achieve the maximum optimal AUC count. For both low and high overlap, lower values of  $\gamma$  are selected more frequently on average as optimal. Across all overlap levels,  $\gamma$  has a more

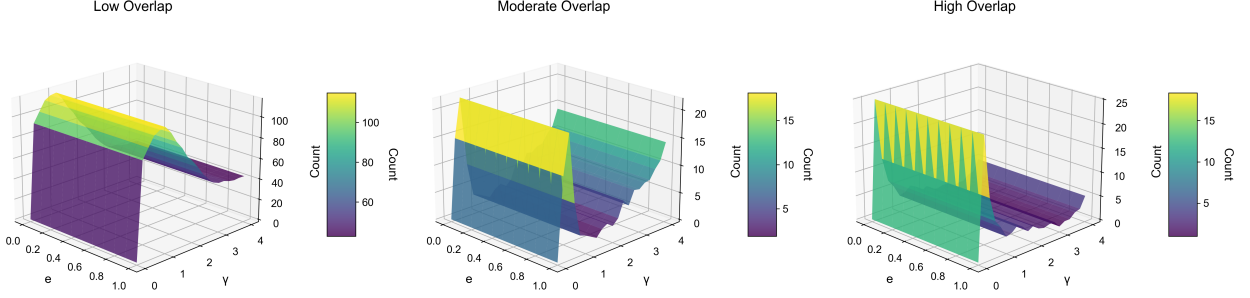


Figure 4.4: Surface plots of optimal  $F_1$ -score counts for each  $(e, \gamma)$  combination of SV-CCCD classifiers on balanced, overlapping simulations. Kernel parameter  $\gamma$  plays a bigger role in tuning for optimal  $F_1$ -score than  $e$ .

pronounced and consistent influence on optimal AUC performance than  $e$  in the SV-CCCD framework.

Parameter  $\gamma$  plays a central role in controlling the complexity of the SVM decision boundary. Smaller values lead to broader kernels, allowing distant points to influence the margin, thus promoting smoother decision boundaries [50]. This is especially advantageous in high-overlap settings where noise and class ambiguity are intrinsic and overfitting becomes a critical risk. We observe comparable selection patterns of the optimal  $(e^*, \gamma^*)$  values when using  $F_1$ -score and G-mean as evaluation criteria. Figures (4.4) and (4.5) display the respective 3D surface plots. Class overlap affects the geometry of the decision margin by increasing the number of support vectors, which is why tuning for  $\gamma$  becomes more important in such scenarios.

Table (4.2) presents pairwise mean differences between SV-CCCD and each alternative strong classifier under the AUC,  $F_1$ -score, and G-mean evaluation metrics. Positive values indicate that SV-CCCD outperforms the competitor, while negative values indicate the opposite. SV-CCCD performed better than other CCCD variants, with the mean difference becoming larger as the overlap increases. However, none of the differences are practically significant. Against other strong classifier variants, the mean difference against SV-CCCD was positive across all metrics, with the exception of EC4.5 and C5.0 for moderate overlap

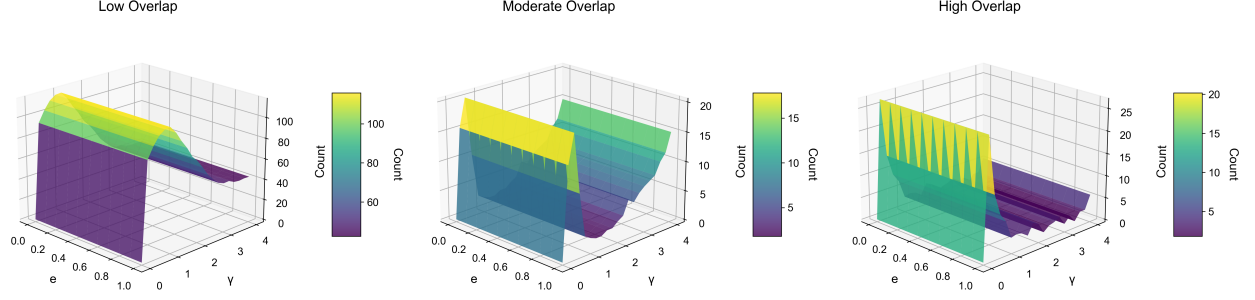


Figure 4.5: Surface plots of optimal G-mean counts for each  $(e, \gamma)$  combination of SV-CCCD classifiers on balanced, overlapping simulations. Kernel parameter  $\gamma$  plays a bigger role in tuning for optimal G-mean than  $e$ .

settings. Again, none of the differences are practically significant. In settings of class overlap, the initial SVM decision bound is prone to making errors, which in turn allows for more training points of RW-CCCD classifiers and ultimately a better generalization of SV-CCCD compared to other classifiers. Higher SV-CCCD mean differences implies the model is at least as good as the other models in balanced, overlapping settings with the potential to be better under certain underlying data geometries.

| Overlap Category | Metric       | vs MP-CCCD | vs P-CCCD | vs RW-CCCD | vs Sk-NN | vs SSVM | vs SC4.5 | vs Ek-NN | vs ESVM | vs EC4.5 | vs Ck-NN | vs CSVM | vs C5.0 |
|------------------|--------------|------------|-----------|------------|----------|---------|----------|----------|---------|----------|----------|---------|---------|
| Low Overlap      | AUC          | 0.003      | 0.002     | 0.003      | 0.001    | 0.001   | 0.007    | 0.002    | 0.001   | 0.000    | 0.002    | 0.001   | 0.006   |
|                  | $F_1$ -Score | 0.003      | 0.002     | 0.002      | 0.001    | 0.001   | 0.007    | 0.002    | 0.001   | 0.000    | 0.002    | 0.001   | 0.006   |
|                  | G-Mean       | 0.003      | 0.002     | 0.002      | 0.001    | 0.001   | 0.007    | 0.002    | 0.001   | 0.000    | 0.002    | 0.001   | 0.006   |
| Moderate Overlap | AUC          | 0.017      | 0.015     | 0.011      | 0.008    | 0.006   | 0.000    | 0.021    | 0.006   | -0.011   | 0.008    | 0.004   | -0.003  |
|                  | $F_1$ -Score | 0.018      | 0.016     | 0.011      | 0.009    | 0.006   | 0.001    | 0.021    | 0.005   | -0.011   | 0.006    | 0.004   | -0.002  |
|                  | G-Mean       | 0.017      | 0.016     | 0.011      | 0.008    | 0.006   | 0.001    | 0.022    | 0.006   | -0.011   | 0.007    | 0.004   | -0.002  |
| High Overlap     | AUC          | 0.022      | 0.019     | 0.018      | 0.017    | 0.010   | 0.005    | 0.027    | 0.007   | 0.004    | 0.015    | 0.007   | 0.004   |
|                  | $F_1$ -Score | 0.026      | 0.021     | 0.020      | 0.019    | 0.011   | 0.011    | 0.028    | 0.008   | 0.006    | 0.016    | 0.009   | 0.015   |
|                  | G-Mean       | 0.024      | 0.019     | 0.019      | 0.017    | 0.011   | 0.016    | 0.027    | 0.007   | 0.003    | 0.015    | 0.008   | 0.029   |

Table 4.2: Performance difference of SV-CCCD versus CCCD and other strong classifiers for balanced data across different scenarios of class overlap and evaluation metrics. Positive differences indicate SV-CCCD performed better.

### 4.2.2 Performance Under Various Class Imbalance and Overlap Settings

To understand the effects of class imbalance and class overlap on SV-CCCD classification, we generated majority class observations from  $F_{\mathcal{X}_0} = \text{Uniform}(0, 1)^d$  distribution and minority class observations from  $F_{\mathcal{X}_1} = \text{Uniform}(\Delta, 1 + \Delta)^d$  distribution with  $n_0 = 500$  and  $n_1 = 500q$  where  $q = 0.25, 0.50, 0.75$  observations. We simulate test datasets until the standard error for the evaluation metric of all classifiers is below 0.0005 for the Monte Carlo simulations. Under these settings, we used CSVM to construct the a more robust initial separating hyperplane,  $P$ .

Even under class imbalance, the selection of  $e^*$  and  $\gamma^*$  for optimal AUC remains consistent with the findings in the previous section. Under AUC, tuning for  $e$  is less important than tuning for  $\gamma$  even under imbalanced settings. This is due to all values of  $e$  corresponding to the same optimal AUC count for a fixed  $\gamma$ . Figure 4.6 provides the average optimal AUC counts across  $\gamma$ . SV-CCCD favors lower  $\gamma^*$  across all imbalance levels. Higher  $\gamma$  makes the decision boundary sensitive to individual support vectors. When the majority class dominates, it becomes difficult to create generalizable decision regions compared to lower  $\gamma$  kernel values where the boundary becomes flexible. Under both  $F_1$ -score and G-mean evaluation, class imbalance introduces a distinct shift in optimal parameter selection.

Figures (4.7) and (4.8) display the frequency of selected  $\gamma$  values for  $F_1$ -score and G-mean, respectively. Under both metrics, when class overlap is high higher  $\gamma^*$  are preferred as classes become more imbalanced. In high-overlap regions, a smooth boundary from low  $\gamma$  may include too many false positives, which are penalized under  $F_1$ -score and G-mean. Higher  $\gamma$  tightens the decision boundary, potentially improving precision. Conversely, a wide kernel from lower  $\gamma$  may preserve sensitivity, but hurt specificity even at the cost of smoothness or overall fit. This makes tuning  $\gamma$  sensitive to not only class overlap and imbalance, but also the metric being considered for evaluation. In all scenarios, the selection of  $e$  remains uniform.

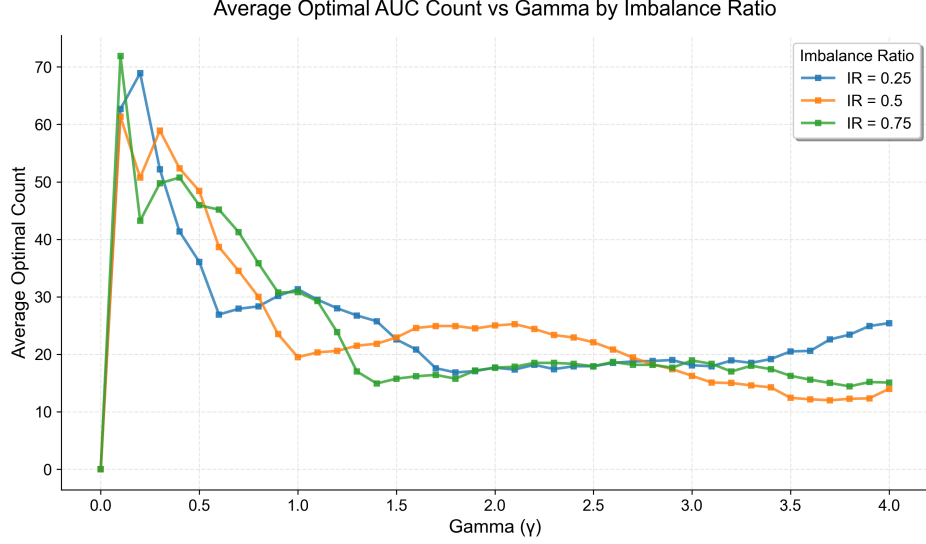


Figure 4.6: Average optimal AUC counts for  $\gamma$  kernel parameter in SV-CCCD across different imbalance levels.

Table (5.3) compares SV-CCCD performance against selected CCCD and strong classifier variants across different levels of class imbalance, class overlap, and evaluation metrics. Positive values indicate improved SV-CCCD performance for the respective metric. There was no practically significant differences between SV-CCCD and other classifiers under low and moderate imbalance settings. In high imbalance settings with moderate overlap, SV-CCCD performed significantly higher on  $F_1$ -scores  $Sk$ -NN, SC4.5, and  $Ek$ -NN variants. In high imbalance settings with high overlap, SV-CCCD performed significantly higher on  $F_1$ -scores than all other variants, and had a significantly higher G-mean than MP-CCCD. This might indicate that SV-CCCD is more effective at classifying the minority class observations. The absence of a corresponding increase in G-mean suggests that this improvement may come at the cost of reduced accuracy for the majority class, making SV-CCCD overly aggressive in its classification of minority class observations.



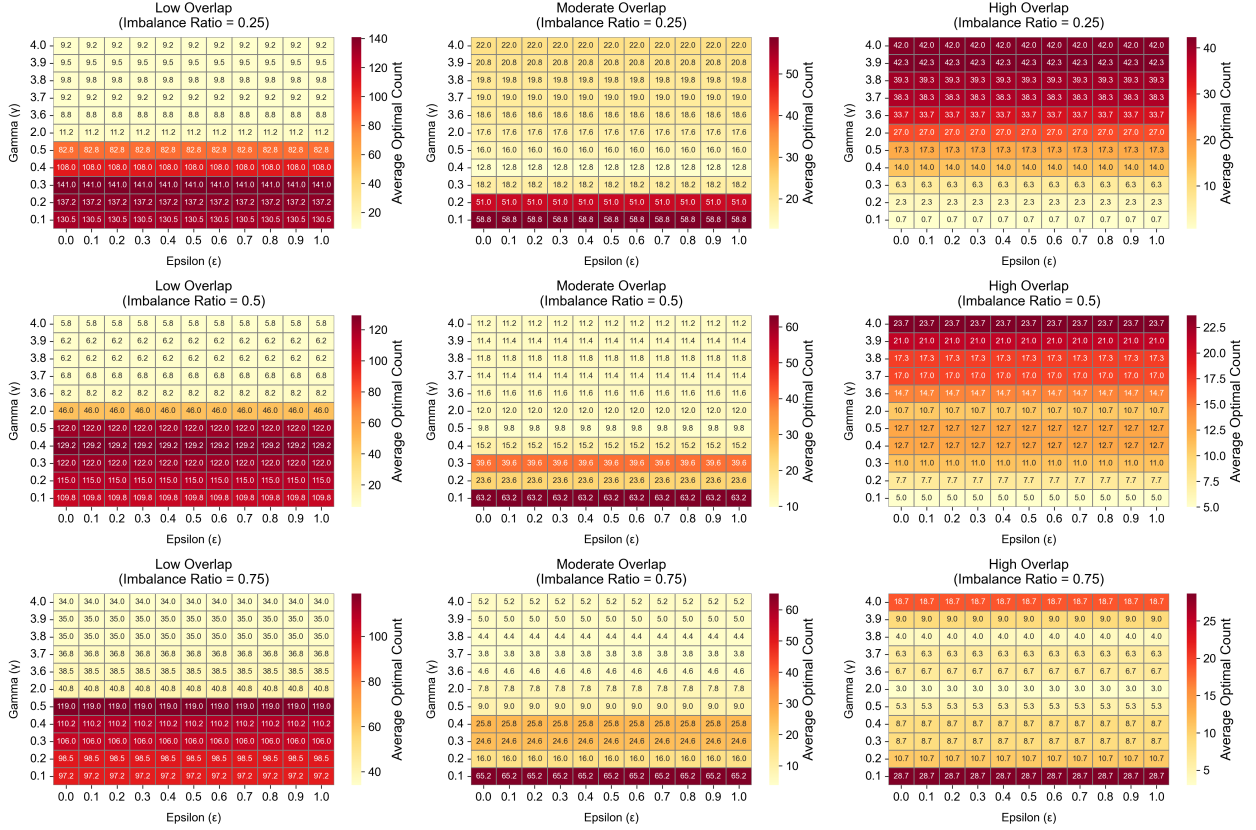


Figure 4.7: Heatmaps of the average optimal  $F_1$ -score counts for each  $(\epsilon, \gamma)$  combination of SV-CCCD classifiers on imbalanced, overlapping simulations. In high overlap and imbalance settings, larger  $\gamma$  are more often selected as optimal.

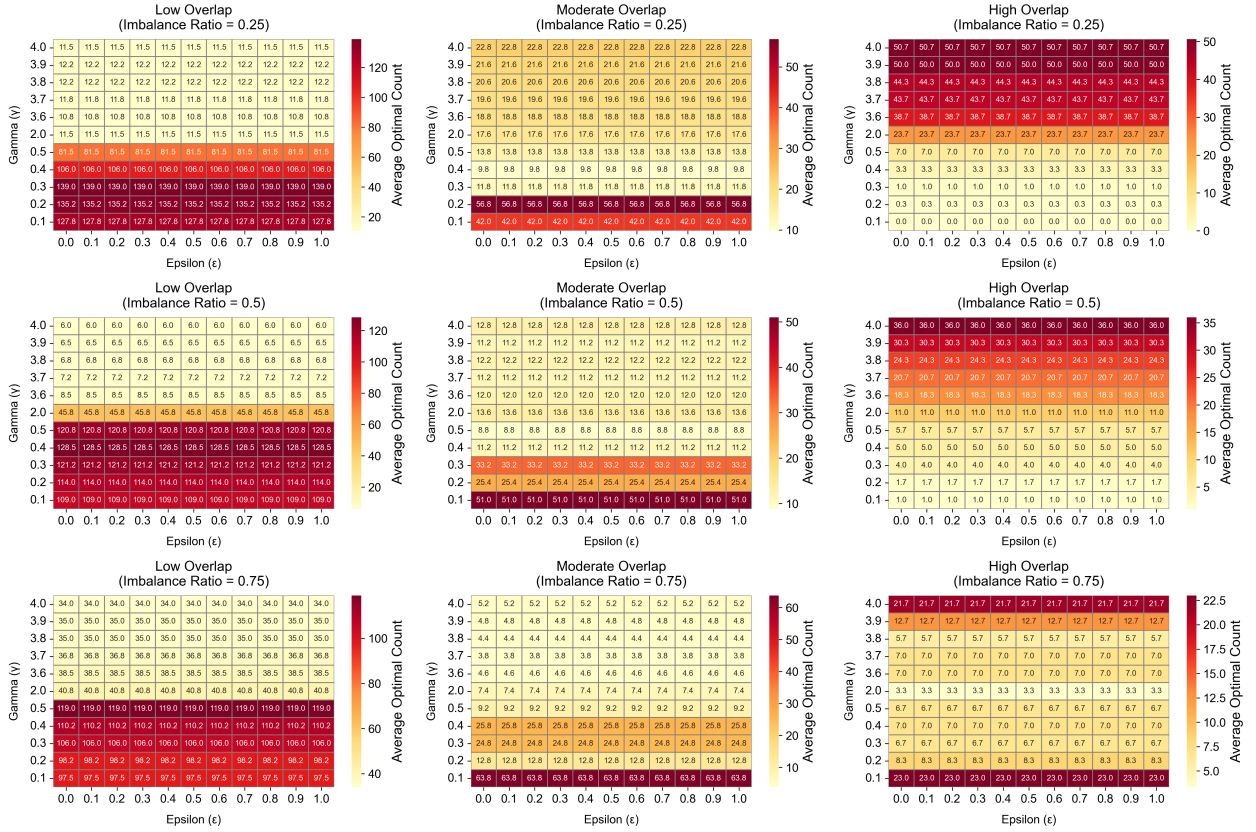


Figure 4.8: Heatmaps of the average optimal G-mean counts for each  $(\epsilon, \gamma)$  combination of SV-CCCD classifiers on imbalanced, overlapping simulations. In high overlap and imbalance settings, larger  $\gamma$  are more often selected as optimal.

| Imbalance          | Alpha            | Metric       | vs MP-CCCD   | vs P-CCCD    | vs RW-CCCD   | vs Sk-NN     | vs SSVM      | vs SC4.5     | vs Ek-NN     | vs ESVM      | vs EC4.5     | vs Ck-NN     | vs CSVM      | vs C5.0      |
|--------------------|------------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| Low Imbalance      | Low Overlap      | AUC          | 0.001        | 0.002        | 0.002        | 0.002        | 0.001        | 0.009        | 0.002        | 0.001        | 0.000        | 0.001        | 0.001        | 0.008        |
|                    |                  | $F_1$ -Score | 0.001        | 0.002        | 0.002        | 0.002        | 0.001        | 0.010        | 0.002        | 0.001        | 0.000        | 0.002        | 0.001        | 0.010        |
|                    |                  | G-Mean       | 0.001        | 0.002        | 0.002        | 0.002        | 0.001        | 0.009        | 0.002        | 0.001        | 0.000        | 0.001        | 0.001        | 0.008        |
|                    | Moderate Overlap | AUC          | 0.020        | 0.017        | 0.012        | 0.009        | 0.005        | -0.000       | 0.021        | 0.005        | -0.011       | 0.007        | 0.004        | -0.004       |
|                    |                  | $F_1$ -Score | 0.023        | 0.020        | 0.014        | 0.011        | 0.007        | 0.001        | 0.024        | 0.008        | -0.012       | 0.009        | 0.005        | -0.003       |
|                    |                  | G-Mean       | 0.020        | 0.017        | 0.011        | 0.009        | 0.005        | 0.001        | 0.020        | 0.005        | -0.011       | 0.007        | 0.003        | -0.003       |
|                    | High Overlap     | AUC          | 0.028        | 0.021        | 0.019        | 0.016        | 0.011        | 0.007        | 0.028        | 0.008        | 0.002        | 0.014        | 0.007        | 0.006        |
|                    |                  | $F_1$ -Score | 0.035        | 0.025        | 0.015        | 0.015        | 0.007        | 0.005        | 0.029        | 0.006        | 0.001        | 0.014        | 0.006        | 0.008        |
|                    |                  | G-Mean       | 0.029        | 0.020        | 0.018        | 0.016        | 0.011        | 0.014        | 0.027        | 0.007        | 0.001        | 0.013        | 0.007        | 0.029        |
|                    |                  |              |              |              |              |              |              |              |              |              |              |              |              |              |
| Moderate Imbalance | Low Overlap      | AUC          | 0.003        | 0.002        | 0.003        | 0.002        | 0.001        | 0.011        | 0.002        | 0.001        | -0.000       | 0.001        | 0.001        | 0.012        |
|                    |                  | $F_1$ -Score | 0.004        | 0.003        | 0.004        | 0.003        | 0.002        | 0.016        | 0.004        | 0.002        | 0.001        | 0.003        | 0.002        | 0.017        |
|                    |                  | G-Mean       | 0.003        | 0.002        | 0.003        | 0.002        | 0.001        | 0.012        | 0.002        | 0.001        | -0.000       | 0.002        | 0.001        | 0.012        |
|                    | Moderate Overlap | AUC          | 0.021        | 0.015        | 0.009        | 0.008        | 0.003        | -0.001       | 0.017        | 0.004        | -0.015       | 0.004        | 0.002        | -0.002       |
|                    |                  | $F_1$ -Score | 0.030        | 0.022        | 0.019        | 0.022        | 0.014        | 0.008        | 0.034        | 0.015        | -0.012       | 0.011        | 0.009        | 0.003        |
|                    |                  | G-Mean       | 0.022        | 0.016        | 0.009        | 0.008        | 0.004        | -0.001       | 0.017        | 0.005        | -0.015       | 0.005        | 0.002        | -0.002       |
|                    | High Overlap     | AUC          | 0.034        | 0.022        | 0.017        | 0.017        | 0.008        | 0.005        | 0.025        | 0.006        | 0.001        | 0.013        | 0.005        | 0.003        |
|                    |                  | $F_1$ -Score | 0.046        | 0.032        | 0.028        | 0.027        | 0.020        | 0.017        | 0.040        | 0.015        | 0.008        | 0.021        | 0.013        | 0.008        |
|                    |                  | G-Mean       | 0.035        | 0.024        | 0.016        | 0.016        | 0.008        | 0.012        | 0.022        | 0.003        | -0.002       | 0.009        | 0.003        | 0.023        |
|                    |                  |              |              |              |              |              |              |              |              |              |              |              |              |              |
| High Imbalance     | Low Overlap      | AUC          | 0.002        | 0.002        | 0.007        | 0.001        | 0.001        | 0.013        | 0.002        | 0.001        | 0.001        | 0.002        | 0.000        | 0.017        |
|                    |                  | $F_1$ -Score | 0.004        | 0.004        | 0.007        | 0.009        | 0.007        | 0.030        | 0.010        | 0.005        | 0.003        | 0.006        | 0.003        | 0.031        |
|                    |                  | G-Mean       | 0.002        | 0.002        | 0.007        | 0.001        | 0.001        | 0.013        | 0.002        | 0.001        | 0.001        | 0.002        | 0.000        | 0.017        |
|                    | Moderate Overlap | AUC          | 0.021        | 0.017        | 0.008        | 0.002        | -0.002       | 0.001        | 0.009        | -0.002       | -0.021       | -0.003       | -0.004       | -0.004       |
|                    |                  | $F_1$ -Score | 0.036        | 0.031        | 0.041        | <b>0.064</b> | 0.047        | <b>0.052</b> | <b>0.073</b> | 0.048        | 0.006        | 0.023        | 0.029        | 0.025        |
|                    |                  | G-Mean       | 0.023        | 0.018        | 0.008        | -0.000       | -0.003       | -0.001       | 0.008        | -0.003       | -0.022       | -0.003       | -0.005       | -0.005       |
|                    | High Overlap     | AUC          | 0.041        | 0.024        | 0.014        | 0.015        | 0.008        | 0.009        | 0.019        | 0.001        | -0.008       | 0.003        | -0.001       | -0.009       |
|                    |                  | $F_1$ -Score | <b>0.078</b> | <b>0.055</b> | <b>0.100</b> | <b>0.100</b> | <b>0.097</b> | <b>0.099</b> | <b>0.112</b> | <b>0.082</b> | <b>0.067</b> | <b>0.072</b> | <b>0.071</b> | <b>0.061</b> |
|                    |                  | G-Mean       | <b>0.053</b> | 0.031        | 0.007        | 0.006        | 0.000        | 0.004        | 0.011        | -0.007       | -0.017       | -0.005       | -0.009       | 0.001        |
|                    |                  |              |              |              |              |              |              |              |              |              |              |              |              |              |

Table 4.3: Performance difference of SV-CCCD versus CCCD and other strong classifiers for imbalanced data across different scenarios of class overlap and evaluation metrics. Positive differences indicate SV-CCCD performed better.

SV-CCCD is a hybrid between SVMs and RW-CCCD. SVMs and RW-CCCDs both involve training times of  $\mathcal{O}(N^3 + N^2d)$  [76]. Since SV-CCCD is a sequential hybrid algorithm, the worst case complexity for SV-CCCD is also  $\mathcal{O}(N^3 + N^2d)$ . However, since both SVM and RW-CCCD act as prototype selection algorithms, practical runtimes should be quick. To evaluate its practical efficiency, we benchmarked the runtime of SV-CCCD against all selected classifiers under the previous run-time experimental settings. Runtimes for  $Sk$ -NN, SSVM, and SC4.5 algorithms include the preprocessing times of applying SMOTE+ENN. Each configuration was run for 200 iterations, and mean runtimes along with standard deviations were recorded.

Results are summarized in Table (4.4). SV-CCCD runtimes are significantly lower than other CCCD variants, SMOTE+ENN variants, and EasyEnsemble variants. Given its comparative performance to other strong classifiers, this makes SV-CCCD an appealing alternative to other state-of-the-art classifiers; especially when the data has high class overlap and imbalance.

| Index    | Mean $\pm$ SD         |
|----------|-----------------------|
| SV-CCCD  | 0.0611 $\pm$ 0.0273   |
| P-CCCD   | 0.1681 $\pm$ 0.1964   |
| RW-CCCD  | 0.1435 $\pm$ 0.1363   |
| MP-CCCD  | 3.4688 $\pm$ 5.2414   |
| $Sk$ -NN | 9.4283 $\pm$ 12.1143  |
| SSVM     | 11.7459 $\pm$ 11.8681 |
| SC4.5    | 10.0818 $\pm$ 13.8776 |
| $Ek$ -NN | 7.7427 $\pm$ 9.2585   |
| ESVM     | 0.8749 $\pm$ 0.9784   |
| EC4.5    | 0.4209 $\pm$ 0.5447   |
| $Ck$ -NN | 0.0492 $\pm$ 0.0139   |
| CSVM     | 0.0215 $\pm$ 0.0175   |
| C5.0     | 0.0217 $\pm$ 0.0109   |

Table 4.4: Aggregated mean and standard deviations for SV-CCCD and other selected strong classifiers runtimes across simulated data settings.

We provide a quick summary of the results from the Monte Carlo simulations,

- (i) SV-CCCD favors a lower kernel parameter  $\gamma^*$ . The model considers data points that are relatively far away from a support vector as still being relevant, which produces smoother decision boundary and  $\delta$ -separable regions.
- (ii) SV-CCCD shows practical significance in  $F_1$ -score against other state-of-the-art classifiers when class imbalance is high and class overlap is moderate to high.

### 4.3 Real Data Experiments

We now consider the performance of SV-CCCD on real data settings. We use the same datasets and methods considered in Chapter 3. Tables (4.5), (4.6), and (4.7) present the mean and standard deviation AUC,  $F_1$ -score, and G-mean respectively for SV-CCCD against other comparative CCCD and strong classifier variants. The results of the comparative  $5 \times 2$  CV  $F$ -test are presented in parenthesis. Values in blue represent the highest average performer out of CCCD variants, while values in red represent the highest average overall.

SV-CCCD had a higher average AUC than other CCCD variants for the Vowel and Pageblocks datasets. For the Vowel dataset, the result was significant against RW-CCCD ( $p = 0.0570$ ), and was not significant against the other CCCD variants for Pageblocks. MP-CCCD significantly outperformed SV-CCCD on the Ionosphere dataset ( $p = 0.0482$ ), RW-CCCD significantly outperformed SV-CCCD on the Yeast4 dataset ( $p = 0.0438$ ), and P-CCCD significantly outperformed SV-CCCD on the Segment dataset ( $p < 0.0001$ ).

SV-CCCD was never a top performer against the other strong classifiers for AUC. Among the top performing strong classifiers,  $Ek$ -NN significantly outperformed SV-CCCD for Yeast6 ( $p = 0.0113$ ), Yeast5 ( $p = 0.0087$ ), Yeast4 ( $p = 0.0006$ ), and Abalone ( $p = 0.0004$ ); EC4.5 significantly outperformed SV-CCCD for the Pageblocks dataset ( $p = 0.0092$ ); and  $Ck$ -NN significantly for Segment ( $p < 0.0001$ ).

SV-CCCD was the highest performing CCCD variant for  $F_1$ -score on the Vowel and Pageblocks datasets again. For Vowel, SV-CCCD significantly outperformed both RW-CCCD ( $p = 0.0014$ ) and MP-CCCD ( $p = 0.0282$ ). For Pageblocks, SV-CCCD significantly

outperformed RW-CCCD (0.0008). RW-CCCD significantly outperformed SV-CCCD on the Abalone dataset ( $p = 0.0053$ ) and was tied as the highest performing model. SV-CCCD performed significantly worse than all other models on the Segment dataset.

Lastly, SV-CCCD was the highest performing CCCD variant for G-mean on the Vowel and Pageblocks datasets, performing significantly better than RW-CCCD on both ( $p = 0.0638$ ,  $p = 0.0029$ ) respectively. MP-CCCD was the highest performing CCCD variant for Ionosphere, Sonar, Yeast6, Shuttle, and Segment datasets. It was also the highest performing of all classifiers for the Sonar dataset.

Overall, the results indicate that SV-CCCD has good discriminatory ability, but is aggressive in classifying the minority class. Both Vowel and Pageblocks are characterized by little to no class overlap. This indicates that SV-CCCD is constructing CSVM decision boundaries, and that the proposed region of hard-overlap is very separable but still contains some training observations. In the Abalone dataset SV-CCCD has a high  $F_1$ -score but not a corresponding high G-mean score. This indicates that SV-CCCD potentially overfits and does not generalize well on real data settings. We consider two additional variants of SV-CCCD

- (i) We retrain using  $\delta$  to be equal to the distance of the second largest misclassified point (instead of the absolute largest), and
- (ii) We pre-process the data to account for class overlap in the initialization of the boundary using SMOTE+ENN.

We present the new average and standard deviation evaluation metrics in the datasets where SV-CCCD performed worse compared to other strong classifiers (Ionosphere, Yeast6, Yeast4, Yeast1289vs7, and Abalone) in Table 4.8.

Both the modified  $\delta$  SV-CCCD and the SMOTE+ENN SV-CCCD showed improved performance across all datasets for AUC. Additionally, the variants showed substantial improvements for  $F_1$ -score and G-mean for the subset of real datasets. Abalone, while increased,

was still low for G-mean. It should be noted, however, that all SMOTE+ENN methods in Table (4.7) performed poorly for Abalone due to the local imbalance.

We provide a brief summary for the real data settings,

- (i) SV-CCCD performance is highly dependent on initialization of  $P$  and the definition of  $\delta$ .
- (ii) Overall, when considering noisy, heterogeneous data SV-CCCD usage should not be considered over current state-of-the-art models without first either doing a thorough exploration of the data to be able to more effectively define  $\delta$ , or preprocessing the data to provide a better initialization for  $P$ .

|                      | Ionosphere         | Sonar              | Yeast6             | Yeast5             | Yeast4             | Yeast1289vs7       | Vowel              | Shuttle            | Abalone            | Segment               | Pageblocks         |
|----------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|-----------------------|--------------------|
| SV-CCCD              | 0.906±0.003        | 0.736±0.056        | 0.648±0.048        | 0.713±0.068        | 0.589±0.033        | 0.591±0.058        | <b>0.983±0.033</b> | 0.984±0.012        | 0.506±0.015        | 0.590±0.011           | <b>0.906±0.016</b> |
| RW-CCCD              | 0.904±0.021        | 0.730±0.055        | <b>0.834±0.054</b> | <b>0.858±0.077</b> | <b>0.771±0.056</b> | <b>0.657±0.067</b> | 0.870±0.051        | 0.947±0.105        | <b>0.622±0.049</b> | 0.910±0.025           | 0.875±0.019        |
| (p-value vs SV-CCCD) | (0.4115)           | (0.5991)           | (0.0724)           | (0.2360)           | (0.0438)           | (0.1365)           | (0.0570)           | (0.5424)           | (0.1986)           | (<0.0001)             | (0.3981)           |
| P-CCCD               | 0.926±0.028        | 0.795±0.052        | 0.755±0.054        | 0.792±0.095        | 0.598±0.051        | 0.564±0.027        | 0.965±0.031        | 0.988±0.016        | 0.539±0.037        | <b>0.957±0.010</b>    | 0.869±0.009        |
| (p-value vs SV-CCCD) | (0.1333)           | (0.1998)           | (0.1486)           | (0.1735)           | (0.7601)           | (0.1022)           | (0.0833)           | (0.6501)           | (0.4837)           | ( <b>&lt;0.0001</b> ) | (0.1180)           |
| MP-CCCD              | <b>0.946±0.025</b> | <b>0.820±0.046</b> | 0.766±0.039        | 0.853±0.073        | 0.653±0.048        | 0.638±0.048        | 0.969±0.035        | <b>0.991±0.013</b> | 0.517±0.021        | 0.963±0.008           | 0.869±0.009        |
| (p-value vs SV-CCCD) | (0.0482)           | (0.1026)           | (0.0445)           | (0.0103)           | (0.1061)           | (0.2530)           | (0.2893)           | (0.4894)           | (0.3258)           | (<0.0001)             | (0.1182)           |
| Sk-NN                | 0.838±0.022        | 0.739±0.063        | 0.882±0.040        | 0.963±0.018        | 0.802±0.049        | 0.636±0.059        | 0.940±0.040        | 0.999±0.001        | 0.573±0.032        | 0.978±0.007           | 0.816±0.012        |
| (p-value vs SV-CCCD) | (0.0780)           | (0.5880)           | (0.0193)           | (0.0047)           | (0.0002)           | (0.3900)           | (0.0330)           | (0.4244)           | (0.1510)           | (<0.0001)             | (0.0103)           |
| SSVM                 | 0.948±0.010        | 0.636±0.045        | 0.864±0.032        | 0.947±0.031        | 0.803±0.025        | 0.680±0.043        | <b>0.988±0.016</b> | 0.998±0.001        | 0.522±0.017        | 0.987±0.004           | 0.856±0.011        |
| (p-value vs SV-CCCD) | (0.1557)           | (0.3685)           | (0.0220)           | (0.0078)           | (0.0010)           | (0.0837)           | (0.5551)           | (0.4162)           | (0.2548)           | (<0.0001)             | (0.0532)           |
| SC4.5                | 0.866±0.022        | 0.707±0.033        | 0.764±0.056        | 0.883±0.073        | 0.660±0.065        | 0.595±0.041        | 0.944±0.028        | 0.998±0.003        | 0.503±0.019        | 0.982±0.005           | 0.925±0.012        |
| (p-value vs SV-CCCD) | (0.2143)           | (0.3870)           | (0.2864)           | (0.0037)           | (0.2558)           | (0.0517)           | (0.2216)           | (0.5213)           | (0.3303)           | (<0.0001)             | (0.2071)           |
| Ek-NN                | 0.858±0.021        | 0.817±0.016        | <b>0.89±0.038</b>  | <b>0.965±0.004</b> | <b>0.865±0.022</b> | 0.750±0.040        | 0.965±0.008        | 0.996±0.004        | <b>0.731±0.031</b> | 0.957±0.004           | 0.914±0.006        |
| (p-value vs SV-CCCD) | (0.2120)           | (0.1749)           | (0.0113)           | (0.0087)           | (0.0006)           | (0.0219)           | (0.5132)           | (0.3770)           | (0.0004)           | (<0.0001)             | (0.5227)           |
| ESVM                 | 0.802±0.026        | 0.649±0.049        | 0.854±0.040        | 0.954±0.007        | 0.825±0.029        | 0.619±0.034        | 0.870±0.043        | 0.994±0.004        | 0.640±0.044        | 0.895±0.017           | 0.519±0.006        |
| (p-value vs SV-CCCD) | (0.0141)           | (0.2935)           | (0.0262)           | (0.0128)           | (0.0027)           | (0.4497)           | (0.0112)           | (0.4921)           | (0.0117)           | (<0.0001)             | (<0.0001)          |
| EC4.5                | 0.897±0.013        | 0.783±0.024        | 0.855±0.031        | 0.947±0.010        | 0.833±0.037        | 0.621±0.063        | 0.965±0.009        | 0.999±0.001        | 0.719±0.04         | 0.992±0.003           | <b>0.952±0.006</b> |
| (p-value vs SV-CCCD) | (0.6628)           | (0.4767)           | (0.0222)           | (0.0102)           | (0.0067)           | (0.6490)           | (0.4620)           | (0.3792)           | (0.0019)           | (<0.0001)             | (0.0092)           |
| Ck-NN                | 0.873±0.018        | <b>0.825±0.025</b> | 0.876±0.045        | 0.970±0.010        | 0.823±0.027        | 0.683±0.047        | 0.980±0.033        | 0.999±0.000        | 0.572±0.035        | <b>0.994±0.001</b>    | 0.889±0.01         |
| (p-value vs SV-CCCD) | (0.1931)           | (0.0815)           | (0.0277)           | (0.0058)           | (0.0001)           | (0.1038)           | (0.5210)           | (0.3882)           | (0.0224)           | ( <b>&lt;0.0001</b> ) | (0.3759)           |
| CSVM                 | <b>0.951±0.010</b> | 0.735±0.062        | 0.835±0.042        | 0.948±0.025        | 0.766±0.045        | <b>0.690±0.062</b> | 0.985±0.030        | 0.984±0.012        | 0.685±0.057        | 0.590±0.011           | 0.605±0.006        |
| (p-value vs SV-CCCD) | (0.0920)           | (0.5349)           | (0.0680)           | (0.0147)           | (0.0200)           | (0.2301)           | (0.0651)           | (NA)               | (0.0223)           | (NA)                  | (<0.0001)          |
| C5.0                 | 0.907±0.020        | 0.768±0.042        | 0.730±0.035        | 0.821±0.069        | 0.590±0.042        | 0.567±0.051        | 0.948±0.027        | <b>1.000±0.000</b> | 0.503±0.010        | 0.992±0.003           | 0.926±0.016        |
| (p-value vs SV-CCCD) | (0.7136)           | (0.5957)           | (0.0394)           | (0.0737)           | (0.5789)           | (0.2240)           | (0.1740)           | (0.3598)           | (<0.0001)          | (<0.0001)             | (0.1982)           |

Table 4.5: Mean and standard deviation of SV-CCCD and other chosen models’ AUCs across chosen real data. Values highlighted in red indicate the best performer of all models for a specific dataset. Values highlighted in blue indicate the best performer out of CCCD variant models. The results of the relevant 5x2 CV  $F$ -test are presented in parenthesis below. A p-value of “NA” indicates the same performance across all folds.



|                      | Ionosphere  | Sonar       | Yeast6      | Yeast5      | Yeast4      | Yeast1289vs7 | Vowel       | Shuttle     | Abalone     | Segment     | Pageblocks  |
|----------------------|-------------|-------------|-------------|-------------|-------------|--------------|-------------|-------------|-------------|-------------|-------------|
| SV-CCCD              | 0.939±0.019 | 0.652±0.091 | 0.393±0.102 | 0.522±0.109 | 0.258±0.073 | 0.274±0.154  | 0.974±0.037 | 0.982±0.012 | 0.993±0.001 | 0.304±0.031 | 0.828±0.017 |
| RW-CCCD              | 0.924±0.016 | 0.726±0.055 | 0.45±0.065  | 0.574±0.088 | 0.334±0.045 | 0.177±0.048  | 0.628±0.067 | 0.834±0.308 | 0.996±0.000 | 0.706±0.035 | 0.676±0.199 |
| (p-value vs SV-CCCD) | (0.1959)    | (0.5467)    | (0.6495)    | (0.7324)    | (0.5522)    | (0.1254)     | (0.0014)    | (0.5052)    | (0.0053)    | (<0.0001)   | (0.0008)    |
| P-CCCD               | 0.947±0.017 | 0.777±0.057 | 0.556±0.089 | 0.600±0.132 | 0.250±0.101 | 0.165±0.073  | 0.924±0.059 | 0.978±0.020 | 0.994±0.001 | 0.927±0.012 | 0.786±0.011 |
| (p-value vs SV-CCCD) | (0.3448)    | (0.1586)    | (0.1600)    | (0.4433)    | (0.7973)    | (0.0465)     | (0.3780)    | (0.8010)    | (0.3089)    | (<0.0001)   | (0.1128)    |
| MP-CCCD              | 0.960±0.017 | 0.810±0.048 | 0.582±0.076 | 0.693±0.076 | 0.358±0.102 | 0.303±0.102  | 0.956±0.016 | 0.983±0.016 | 0.996±0.000 | 0.931±0.013 | 0.788±0.010 |
| (p-value vs SVCCCD)  | (0.0458)    | (0.0722)    | (0.1268)    | (0.0566)    | (0.4912)    | (0.2947)     | (0.0282)    | (0.7651)    | (0.0045)    | (<0.001)    | (0.1135)    |
| Sk-NN                | 0.905±0.012 | 0.708±0.058 | 0.441±0.038 | 0.682±0.052 | 0.386±0.028 | 0.242±0.082  | 0.902±0.049 | 0.980±0.013 | 0.991±0.003 | 0.934±0.010 | 0.791±0.014 |
| (p-value vs SV-CCCD) | (0.1815)    | (0.6497)    | (0.7021)    | (0.3788)    | (0.2501)    | (0.3245)     | (0.0685)    | (0.8262)    | (0.4636)    | (<0.0001)   | (0.0605)    |
| SSVM                 | 0.96±0.006  | 0.448±0.141 | 0.590±0.037 | 0.754±0.039 | 0.397±0.041 | 0.345±0.103  | 0.949±0.032 | 0.976±0.011 | 0.993±0.002 | 0.974±0.012 | 0.719±0.056 |
| (p-value vs SV-CCCD) | (0.1609)    | (0.4361)    | (0.1297)    | (0.1292)    | (0.2410)    | (0.2283)     | (0.2694)    | (0.6759)    | (0.4145)    | (<0.0001)   | (0.1354)    |
| SC4.5                | 0.913±0.016 | 0.663±0.059 | 0.484±0.104 | 0.658±0.068 | 0.309±0.095 | 0.218±0.076  | 0.836±0.045 | 0.969±0.033 | 0.993±0.002 | 0.966±0.012 | 0.835±0.017 |
| (p-value vs SV-CCCD) | (0.1047)    | (0.7348)    | (0.6330)    | (0.0463)    | (0.4312)    | (0.0112)     | (0.0043)    | (0.7061)    | (0.5751)    | (<0.0001)   | (0.0352)    |
| Ek-NN                | 0.919±0.011 | 0.799±0.031 | 0.282±0.023 | 0.461±0.027 | 0.319±0.036 | 0.204±0.043  | 0.741±0.037 | 0.996±0.004 | 0.818±0.05  | 0.806±0.019 | 0.719±0.012 |
| (p-value vs SV-CCCD) | (0.2881)    | (0.1777)    | (0.3501)    | (0.6369)    | (0.6712)    | (0.2016)     | (0.0007)    | (0.2432)    | (0.0027)    | (<0.0001)   | (0.0085)    |
| ESVM                 | 0.890±0.017 | 0.643±0.040 | 0.181±0.026 | 0.391±0.030 | 0.268±0.034 | 0.087±0.023  | 0.590±0.046 | 0.968±0.010 | 0.711±0.128 | 0.630±0.025 | 0.686±0.020 |
| (p-value vs SV-CCCD) | (0.0798)    | (0.7883)    | (0.1194)    | (0.3492)    | (0.8173)    | (0.0312)     | (<0.0001)   | (0.5377)    | (0.0530)    | (<0.0001)   | (0.0079)    |
| EC4.5                | 0.931±0.015 | 0.783±0.044 | 0.247±0.030 | 0.431±0.043 | 0.238±0.029 | 0.095±0.021  | 0.789±0.052 | 0.989±0.012 | 0.793±0.036 | 0.966±0.012 | 0.798±0.020 |
| (p-value vs SV-CCCD) | (0.3457)    | (0.2317)    | (0.2505)    | (0.4196)    | (0.7637)    | (0.0531)     | (0.0102)    | (0.6503)    | (0.0020)    | (<0.0001)   | (0.5229)    |
| Ck-NN                | 0.929±0.010 | 0.810±0.026 | 0.470±0.052 | 0.568±0.046 | 0.456±0.048 | 0.281±0.052  | 0.963±0.054 | 0.988±0.007 | 0.996±0.000 | 0.976±0.006 | 0.798±0.012 |
| (p-value vs SV-CCCD) | (0.4462)    | (0.1127)    | (0.6021)    | (0.7370)    | (0.0641)    | (0.4575)     | (0.7393)    | (0.6836)    | (0.0053)    | (<0.0001)   | (0.2523)    |
| CSVM                 | 0.963±0.008 | 0.650±0.100 | 0.525±0.042 | 0.743±0.028 | 0.369±0.074 | 0.205±0.041  | 0.978±0.033 | 0.983±0.012 | 0.994±0.000 | 0.304±0.031 | 0.815±0.009 |
| (p-value vs SV-CCCD) | (0.0721)    | (0.5349)    | (0.4279)    | (0.1175)    | (0.4733)    | (0.2949)     | (0.4014)    | (0.5349)    | (0.1402)    | (<0.0001)   | (0.5948)    |
| C5.0                 | 0.944±0.013 | 0.744±0.041 | 0.519±0.054 | 0.683±0.109 | 0.243±0.083 | 0.195±0.128  | 0.916±0.048 | 1.000±0.000 | 0.996±0.001 | 0.987±0.004 | 0.876±0.015 |
| (p-value vs SV-CCCD) | (0.7197)    | (0.4164)    | (0.1814)    | (0.1730)    | (0.6060)    | (0.2641)     | (0.0843)    | (0.2254)    | (0.0053)    | (<0.0001)   | (0.0417)    |

Table 4.6: Mean and standard deviation of SV-CCCD and other chosen models'  $F_1$ -scores across chosen real data. Values highlighted in red indicate the best performer of all models for a specific dataset. Values highlighted in blue indicate the best performer out of CCCD variant models. The results of the relevant 5x2 CV  $F$ -test are presented in parenthesis when relevant.

|                      | Ionosphere         | Sonar              | Yeast6             | Yeast5             | Yeast4             | Yeast1289vs7       | Vowel              | Shuttle            | Abalone               | Segment               | Pageblocks         |
|----------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|-----------------------|-----------------------|--------------------|
| SV-CCCD              | 0.904±0.031        | 0.697±0.073        | 0.540±0.097        | 0.647±0.104        | 0.425±0.075        | 0.398±0.175        | <b>0.983±0.034</b> | 0.984±0.012        | 0.195±0.14            | 0.423±0.025           | <b>0.903±0.017</b> |
| RW-CCCD              | 0.904±0.021        | 0.724±0.057        | 0.393±0.165        | <b>0.845±0.098</b> | <b>0.752±0.070</b> | <b>0.599±0.117</b> | 0.866±0.055        | 0.938±0.131        | <b>0.609±0.059</b>    | 0.908±0.025           | 0.677±0.200        |
| (p-value vs SV-CCCD) | (0.44716)          | (0.5540)           | (0.0775)           | ( <b>0.2452</b> )  | ( <b>0.0260</b> )  | ( <b>0.0136</b> )  | (0.0638)           | (0.5369)           | ( <b>0.1191</b> )     | (<0.0001)             | (0.0029)           |
| P-CCCD               | 0.925±0.028        | 0.793±0.053        | 0.715±0.078        | 0.758±0.129        | 0.450±0.116        | 0.403±0.093        | 0.964±0.032        | 0.988±0.016        | 0.272±0.229           | 0.956±0.011           | 0.862±0.011        |
| (p-value vs SV-CCCD) | (0.1319)           | (0.1482)           | (0.1620)           | (0.1433)           | (0.7454)           | (0.3913)           | (0.0892)           | (0.6520)           | (0.4640)              | (<0.0001)             | (0.8104)           |
| MP-CCCD              | <b>0.945±0.025</b> | <b>0.824±0.039</b> | <b>0.730±0.055</b> | 0.697±0.096        | 0.573±0.058        | 0.526±0.110        | 0.968±0.036        | <b>0.991±0.013</b> | 0.135±0.146           | <b>0.963±0.008</b>    | 0.863±0.011        |
| (p-value vs SV-CCCD) | ( <b>0.0528</b> )  | ( <b>0.0742</b> )  | ( <b>0.0730</b> )  | (0.3946)           | (0.1157)           | (0.0344)           | (0.2965)           | ( <b>0.4901</b> )  | (0.1830)              | ( <b>&lt;0.0001</b> ) | (0.1088)           |
| Sk-NN                | 0.819±0.019        | 0.712±0.073        | 0.875±0.046        | <b>0.970±0.016</b> | 0.781±0.06         | 0.545±0.113        | 0.939±0.047        | 0.999±0.001        | 0.392±0.081           | 0.977±0.007           | 0.906±0.014        |
| (p-value vs SV-CCCD) | (0.0512)           | (0.6371)           | (0.0328)           | ( <b>0.0141</b> )  | (0.0004)           | (0.1985)           | (0.2986)           | (0.4282)           | (0.0154)              | (<0.0001)             | (0.4771)           |
| SSVM                 | 0.947±0.012        | 0.527±0.081        | 0.86±0.045         | 0.948±0.026        | 0.77±0.051         | 0.646±0.057        | <b>0.986±0.017</b> | 0.998±0.001        | 0.204±0.147           | 0.987±0.004           | 0.905±0.01         |
| (p-value vs SV-CCCD) | (0.1623)           | (0.2085)           | (0.0439)           | (0.0220)           | (0.0006)           | (0.0031)           | ( <b>0.5726</b> )  | (0.4255)           | (0.4017)              | (<0.0001)             | (0.4634)           |
| SC4.5                | 0.841±0.021        | 0.660±0.052        | 0.748±0.057        | 0.836±0.091        | 0.579±0.140        | 0.436±0.082        | 0.945±0.029        | 0.997±0.003        | 0.075±0.120           | 0.982±0.004           | 0.922±0.013        |
| (p-value vs SV-CCCD) | (0.2200)           | (0.7545)           | (0.1770)           | (0.0089)           | (0.0027)           | (0.0511)           | (0.2877)           | (0.5270)           | (0.5347)              | (<0.0001)             | (0.3662)           |
| Ek-NN                | 0.847±0.024        | 0.817±0.025        | <b>0.888±0.041</b> | 0.964±0.005        | <b>0.858±0.024</b> | <b>0.730±0.037</b> | 0.962±0.009        | 0.996±0.004        | 0.710±0.023           | 0.957±0.004           | 0.915±0.003        |
| (p-value vs SV-CCCD) | (0.1645)           | (0.1368)           | ( <b>0.0165</b> )  | (0.0208)           | ( <b>0.0015</b> )  | ( <b>0.0031</b> )  | (0.3969)           | (0.3793)           | (0.0001)              | (<0.0001)             | (0.3653)           |
| ESVM                 | 0.791±0.033        | 0.637±0.038        | 0.846±0.031        | 0.949±0.005        | 0.826±0.038        | 0.616±0.075        | 0.865±0.041        | 0.994±0.004        | 0.477±0.227           | 0.891±0.02            | 0.875±0.018        |
| (p-value vs SV-CCCD) | (0.0522)           | (0.5754)           | (0.0231)           | (0.0241)           | (0.0028)           | (0.0144)           | (0.0075)           | (0.4941)           | (0.0206)              | (<0.0001)             | (0.7261)           |
| EC4.5                | 0.906±0.016        | 0.791±0.043        | 0.86±0.039         | 0.953±0.011        | 0.82±0.033         | 0.616±0.078        | 0.963±0.011        | 0.999±0.001        | <b>0.718±0.039</b>    | 0.993±0.002           | <b>0.957±0.006</b> |
| (p-value vs SV-CCCD) | (0.6413)           | (0.3487)           | (0.0265)           | (0.0214)           | (0.0070)           | (0.0466)           | (0.4707)           | (0.3790)           | ( <b>&lt;0.0001</b> ) | (<0.0001)             | ( <b>0.0992</b> )  |
| Ck-NN                | 0.866±0.021        | 0.823±0.025        | 0.872±0.050        | <b>0.970±0.010</b> | 0.813±0.031        | 0.648±0.071        | 0.98±0.034         | 0.999±0.000        | 0.428±0.077           | <b>0.994±0.001</b>    | 0.891±0.014        |
| (p-value vs SV-CCCD) | (0.1626)           | (0.0953)           | (0.0369)           | ( <b>0.0159</b> )  | (0.0006)           | (0.0092)           | (0.5152)           | (0.3900)           | (0.0048)              | ( <b>&lt;0.0001</b> ) | (0.6459)           |
| CSVM                 | <b>0.950±0.010</b> | 0.696±0.081        | 0.825±0.049        | 0.947±0.026        | 0.75±0.056         | 0.667±0.078        | 0.984±0.032        | 0.984±0.012        | 0.656±0.079           | 0.423±0.025           | 0.952±0.006        |
| (p-value vs SV-CCCD) | ( <b>0.0978</b> )  | (0.5349)           | (0.0844)           | (0.0290)           | (0.0138)           | (0.0464)           | (0.0840)           | NA                 | (0.0004)              | NA                    | (0.1138)           |
| C5.0                 | 0.904±0.022        | 0.764±0.043        | 0.680±0.050        | 0.798±0.095        | 0.426±0.098        | 0.323±0.195        | 0.946±0.029        | <b>1.000±0.000</b> | 0.025±0.079           | 0.992±0.003           | 0.924±0.017        |
| (p-value vs SV-CCCD) | (0.6992)           | (0.4834)           | (0.0786)           | (0.0915)           | (0.5122)           | (0.4934)           | (0.1814)           | ( <b>0.3619</b> )  | (<0.0001)             | (<0.0001)             | (0.2619)           |

Table 4.7: Mean and standard deviation of SV-CCCD and other chosen models' G-mean scores across chosen real data. Values highlighted in red indicate the best performer of all models for a specific dataset. Values highlighted in blue indicate the best performer out of CCCD variant models. The results of the relevant 5x2 CV  $F$ -test are presented in parenthesis when relevant.

| AUC                            | Ionosphere           | Yeast6               | Yeast4               | Yeast1289vs7         | Abalone       | Segment              |
|--------------------------------|----------------------|----------------------|----------------------|----------------------|---------------|----------------------|
| SV-CCCD                        | 0.906 + 0.030        | 0.648 + 0.048        | 0.589 + 0.033        | 0.591 + 0.058        | 0.506 + 0.015 | 0.525 + 0.220        |
| SV-CCCD with Modified $\delta$ | <b>0.952 + 0.011</b> | 0.755 + 0.055        | 0.634 + 0.053        | 0.604 + 0.053        | 0.525 + 0.220 | 0.589 + 0.011        |
| SMOTE+ENN SV-CCCD              | 0.950 + 0.013        | 0.817 + 0.025        | <b>0.783 + 0.024</b> | <b>0.672 + 0.045</b> | 0.529 + 0.030 | <b>0.991 + 0.005</b> |
| <hr/>                          |                      |                      |                      |                      |               |                      |
| $F_1$ - Score                  |                      |                      |                      |                      |               |                      |
| SV-CCCD                        | 0.939 + 0.019        | 0.522 + 0.109        | 0.258 + 0.073        | 0.274 + 0.154        | 0.993 + 0.001 | 0.304 + 0.031        |
| SV-CCCD with Modified $\delta$ | <b>0.963 + 0.010</b> | 0.540 + 0.054        | 0.146 + 0.190        | 0.263 + 0.070        | 0.994 + 0.001 | 0.937 + 0.008        |
| SMOTE+ENN SV-CCCD              | 0.960 + 0.008        | 0.551 + 0.041        | 0.350 + 0.042        | 0.232 + 0.059        | 0.993 + 0.001 | 0.757 + 0.022        |
| <hr/>                          |                      |                      |                      |                      |               |                      |
| G-mean                         |                      |                      |                      |                      |               |                      |
| SV-CCCD                        | 0.904 + 0.031        | 0.540 + 0.097        | 0.425 + 0.075        | 0.398 + 0.175        | 0.195 + 0.140 | 0.423 + 0.250        |
| SV-CCCD with Modified $\delta$ | 0.937 + 0.008        | 0.540 + 0.263        | 0.425 + 0.075        | <b>0.678 + 0.215</b> | 0.196 + 0.141 | 0.423 + 0.250        |
| SMOTE+ENN SV-CCCD              | 0.757 + 0.019        | <b>0.811 + 0.036</b> | <b>0.756 + 0.040</b> | 0.641 + 0.070        | 0.212 + 0.158 | <b>0.992 + 0.004</b> |

Table 4.8: Comparison of original SV-CCCD, modified  $\delta^*$  SV-CCCD, and SMOTE+ENN preprocessed SV-CCCD average and standard deviation for selected real datasets. Values in blue would be the highest performing CCCD variant. Values in red would be the highest performing of all models.

#### 4.4 Conclusions and Discussions

This chapter introduces a hybrid classification method, called SV-CCCD, based on identifying regions of soft- and hard-overlap using SVM decision boundaries. Points in the soft-overlap region are classified using the SVM decision boundary while points in the hard-overlap region are classified using RW-CCCD.

Simulated results show that SV-CCCD performs as well as other state-of-the-art classifiers in classification of imbalanced and class overlapping datasets. Real data studies showed that SV-CCCD is aggressive in classifying minority class observations, often at the detriment to majority class observations. The runtime for SV-CCCD makes it a potentially appealing alternative to certain other strong classification methods, especially in settings where having high precision is more important than accuracy or recall.

We close the chapter by noting that both the construction the original separating hyperplane and  $\delta$  greatly affect classification performance of SV-CCCD. This leads to two potential areas of future work,

- (i) finding an optimal selection for both  $\delta$  and a way to construct  $P$ , or

(ii) constructing a hybrid classifier that is not dependent on either  $\delta$  or  $P$ .

For (ii), consider the convex hull (or SVDD descriptions) of each class. A similar hybrid classifier can be built such that observations within the intersection of the class hulls are considered the region of hard-overlap and the rest are soft-overlap. In this construction, different CCCD classifier families could be used for soft- and hard-overlapping regions respectively. This construction is not dependent on  $\delta$  to identify these regions, nor does it depend on a prior processing with SVM. In fact, after finding the convex hulls and identifying points in the overlap, both classification of soft- and hard-overlapping regions using CCCD families can be done in parallel.

## Chapter 5

### Adaptive Ensemble Classification with CCCDs: The ACDC Algorithm

#### 5.1 Statistical Complexity

In this chapter, we explore combining different ensemble methods with CCCD classifiers. CCCD classifiers are inherently *statistically complex* by construction, where statistical complexity refers to a model’s capacity to approximate a wide range of functions or data-generating processes. This complexity directly impacts the bias–variance tradeoff and generalization error [113]. P-CCCDs are moderate- to low-bias classifiers, depending on the granularity of the covering. Because the classifier encodes a detailed geometric structure, especially when many small covering balls are used, it is capable of approximating highly intricate decision boundaries. P-CCCD classifiers are high-variance models as small perturbations in the training data can lead to significant changes in the cover and, consequently, prediction. Moreover, the greedy algorithm used to select prototype points does not guarantee convergence to a consistent cover, as the resulting structure depends heavily on the digraph’s configuration.

Currently, no single unifying measure of statistical complexity exists. In this section, we briefly introduce two commonly used measures, the *Vapnik–Chervonenkis (VC) dimension* and *empirical Rademacher complexity*. Both VC-dimension and empirical Rademacher complexity provide theoretical approaches to measure classifier complexity, and each offers insights into how the difference between training and test error (i.e., the *generalization bound*) can be controlled [74]. The VC-dimension defines the largest number of points that a model can *shatter* in the *hypothesis space* [131]. The hypothesis space,  $\mathcal{F}$ , refers to the set of functions or models that a learning algorithm can choose from to map inputs to binary outputs (i.e., binary class labels). A set of points is shattered if, for every possible labeling

of points, there exists a classifier that produces that particular labeling from the hypothesis space [135].

VC-dimension measures worst-case rather than average-case complexity and can provide looser generalization bounds than other measures [135]. VC-dimension does not depend on how the cardinality of the hypothesis space. In fact,  $\mathcal{F}$  may be infinite in cardinality, but its VC-dimension can still be finite due to geometric restrictions. Examples of such are linear classifiers, which have an infinite cardinality hypothesis class with VC-dimension of  $d + 1$  [34].

Data-dependent classifiers like P-CCCD do not define a fixed function class before seeing the data. Instead, the classifier is constructed from the data as covering ball centers rely on target class points and radii are distances to the nearest non-target class point. Thus, we are not choosing from a static class  $\mathcal{F}$  but generating a data-dependent function  $f_S$  from the sample  $S$ . However, the VC-dimension can still be finite under an appropriate interpretation. To define VC-dimension for P-CCCD, we reinterpret  $\mathcal{F}$  as the set of all classifiers that can be constructed from *any finite sample* in  $\mathbb{R}^d$

$$\mathcal{F} = \bigcup_{n=1}^{\infty} \{f_S : S \subset \mathbb{R}^d, |S| = n, f_S \text{ is a P-CCCD trained on } S\}. \quad (5.1)$$

P-CCCD classifiers are structured enough to admit only finitely many dichotomies on finite point sets. We show theoretical results that the class of all P-CCCD classifiers built over finite samples in  $\mathbb{R}^d$ , with radius defined by nearest cross-class distance and classification based on MDS of the induced PCD, has finite VC-dimension due to its restricted combinatorial and geometric complexity.

**Theorem 5.1** (Finite VC-Dimension of P-CCCD Classifiers). Let  $\mathcal{F}$  denote the class of classifiers defined by the P-CCCD construction in  $\mathbb{R}^d$ . Each classifier  $f \in \mathcal{F}$  is induced from a finite labeled dataset  $S = S_0 \cup S_1 \subset \mathbb{R}^d$ , with  $S_0$  labeled class 0 and  $S_1$  labeled class 1, as follows:

1. For each  $x \in S_0$ , construct an open  $L_p$ -ball  $B_p(x, r_x)$  with radius  $r_x = \min_{y \in S_1} \|x - y\|_p$ .
2. Form a proximity digraph on  $S_0$  with edges from  $x$  to  $x'$  if  $x' \in B_p(x, r_x)$ .
3. Extract an approximate minimum dominating set (MDS)  $D \subseteq S_0$  from this digraph.
4. Classify a query point  $z$  as class 0 if  $z \in \bigcup_{x \in D} B_p(x, r_x)$ ; otherwise, assign class 1.

Then the VC-dimension of  $\mathcal{F}$  satisfies:

$$d + 1 \leq \text{VC}(\mathcal{F}) \leq \mathcal{O}(nd \log n).$$

*Proof.* We prove the result in two parts.

**(1) Lower Bound:**  $\text{VC}(\mathcal{F}) \geq d + 1$ .

Let  $S = \{x_1, \dots, x_{d+1}\} \subset \mathbb{R}^d$  be the vertices of a regular  $d$ -simplex with edge length  $\delta > 0$ . For each binary labeling  $\{y_1, \dots, y_{d+1}\} \in \{0, 1\}^{d+1}$ , define:

$$S_0 = \{x_i \mid y_i = 0\}, \quad S_1 = \{x_i \mid y_i = 1\}.$$

Since all pairwise distances are  $\delta$ , each ball  $B_p(x, \delta)$  for  $x \in S_0$  excludes all other points in  $S$ , thus guaranteeing class-pure, disjoint covers.

For each labeling, P-CCCD correctly classifies all points, as MDS selects every  $x \in S_0$  (no domination is possible otherwise), and no ball covers any point in  $S_1$ . Therefore, all  $2^{d+1}$  labelings are realized, and thus:

$$\text{VC}(\mathcal{F}) \geq d + 1.$$

**(2) Upper Bound:**  $\text{VC}(\mathcal{F}) \leq \mathcal{O}(nd \log n)$ .

Each classifier in  $\mathcal{F}$  is defined by a union of  $L_p$  balls centered at a subset of training points, where:

- Each radius is constrained by the nearest cross-class neighbor.

- Centers are restricted to training points and selected via a minimal dominating set.

Let  $\mathcal{B}_n$  denote the family of unions of up to  $n$  open  $L_p$ -balls in  $\mathbb{R}^d$  with arbitrary centers and radii. From known results (see Devroye et al., 1996, Chapter 13), such a class has VC-dimension bounded by:

$$\text{VC}(\mathcal{B}_n) = \mathcal{O}(nd \log n).$$

The notation  $\text{VC}(\mathcal{B}_n) = \mathcal{O}(nd \log n)$  indicates an asymptotic upper bound on the VC-dimension for a fixed number of balls  $n$  in  $\mathbb{R}^d$ . Since both  $n$  and  $d$  are finite in practice, it follows that  $\text{VC}(\mathcal{B}_n)$  is finite for any fixed  $n$  and  $d$ . More specifically, for each fixed  $n$  and  $d$ , the VC-dimension is bounded above by some constant  $C \cdot nd \log n$ , where  $C > 0$  is universal. P-CCCD imposes *additional structure*:

- Radii are not free parameters—they are data-dependent.
- MDS constraints limit the number of selected centers.

Hence, P-CCCD classifiers form a subclass of  $\mathcal{B}_n$ , with stricter geometric and combinatorial constraints. Since subclasses cannot increase VC-dimension, we conclude:

$$\text{VC}(\mathcal{F}) \leq \text{VC}(\mathcal{B}_n) = \mathcal{O}(nd \log n).$$

This completes the proof. □

*Empirical Rademacher complexity* provides a theoretically principled, computationally tractable framework for justifying a simpler models in the selection of a base learner for AdaBoost when traditional VC-dimension analysis becomes intractable [9]. Empirical Rademacher complexity quantifies the ability of a function class to fit random noise and provides a data-dependent measure of capacity. Unlike VC-dimension, it reflects average-case complexity. The empirical Rademacher complexity of a hypothesis class  $\mathcal{F}$  with respect to



a sample  $S = \{x_1, x_2, \dots, x_n\}$  is defined as

$$\hat{\mathfrak{R}}_S(\mathcal{F}) = \mathbb{E}_{\boldsymbol{\sigma}} \left[ \sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n \sigma_i f(x_i) \right] \quad (5.2)$$

where  $\sigma_i \in \{-1, 1\}$  are called *Rademacher variables* and are uniformly distributed over  $\{-1, 1\}$ . The expression  $\mathbb{E}_{\boldsymbol{\sigma}}[\cdot]$  denotes the expectation over the distribution of the Rademacher variables.

The *Rademacher complexity* is the expectation of empirical Rademacher complexity over all samples of size  $n$  drawn from a distribution  $\mathcal{D}$ ,

$$\mathfrak{R}_n(\mathcal{F}) = \mathbb{E}_{S \sim \mathcal{D}^n} [\hat{\mathfrak{R}}_S(\mathcal{F})]. \quad (5.3)$$

While VC-dimension is distribution independent, the Rademacher complexity is distribution dependent. Let  $\mathcal{F}$  be a hypothesis class of functions mapping from  $\mathcal{X}$  to  $[0, 1]$ , and let  $\ell(f(x), y)$  be a chosen loss function bounded in the space  $[0, 1]$ . Then, with probability at least  $1 - \delta$  over the choice of sample  $S = \{(x_1, y_1), \dots, (x_n, y_n)\}$ , we can bound the generalization error using the empirical Rademacher complexity [84]. The following generalization bound holds for all  $f \in \mathcal{F}$

$$\mathbb{E}_{(x,y) \sim \mathcal{D}} [\ell(\hat{f}(x), y)] \leq \frac{1}{n} \sum_{i=1}^n \ell(\hat{f}(x_i), y_i) + 2\hat{\mathfrak{R}}_S(\mathcal{F}) + 3\sqrt{\frac{\log_2(2/\delta)}{2n}}. \quad (5.4)$$

We can also bound the empirical Rademacher complexity using VC-dimension as a worse case [84, 9]. Let  $\mathcal{F}$  be a class of binary-valued functions with VC-dimension  $VC(\mathcal{F})$ , and let  $S = \{x_1, \dots, x_n\}$  be a sample of size  $n$ . Then, the empirical Rademacher complexity satisfies the following bound

$$\hat{\mathfrak{R}}_S(\mathcal{F}) \leq C \sqrt{\frac{VC(\mathcal{F}) \log_2 \left( \frac{n}{VC(\mathcal{F})} \right)}{n}} \quad (5.5)$$

for some universal constant  $C > 0$ , and assuming  $VC(\mathcal{F}) < n$ .

The rest of the chapter is organized by providing a novel CCCD framework and theoretical justifications on the VC-dimension and computational complexity for classification under the new framework. We also show comparison of empirical Rademacher complexities of this new framework and P-CCCDs as justification for using the new framework as the weak learner in a boosting scheme. We detail the new boosted CCCD classifier and provide results on both synthetic and real datasets, ending by summarizing the effectiveness of the new classifier on imbalanced and overlapping data compared to other strong model variants.

## 5.2 FlexiBall Classifiers

In this section, we propose a new family of class cover catch digraph called “FlexiBalls”. A *FlexiBall of size  $k$*  selects  $k$  observations from the target class and constructs metric covering balls around them to form the cover. That is, for target class cover we select  $x_i \in \mathcal{X}_j$  for  $i = 1, \dots, k$ . Given  $\tau \in (0, 1]$ , each selected  $x_i$  constructs a metric covering ball  $B_i(x_i, r(x_i))$  such that radius  $r(x_i)$  is determined by

$$r(x_i) = \tau d(x_i, u(x'_i)), \quad (5.6)$$

where

$$u(x'_i) = \arg \min_{x'_i \in \mathcal{X}_{1-j}} d(x_i, x'_i). \quad (5.7)$$

Instead of finding an MDS as with P-CCCD, the union of the  $k$ -covering balls act as the cover approximation,  $\mathcal{C}_{\mathcal{X}_j} = \bigcup_{i=1}^k B_i(x_i, r(x_i))$ . Depending on the cardinality of  $k$ , the resultant cover is no longer guaranteed to be proper, but is still pure. Classification with FlexiBalls is done using Equation (2.31). Equation (5.6) acts similarly to Equation (2.27), however, we allow  $\tau$  to (potentially) change the structure of the digraph.

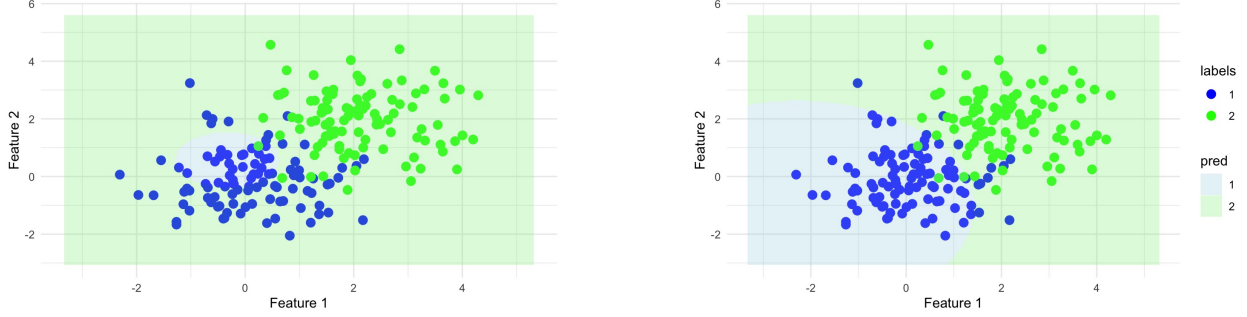


Figure 5.1: Example of how the classification decision rule for SBCs change depending on which point is selected.

We denote the special case of FlexiBall of size  $k = 1$  classifiers as “single-ball” classifiers (SBC). Figure (5.1) shows how the classification boundary for SBCs change depending on the prototype being selected. We present the following result about the hypothesis space of SBC classifiers.

**Lemma 5.2.1** (VC-Dimension of Single-Ball FlexiBall Classifiers). Let  $\mathcal{F}_1$  denote the hypothesis class of classifiers in  $\mathbb{R}^d$  defined as follows: each classifier  $f \in \mathcal{F}_1$  is constructed from a finite labeled dataset  $S = S_0 \cup S_1$ , where  $S_0$  (class 0) and  $S_1$  (class 1) are disjoint subsets. The classifier assigns class 0 to a query point  $z \in \mathbb{R}^d$  if it lies within a single open  $L_p$ -ball centered at a point  $x \in S_0$ , with radius  $r_x = \tau \min_{y \in S_1} \|x - y\|_p$  for some fixed  $\tau \in (0, 1]$ ; otherwise,  $z$  is assigned to class 1. Then,

$$\text{VC}(\mathcal{F}_1) = 2.$$

*Proof.* We prove this in two parts.

**(i) Lower bound:**  $\text{VC}(\mathcal{F}_1) \geq 2$ .

Let  $S = \{x_1, x_2\} \subset \mathbb{R}^d$  be two distinct points. For each of the  $2^2 = 4$  binary labelings of these points. If both are labeled the same (00 or 11), the classification is trivial. If one is 0 and the other 1, place the ball at the class-0 point. Since  $\tau \in (0, 1]$ , the ball can be

made small enough to exclude the other point. Hence, all 4 dichotomies are realizable, so  $\text{VC}(\mathcal{F}_1) \geq 2$ .

**(ii) Upper bound:**  $\text{VC}(\mathcal{F}_1) < 3$ .

Suppose, for contradiction, that 3 points  $x_1, x_2, x_3 \in \mathbb{R}^d$  can be shattered. Consider the labeling  $y = (0, 1, 0)$ , i.e., class 0 for  $x_1$  and  $x_3$ , class 1 for  $x_2$ . To classify both  $x_1$  and  $x_3$  as class 0, the single ball must contain both—but also exclude  $x_2$ , which lies between them (in some configurations). This is geometrically impossible in general for open balls. Thus, some labelings on 3 points cannot be realized.

Therefore,  $\text{VC}(\mathcal{F}_1) \leq 2$ , and combining both parts:

$$\text{VC}(\mathcal{F}_1) = 2.$$

□

Generalizing for any  $k$ , we have the following theorem,

**Theorem 5.2** (VC-Dimension of  $k$ -Ball FlexiBall Classifiers). Let  $\mathcal{F}_k$  denote the hypothesis class of FlexiBall classifiers of size  $k$ . Each classifier  $f \in \mathcal{F}_k$  is determined by a finite labeled dataset  $S = S_0 \cup S_1 \subset \mathbb{R}^d$ , with  $S_0$  labeled as class 0 and  $S_1$  as class 1. Select one class as the target class. The classifier places open  $L_p$ -balls around  $k$  target class points  $\{x_i\}_{i=1}^k \subset S_j$  for either  $j = 0, 1$ , each with radius  $r_i = \tau \min_{y \in S_{1-j}} \|x_i - y\|_p$  for some  $\tau \in (0, 1]$ , and assigns the target class label to any query point lying in the union of the  $k$  balls, and the non-target class label otherwise. Then,

$$\text{VC}(\mathcal{F}_k) = 2k.$$

*Proof.* We show the result in two parts.

**(i) Lower Bound:**  $\text{VC}(\mathcal{F}_k) \geq 2k$ .

Let  $S = \{x_1, \dots, x_{2k}\} \subset \mathbb{R}^d$  be a set of  $2k$  points in general position (i.e., no three collinear, no degeneracies). There are a total of  $2^{2k}$  different class labels. Partition the

points into  $k$  disjoint pairs:

$$(x_1, x_2), (x_3, x_4), \dots, (x_{2k-1}, x_{2k}).$$

For each pair, assign one point to class 0 and the other to class 1. Construct a FlexiBall classifier that places a ball around each of the  $k$  class 0 points, with radius equal to the distance to its paired class 1 point. Since the pairs are disjoint and in general position, these  $k$  balls do not overlap and classify all  $2k$  points correctly.

We can repeat this for all  $2^k$  labelings where exactly one point in each pair is labeled class 0 and the other class 1. The other labelings will occur when classes are imbalanced. Without loss of generality, assume there are more class 1 than class 0 points. This means that there the cardinality for class 0 is less than  $k$ . But, we can define the class 0 points as the target points and construct a cover of up to  $k$  balls. Since there are less than  $k$  points, we use just up to that many of the total  $k$  covering balls available. In this way, we realize a family of labelings over  $2k$  points.

Therefore, there exists a configuration of  $2k$  points that can be shattered by  $\mathcal{F}_k$ , and hence:

$$\text{VC}(\mathcal{F}_k) \geq 2k.$$

**(ii) Upper Bound:**  $\text{VC}(\mathcal{F}_k) < 2k + 1$ .

Assume for contradiction that some set of  $2k+1$  points can be shattered by  $\mathcal{F}_k$ . Then for every possible labeling of this set, there exists a selection of  $k$  class 0 centers and associated balls that realize the labeling.

Consider the labeling where  $k+1$  points are labeled class 0 and the remaining  $k$  points are class 1. Since only  $k$  centers are allowed, at least one class 0 point—say  $x^*$ —cannot serve as a center. To classify  $x^*$  as class 0, it must be covered by the ball of another center. But for general configurations, especially if  $x^*$  is far from all other class 0 points, and radii are constrained by distances to nearby class 1 points, this cannot be guaranteed. Hence, there

exists a labeling that cannot be realized, and thus the class  $\mathcal{F}_k$  cannot shatter  $2k + 1$  points. Therefore,

$$\text{VC}(\mathcal{F}_k) \leq 2k.$$

Combining both bounds, we conclude:

$$\text{VC}(\mathcal{F}_k) = 2k.$$

□

FlexiBall classifiers of size  $n$  where  $n$  is the cardinality are a special case of P-CCCD classifiers when  $\tau = 1$ . FlexiBalls, however, allow for alternative digraph structures when  $\tau < 1$ . This flexibility can potentially form more adaptive covers than the more rigid P-CCCD formulation, which motivates the FlexiBall name. We conclude this section by presenting theoretical results on the computational complexity of the FlexiBall learning framework.

**Theorem 5.3** (Computational Complexity of FlexiBall Classifiers). Let  $\mathcal{X} = \mathcal{X}_j \cup \mathcal{X}_{1-j} \subset \mathbb{R}^d$  be the training data, where  $|\mathcal{X}_j| = n$  and  $|\mathcal{X}_{1-j}| = m$ . Then, for a fixed number  $k$  of target-class prototypes, the FlexiBall classifier of size  $k$  constructs a (potentially improper) pure class cover  $\mathcal{C}_{\mathcal{X}_j}$  in  $\mathcal{O}(kd(n + m))$  time.

*Proof.* The algorithm proceeds in the following steps:

- Select  $k$  prototypes from  $\mathcal{X}_j$  via a linear scan, which takes  $\mathcal{O}(n)$  time.
- For each prototype  $x'_i$ , compute its distances to all points in  $\mathcal{X}_j$  and  $\mathcal{X}_{1-j}$ . This requires  $\mathcal{O}(knd)$  and  $\mathcal{O}(kmd)$  operations, respectively, yielding a total of  $\mathcal{O}(kd(n + m))$  time.
- Compute the upper radius  $u(x'_i)$  for each prototype as in Equation (5.7), which involves aggregating the distance information and requires  $\mathcal{O}(k(n + m))$  time.

- Construct the arc set  $\mathcal{A}(D)$  based on cover relations between prototypes and in-class points, requiring  $\mathcal{O}(kn)$  time.

Combining all steps, the total time complexity is  $\mathcal{O}(kd(n + m))$ .  $\square$

### 5.3 AdaCover Digraph Classifier (ACDC)

The results of the previous section allows us to bound the empirical Rademacher complexity and in turn the generalization bound for the family of FlexiBall classifiers of size  $k$ . Plugging results from Theorem 5.2, the upper bound of the empirical Rademacher complexity of FlexiBalls of size  $k$  for random sample with labels  $y_i \in \{0, 1\}$ ,  $S = \{(x_1, y_1) \dots, (x_n, y_n)\}$  is

$$\hat{\mathfrak{R}}_S(\mathcal{F}_k) \leq C \sqrt{\frac{2k \log_2 \left( \frac{n}{2k} \right)}{n}}. \quad (5.8)$$

where  $k < n/2$  and  $C$  is an universal constant.

Given a chosen loss function  $\ell(\hat{f}(x), y)$ , we can bound the worst-case generalization error of FlexiBall classifiers to be

$$\mathbb{E}_{(x,y) \sim \mathcal{D}}[\ell(\hat{f}(x), y)] \leq \frac{1}{n} \sum_{i=1}^n \ell(\hat{f}(x_i), y_i) + 2C \sqrt{\frac{2k \log_2 \left( \frac{n}{2k} \right)}{n}} + 3 \sqrt{\frac{\log_2(2/\delta)}{2n}}. \quad (5.9)$$

Both  $\tau$  and  $k$  affect the bound as  $\hat{f}(x_i)$  contains the scaled dissimilarity measure which is affected by  $r(x_i)$  and hence  $\tau$ . The complexity grows logarithmically with the number of covering balls  $k$ , but decays with sample size  $n$ .

We investigate the empirical Rademacher complexity of both P-CCCD and FlexiBall classifiers of size  $k = 1$  and  $k = \sqrt{n}$ . We assume a discrete uniform weighting scheme for the stochastic selection of FlexiBalls. 1000 sample observations were drawn from a bivariate Gaussian distribution and randomly labeled either  $\{0, 1\}$  with equal probability. Figure (5.2) shows the effect over  $n$ . We evaluate the maximum empirical Rademacher complexity over

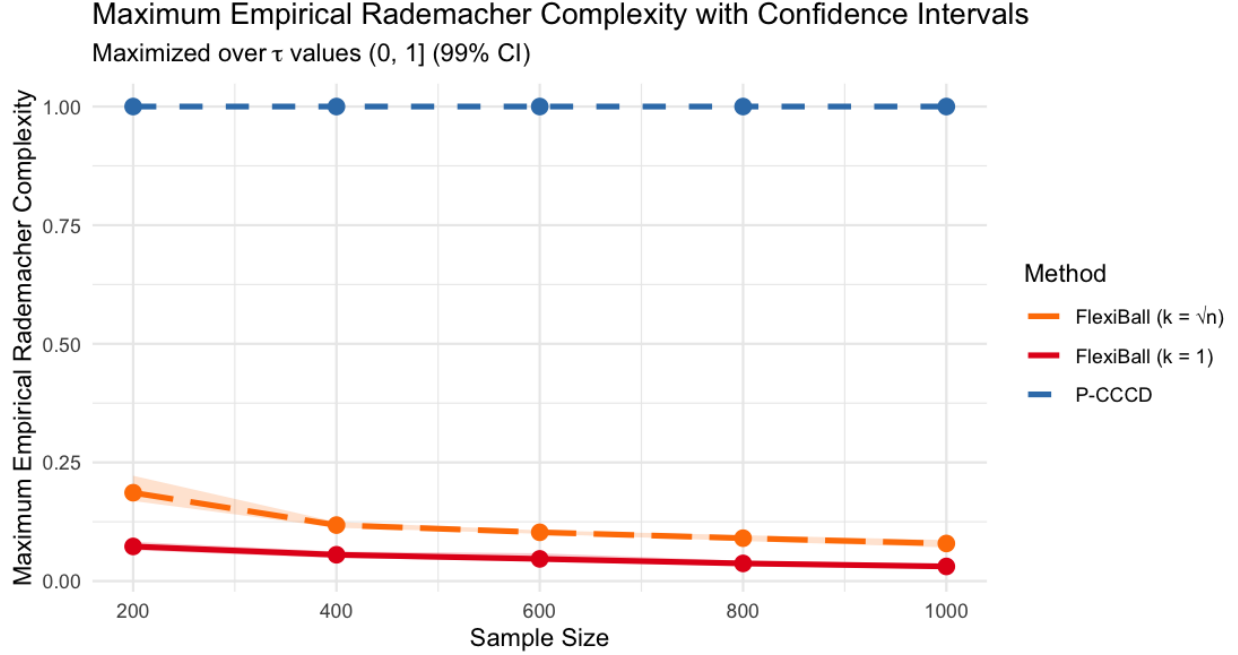


Figure 5.2: The maximum empirical Rademacher complexity with bootstrapped confidence intervals for P-CCCD, FlexiBalls of size  $\sqrt{n}$ , and SBC across different  $n$  values.

all possible values of  $\tau$  and bootstrap using 10 iterations to construct the 99% bootstrap confidence intervals for each empirical Rademacher complexity.

The empirical Rademacher complexity of both FlexiBall variants is substantially lower than that of the P-CCCD classifier. We see the tightening of the CI for the FlexiBall of class  $\sqrt{n}$  as the sample size increases, which suggests that larger  $n$  dominate the effect of  $\tau$  on the bounds.

Although this observation is based on a single distribution, it provides an initial justification for favoring FlexiBalls over P-CCCD as base learners in a boosting framework. When considering Rademacher complexity in the context of choosing a base learner for boosting, a lower Rademacher complexity is generally better, as the empirical Rademacher complexity is a measure of a model class's ability to fit random noise [74]. A higher value means the model class can fit a wider range of functions (even meaningless noise), which typically implies higher variance and greater risk of overfitting.



In the remainder of this section, we propose a boosted classification algorithm based on FlexiBall classifiers called the AdaCover Digraph Classifier (ACDC). Each observation, thus each node of the proximity digraph is weighted according to the probability distribution of the observations. While P-CCCD uses a deterministic selection for the greedy selected minimum dominating set, ACDC stochastically chooses  $k$  prototype covering balls using updated vertex weights at each iteration of boosting. Depending on domain knowledge, initialized weights can be set to any normalized probability vector with the default being uniform in the first iteration.

Note that  $k$  is user defined before training. Proving an exact bound on an optimal  $k$  selection for ACDC is left for future work. We adopt using  $\sqrt{N}$  where  $N$  is the cardinality of the training data as the upper bound as a practical rule-of-thumb. Specifically, we use the floor of  $\sqrt{N}$  when necessary. Similar heuristic approaches appear in some clustering algorithms and decision trees, but the decision to bound  $k$  in this way was done because of computational and generalization advantage [17, 62, 55]. VC-dimension of FlexiBall classifiers grows linearly with  $k$ , therefore choosing  $k$  on the order of  $\sqrt{N}$  means the base learner’s VC-dimension will grow sublinear in  $N$ . This is desirable for generalization as it keeps the model flexible but avoids overfitting even as  $N$  increases. Additionally, the base FlexiBall algorithm’s time complexity scaling with  $\sqrt{N}$  implies worst-case runtime for fitting for each iteration of FlexiBalls becomes  $O(N^{3/2}d)$ .

Classification is done using Equation (2.31). Misclassified points in previous iterations and their covering balls are more likely to be stochastically selected as prototypes in future iterations [113]. The final classification rule is aggregated using a weighted majority vote. Pseudo-code is detailed in Algorithm (5). ACDCs are generalized for multi-class classification problems by using the stagewise additive modeling using a multi-class exponential loss (SAMME) algorithm. More information on the SAMME algorithm for multi-class boosting is found in [51].

---

**Algorithm 5** AdaCover Digraph Classifier (ACDC)

---

**Input:** Query point  $z$ , training data  $\mathcal{X}$  with labels  $\mathcal{Y} \in \{-1, 1\}$ , number of rounds  $L$ , prototype size  $k$ , radius parameter  $\tau$

**Output:** Final classifier  $H(z) \in \mathcal{Y}$

Initialize weights:  $w_1(x_i) \leftarrow 1/N$  for all  $x_i \in \mathcal{X}$ ,  $N = |\mathcal{X}|$

**for**  $l = 1$  to  $L$  **do**

**for**  $j \in \{0, 1\}$  **do**

        Define:  $\mathcal{X}_j \leftarrow$  class- $j$  samples,  $\mathcal{X}_{1-j} \leftarrow$  non-target

        Normalize:  $w_{l,j} \leftarrow w_l|_{\mathcal{X}_j}$

        Sample  $k$  prototypes:  $\mathcal{X}_j^k \subset \mathcal{X}_j$  from  $w_{l,j}$

        Compute radii  $r(x_i)$  for  $x_i \in \mathcal{X}_j^k$  via (5.6)

        Construct cover  $\mathcal{C}_{\mathcal{X}_j} = \{C_1, \dots, C_k\}$

**end for**

    Weak classifier:  $f_l(x) \leftarrow$  prediction via (2.31)

    Error:  $\text{err}_l \leftarrow \sum_i w_l(x_i) \mathbb{I}[f_l(x_i) \neq f(x_i)]$

    Weight:  $\alpha_l \leftarrow \log \left( \frac{1 - \text{err}_l}{\text{err}_l} \right)$

    Update:

$$w_{l+1}(x_i) \leftarrow w_l(x_i) \cdot \exp(\alpha_l \cdot \mathbb{I}[f_l(x_i) \neq f(x_i)])$$

**end for**

Return:

$$H(z) \leftarrow \arg \max_{k \in \{-1, 1\}} \sum_{l=1}^L \alpha_l \cdot \mathbb{I}[f_l(z) = k]$$

---

## 5.4 Monte Carlo Simulations

In this section, we present similar experimental results for synthetic datasets used in Chapters 3 and 4 under different combinations of class overlap and imbalance settings for each of the three evaluation metrics. Table (5.1) provides a detailed description of each strong classifier used in our simulated and real data settings for this chapter. Simulations were performed in R using self-written code for P-CCCD, MP-CCCD, SV-CCCD and ACDC classification, `cccd` for RW-CCCD classification, `RWeka` for C4.5 classification, `DMwR2` for k-NN classification, `e1071` for SVM classification, and `c50` for C5.0 classification [82, 53, 130, 97, 103, 104].

| Method    | Description                                                                                                                                                                                                                |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| P-CCCD    | P-CCCD with the optimal $\tau^*$ (in the pilot study) among $\tau = 0.0, 0.1, \dots, 1.0$ .                                                                                                                                |
| RW-CCCD   | RW-CCCD with the optimal $e^*$ (in the pilot study) among $e = 0.0, 0.1, \dots, 1.0$ .                                                                                                                                     |
| ACDC- $k$ | ACDC with optimal $k^*$ (in the pilot study) among $k = 1, 2, \dots, \lfloor \sqrt{N} \rfloor$ balls considered and optimal $\tau^*$ (in the pilot study) among $\tau = 0.0, 0.1, \dots, 1.0$ with $L = 10$ .              |
| MP-CCCD   | MP-CCCD with the optimal $e^*$ (in the pilot study) among $e = 0.0, 0.1, \dots, 1.0$ and optimal $\tau^*$ (in the pilot study) among $\tau = 0.0, 0.1, \dots, 1.0$ .                                                       |
| SV-CCCD   | SV-CCCD with the optimal $e^*$ (in the pilot study) among $e = 0.0, 0.1, \dots, 1.0$ , optimal $\gamma^*$ among $\gamma = 0.1, \dots, 4.0$ , and $\delta^*$ as the distance between the largest misclassified test points. |
| $Sk$ -NN  | $k$ -NN using a combination of SMOTE ( $t = 2$ and $k = 5$ ) and ENN ( $K = 3$ ) with optimal $k^*$ (in pilot study) among $k = 1, 2, \dots, 30$ [10].                                                                     |
| SSVM      | SVM using a combination of SMOTE ( $t = 2$ and $k = 5$ ) and ENN ( $K = 3$ ) with optimal $\gamma^*$ (in pilot study) among $\gamma = 0.1, 0.2, \dots, 3.9, 4.0$ [58].                                                     |
| SC4.5     | C4.5 using a combination of SMOTE ( $t = 2$ and $k = 5$ ) and ENN ( $K = 3$ ) with Laplace smoothing and no pruning.                                                                                                       |
| $Ek$ -NN  | $k$ -NN using a combination of undersampling ( $t = 4$ ) and Adaboost ( $L = 10$ ) with optimal $k^*$ (in pilot study) among $k = 1, 2, \dots, 30$ [71].                                                                   |
| ESVM      | SVM using a combination of undersampling ( $t = 4$ ) and Adaboost ( $L = 10$ ) for support vector machines with optimal $\gamma^*$ (in pilot study) among $\gamma = 0.1, 0.2, \dots, 3.9, 4.0$ [71].                       |
| EC4.5     | C4.5 decision trees using a combination of undersampling ( $t = 4$ ) and Adaboost ( $L = 10$ ) with optimal $k^*$ (in pilot study) possible trees among $k = 1, 2, \dots, 30$ trees [62].                                  |
| $Ck$ -NN  | A cost-sensitive version of $k$ -NN with optimal $k^*$ (in pilot study) among $k = 1, 2, \dots, 30$ [95].                                                                                                                  |
| CSVM      | A cost-sensitive version of SVM with optimal $\gamma^*$ (in pilot study) among $\gamma = 0.1, 0.2, \dots, 3.9, 4.0$ [95].                                                                                                  |
| C5.0      | C5.0 decision tree with no global pruning [62].                                                                                                                                                                            |

Table 5.1: The description of other CCCD and strong classifier variants used for comparison against ACDC in simulation studies.

### 5.4.1 Performance Under Various Class Overlap Settings

To understand the effects of class overlap on ACDC classification, we generated target class observations we generated observations from  $F_{\mathcal{X}_0} = \text{Uniform}(0, 1)^d$  distribution and observations from  $F_{\mathcal{X}_1} = \text{Uniform}(\Delta, 1 + \Delta)^d$  distribution with  $n_0, n_1 = 500$  observations. We simulate test datasets until all classifiers achieve a standard error below 0.0005 for the Monte Carlo simulations. Class overlap is measured using Equation (3.6).

We first investigate the selection of optimal  $k^*$  and  $\tau^*$  across different overlap levels. Figure (5.3) is the contour plots for the frequency of total AUC optimal counts for each  $(k, \tau)$  combination of ACDC. When classes become more separable, tuning for both  $k$  and  $\tau$  become less important. There does not seem to be a monotonic pattern between  $\tau^*$  and  $k^*$  for ACDC classifiers. For P-CCCD classifiers larger  $\tau$  perform better when classes are separable, even if classes are imbalanced. However, for ACDC, even if  $\tau$  is low a higher  $k$  can offset the model's inability to represent the density of observations in separable spaces.

As overlap increases,  $k^*$  favors higher values of  $k$ , and lower values of  $\tau^*$ . ACDC seemingly does not have the same reliance on  $\tau$  as P-CCCD, instead using smaller more generalizable covers and correcting misclassifications in future iterations.

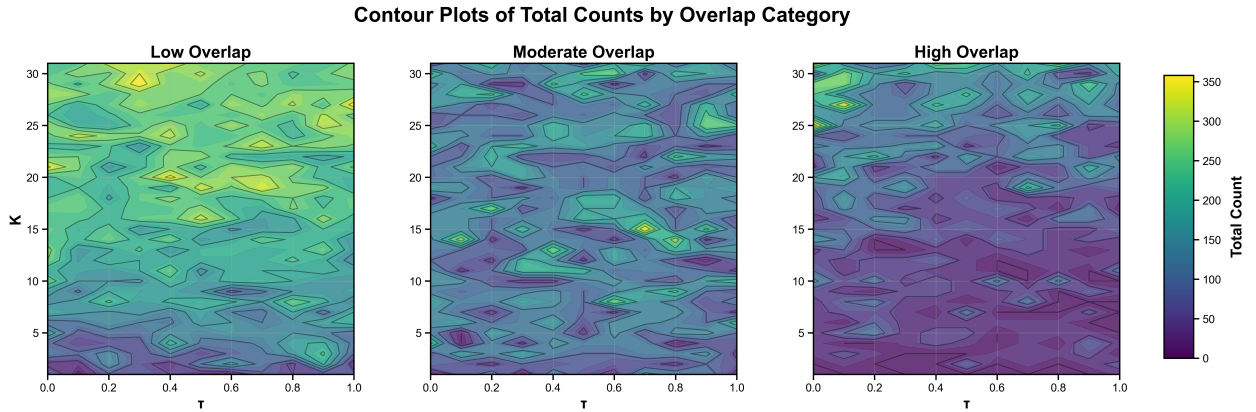


Figure 5.3: Contour plots of optimal AUC counts for each  $(k, \tau)$  combination of ACDC classifiers on balanced, overlapping simulations. As classes become more overlapped, ACDC favors larger  $k$  and smaller  $\tau$ .

Figures (5.4) and (5.5) are the contour plots for the frequency of total  $F_1$ -score and G-mean respectively optimal counts for each  $(k, \tau)$  combination. The effects of the different evaluation metrics for selection of  $k^*$  and  $\tau^*$  are mild. AUC and G-mean have very similar patterns in the contour plot. For  $F_1$ -score there's a large spike in the frequency of optimal  $F_1$ -scores at  $(26, 0.9)$  not present at either AUC or G-mean.

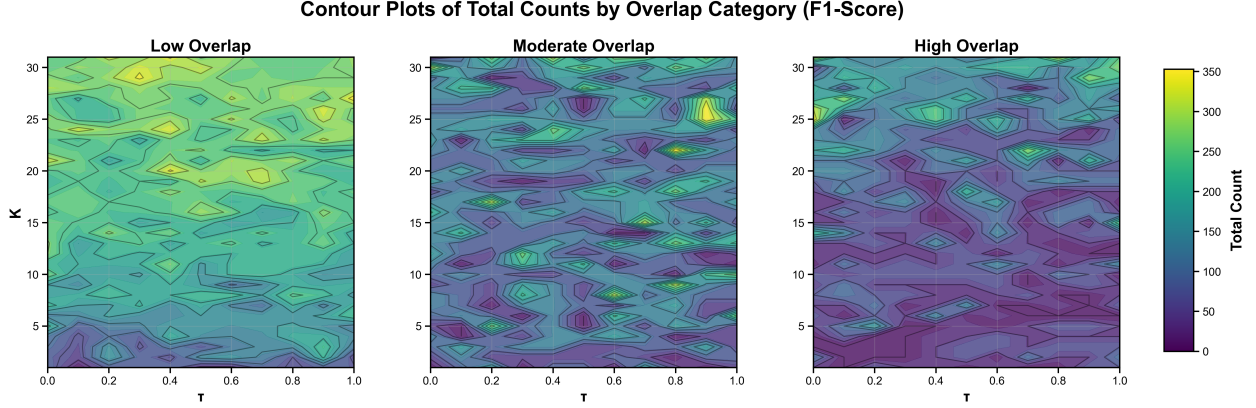


Figure 5.4: Contour plots of optimal  $F_1$ -score counts for each  $(k, \tau)$  combination of ACDC classifiers on balanced, overlapping simulations. As classes become more overlapped, ACDC favors larger  $k$  and smaller  $\tau$ .

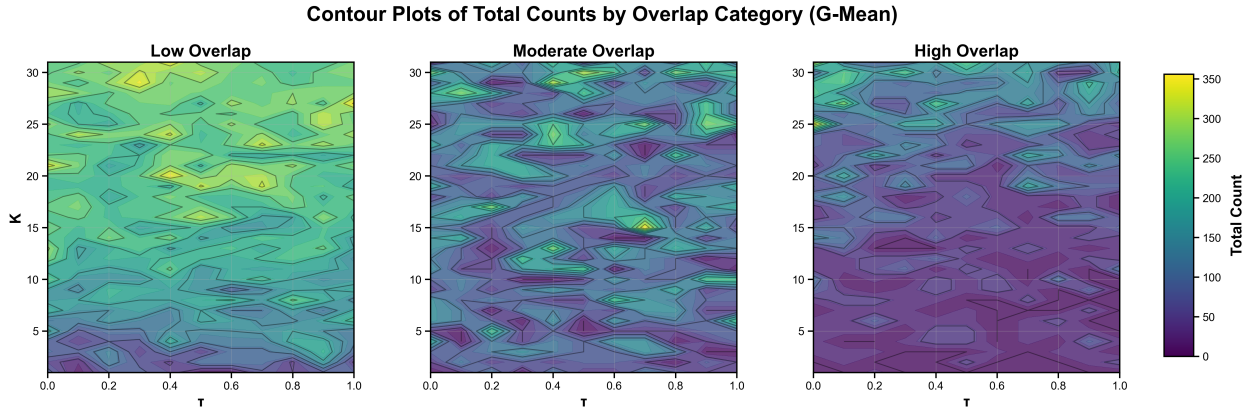


Figure 5.5: Contour plots of optimal G-mean counts for each  $(k, \tau)$  combination of ACDC classifiers on balanced, overlapping simulations. As classes become more overlapped, ACDC favors larger  $k$  and smaller  $\tau$ .

We also provide a multi-metric performance comparison of ACDC classifiers and all other classifiers for each level of overlap. Table (5.2) presents pairwise mean differences between ACDC and each alternative strong classifier under the AUC,  $F_1$ -score, and G-mean evaluation

metrics. Positive values indicate that ACDC outperforms the competitor, while negative values indicate the opposite. ACDC demonstrates consistent competitive performance across all three overlap categories, with its relative advantage varying by evaluation metric. The mean performance differences between ACDC and competitors remain relatively small across all conditions, but increases with higher levels of class overlap. For the AUC metric, ACDC shows marginal improvements over most competitors in the low overlap category, with all mean differences hovering near zero. ACDC maintains slight advantages over several methods including MP-CCCD, SV-CCCD, and P-CCCD for moderate and high class overlap. For  $F_1$ -score, both EC4.5 and C5.0 shows small advantages over ACDC, as indicated by negative mean differences. For G-mean, ACDC maintains relatively neutral performance across all overlap categories. None of the differences were practically significant, however, indicating that ACDC is competitive against other CCCD variants and strong classifiers in balanced, overlapping settings.

| Overlap Category | Metric   | vs MP-CCCD | vs SV-CCCD | vs P-CCCD | vs RW-CCCD | vs Sk-NN | vs SSVM | vs SC4.5 | vs Ek-NN | vs ESVM | vs EC4.5 | vs Ck-NN | vs CSVM | vs C5.0 |
|------------------|----------|------------|------------|-----------|------------|----------|---------|----------|----------|---------|----------|----------|---------|---------|
| Low Overlap      | AUC      | 0.003      | 0.001      | 0.002     | 0.003      | 0.001    | 0.001   | 0.007    | 0.002    | 0.001   | 0.000    | 0.002    | 0.001   | 0.006   |
|                  | F1-Score | 0.003      | 0.001      | 0.002     | 0.002      | 0.001    | 0.001   | 0.007    | 0.002    | 0.001   | 0.000    | 0.002    | 0.001   | 0.006   |
|                  | G-Mean   | 0.003      | 0.001      | 0.002     | 0.002      | 0.001    | 0.001   | 0.007    | 0.002    | 0.001   | 0.000    | 0.002    | 0.001   | 0.006   |
| Moderate Overlap | AUC      | 0.017      | 0.006      | 0.015     | 0.011      | 0.008    | 0.006   | 0.000    | 0.021    | 0.006   | -0.011   | 0.008    | 0.004   | -0.003  |
|                  | F1-Score | 0.018      | 0.006      | 0.016     | 0.011      | 0.009    | 0.006   | 0.001    | 0.021    | 0.005   | -0.011   | 0.006    | 0.004   | -0.002  |
|                  | G-Mean   | 0.017      | 0.007      | 0.016     | 0.011      | 0.008    | 0.006   | 0.001    | 0.022    | 0.006   | -0.011   | 0.007    | 0.004   | -0.002  |
| High Overlap     | AUC      | 0.022      | 0.009      | 0.019     | 0.018      | 0.017    | 0.010   | 0.005    | 0.027    | 0.007   | 0.004    | 0.015    | 0.007   | 0.004   |
|                  | F1-Score | 0.026      | 0.012      | 0.021     | 0.020      | 0.019    | 0.011   | 0.011    | 0.028    | 0.008   | 0.006    | 0.016    | 0.009   | 0.015   |
|                  | G-Mean   | 0.024      | 0.012      | 0.019     | 0.019      | 0.017    | 0.011   | 0.016    | 0.027    | 0.007   | 0.003    | 0.015    | 0.008   | 0.029   |

Table 5.2: Performance difference of ACDC versus CCCD and other strong model variants for balanced data across different scenarios of class overlap and evaluation metrics. Positive differences indicate ACDC performed better.

### 5.4.2 Performance Under Various Class Imbalance and Overlap Settings

To understand the effects of class imbalance and class overlap on SV-CCCD classification, we generated majority class observations from  $F_{\mathcal{X}_0} = \text{Uniform}(0, 1)^d$  distribution and minority class observations from  $F_{\mathcal{X}_1} = \text{Uniform}(\Delta, 1 + \Delta)^d$  distribution with  $n_0 = 500$  and  $n_1 = 500q$  where  $q = 0.25, 0.50, 0.75$  observations. We simulate test datasets until the standard error for the evaluation metric of all classifiers is below 0.0005 for the Monte Carlo simulations.

We begin investigating the effects of imbalance on choosing optimal  $k$  and  $\tau$ . Since the grid search range of  $k$  will vary depending on  $q$ , all visualizations will use a normalized  $k$ , equal to  $k/\max k$ , to provide consistent analysis across all levels of imbalance and overlap. A normalized  $k$  value of 1.0 indicates that the maximum  $k$  was selected.

Figure (5.6) provides the faceted normalized selection rates for optimal AUC for  $k$  and  $\tau$ . In low overlap settings, higher values for  $k^*$  are selected more frequently across all three imbalance levels. However, the difference in selection rate between lower  $k$  and higher  $k$  is not severe until classes are highly imbalanced. When overlap is low, but class imbalance is high, ACDC selects smaller values for  $\tau^*$  at a higher rate. However, under moderate or low imbalance, ACDC selects higher values of  $\tau$  at a larger rate. In moderate overlap settings, there does not seem to be a pattern for selecting  $k^*$  nor  $\tau^*$  across the different imbalance levels. In high overlap, there is a clear monotonic preference for either the highest or close to the highest  $k^*$  values for each imbalance rate. The difference in selection rate is large across all three levels of imbalance. When classes are highly overlapped, each individual metric covering ball volume will be smaller requiring a larger  $k$  to maintain relevant class information.

Surprisingly, there seems to be a decreased selection rate for larger values of  $\tau^*$  under settings of high overlap for all imbalance levels. Local density information is contained in  $\tau$ , as it controls the underlying digraph relationships. Minority covering balls, however, are unlikely to contain multiple class points under high class imbalance levels. In settings of



high class overlap and imbalance, values  $\tau = 0.0, 0.1, 0.2, 0.3$  and  $\tau = 0.4$  were among the highest selection rates. This implies that using a large number of smaller covering balls (that might not necessarily change the underlying digraph structure of the minority class) increases performance. This relationship holds for low and moderate class imbalance levels, but not as extremely. ACDC therefore likely adapts its parameter selection of  $\tau$  based on the degree of class imbalance. These effects are consistent across  $F_1$ -score and G-mean as well. Figures (5.7) and (5.8) give the corresponding selection rates for  $\tau$  and normalized  $k$  under  $F_1$ -score and G-mean respectively.

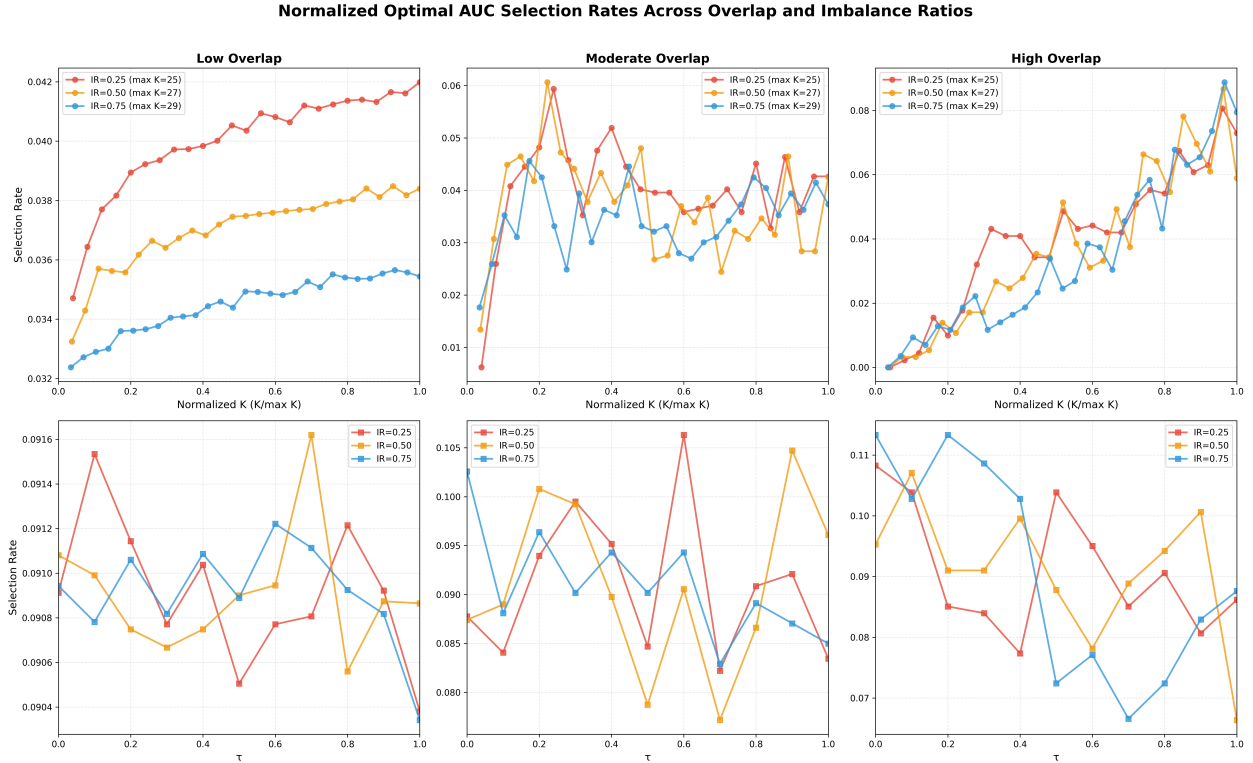


Figure 5.6: Normalized selection rates of  $k$  and  $\tau$  for optimal AUC on imbalanced, overlapping datasets. ACDC selects larger  $k$  for all imbalance levels when class overlap is high.

Table (5.3) provides the mean performance difference of ACDC against CCCD and other strong classifier variants. The differences under low and moderate imbalance settings are not practically different for any combination of overlap and evaluation metric. For data with high class imbalance and moderate overlap, ACDC recorded a large difference in  $F_1$ -score against  $Sk$ -NN, SC4.5, and  $Ek$ -NN models. ACDC had a higher  $F_1$ -score on high imbalance

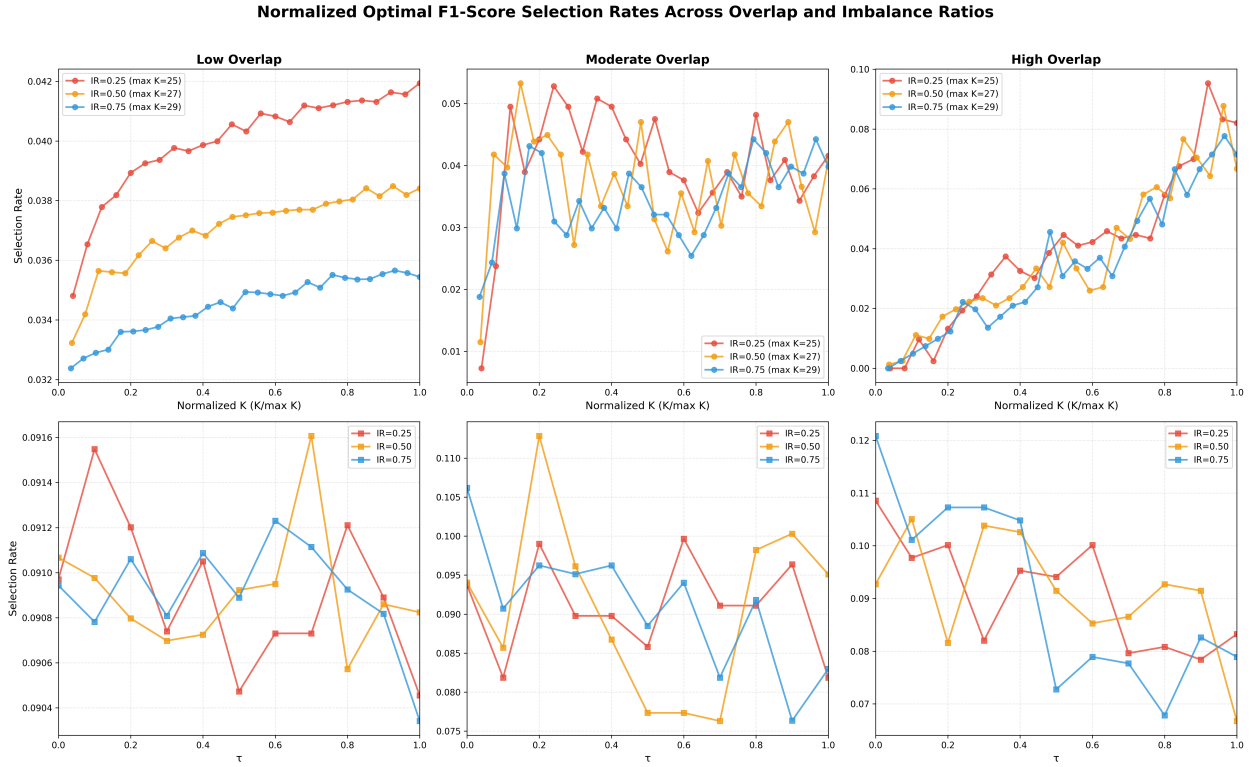


Figure 5.7: Normalized selection rates of  $k$  and  $\tau$  for optimal  $F_1$ -score on imbalanced, overlapping datasets. ACDC selects larger  $k$  for all imbalance levels when class overlap is high.

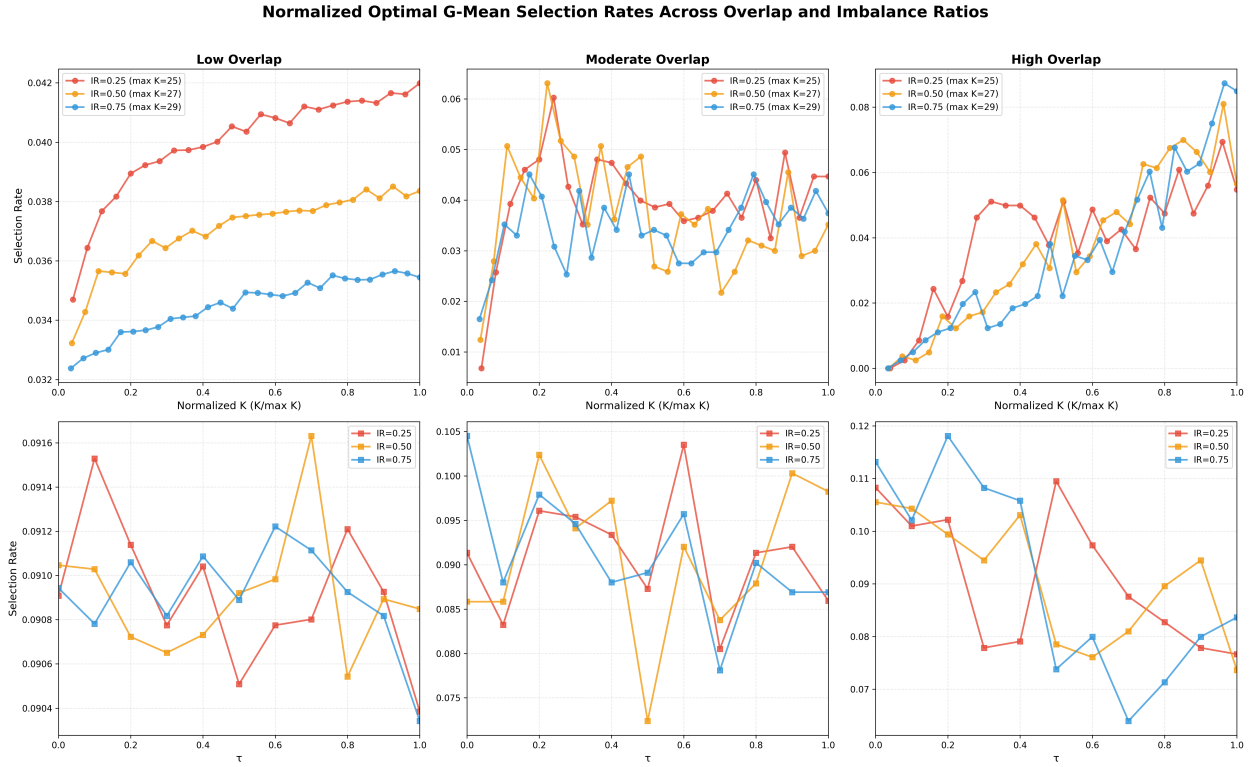


Figure 5.8: Normalized selection rates of  $k$  and  $\tau$  for optimal G-mean on imbalanced, overlapping datasets. ACDC selects larger  $k$  for all imbalance levels when class overlap is high.

and overlapping datasets than all other compared models except SV-CCCD. ACDC also had a large G-mean difference against MP-CCCD for data with high imbalance and overlap.

Compared to other strong models, ACDC is able to more accurately classify minority class observations when classes are extremely imbalanced and overlapped. ACDC constructs localized, class-specific regions that capture the support or boundary of the data distribution. As data becomes more balanced, the effect of class skew diminishes, and the discriminative power of localized boundaries increases, especially in overlapping areas. MP-CCCD are still limited by the lower boundary of the covering ball radii, and thus are unable to learn complex boundary shapes as well as ACDC.

When the classes become more separable, ACDC performs worse compared to EC4.5 and C5.0 classifiers across all metrics. There does not seem to be a practically significant difference in the performance of ACDC with SV-CCCD classifiers under these simulation settings, but ACDC outperformed SV-CCCD in every category.

| Imbalance          | Alpha            | Metric       | vs MP-CCCD   | vs SV-CCCD | vs P-CCCD    | vs RW-CCCD   | vs Sk-NN     | vs SSVM      | vs SC4.5     | vs Ek-NN     | vs ESVM      | vs EC4.5     | vs Ck-NN     | vs CSVM      | vs C5.0      |
|--------------------|------------------|--------------|--------------|------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| Low Imbalance      | Low Overlap      | AUC          | 0.001        | 0.001      | 0.002        | 0.002        | 0.002        | 0.001        | 0.009        | 0.002        | 0.001        | 0.000        | 0.001        | 0.001        | 0.008        |
|                    |                  | $F_1$ -Score | 0.001        | 0.001      | 0.002        | 0.002        | 0.002        | 0.001        | 0.010        | 0.002        | 0.001        | 0.000        | 0.002        | 0.001        | 0.010        |
|                    |                  | G-Mean       | 0.001        | 0.001      | 0.002        | 0.002        | 0.002        | 0.001        | 0.009        | 0.002        | 0.001        | 0.000        | 0.001        | 0.001        | 0.008        |
|                    | Moderate Overlap | AUC          | 0.020        | 0.009      | 0.017        | 0.012        | 0.009        | 0.005        | -0.000       | 0.021        | 0.005        | -0.011       | 0.007        | 0.004        | -0.004       |
|                    |                  | $F_1$ -Score | 0.023        | 0.008      | 0.020        | 0.014        | 0.011        | 0.007        | 0.001        | 0.024        | 0.008        | -0.012       | 0.009        | 0.005        | -0.003       |
|                    |                  | G-Mean       | 0.020        | 0.009      | 0.017        | 0.011        | 0.009        | 0.005        | 0.001        | 0.020        | 0.005        | -0.011       | 0.007        | 0.003        | -0.003       |
|                    | High Overlap     | AUC          | 0.028        | 0.014      | 0.021        | 0.019        | 0.016        | 0.011        | 0.007        | 0.028        | 0.008        | 0.002        | 0.014        | 0.007        | 0.006        |
|                    |                  | $F_1$ -Score | 0.035        | 0.031      | 0.025        | 0.015        | 0.015        | 0.007        | 0.005        | 0.029        | 0.006        | 0.001        | 0.014        | 0.006        | 0.008        |
|                    |                  | G-Mean       | 0.029        | 0.025      | 0.020        | 0.018        | 0.016        | 0.011        | 0.014        | 0.027        | 0.007        | 0.001        | 0.013        | 0.007        | 0.029        |
| Moderate Imbalance | Low Overlap      | AUC          | 0.003        | 0.002      | 0.002        | 0.003        | 0.002        | 0.001        | 0.011        | 0.002        | 0.001        | -0.000       | 0.001        | 0.001        | 0.012        |
|                    |                  | $F_1$ -Score | 0.004        | 0.002      | 0.003        | 0.004        | 0.003        | 0.002        | 0.016        | 0.004        | 0.002        | 0.001        | 0.003        | 0.002        | 0.017        |
|                    |                  | G-Mean       | 0.003        | 0.002      | 0.002        | 0.003        | 0.002        | 0.001        | 0.012        | 0.002        | 0.001        | -0.000       | 0.002        | 0.001        | 0.012        |
|                    | Moderate Overlap | AUC          | 0.021        | 0.010      | 0.015        | 0.009        | 0.008        | 0.003        | -0.001       | 0.017        | 0.004        | -0.015       | 0.004        | 0.002        | -0.002       |
|                    |                  | $F_1$ -Score | 0.030        | 0.009      | 0.022        | 0.019        | 0.022        | 0.014        | 0.008        | 0.034        | 0.015        | -0.012       | 0.011        | 0.009        | 0.003        |
|                    |                  | G-Mean       | 0.022        | 0.012      | 0.016        | 0.009        | 0.008        | 0.004        | -0.001       | 0.017        | 0.005        | -0.015       | 0.005        | 0.002        | -0.002       |
|                    | High Overlap     | AUC          | 0.034        | 0.021      | 0.022        | 0.017        | 0.017        | 0.008        | 0.005        | 0.025        | 0.006        | 0.001        | 0.013        | 0.005        | 0.003        |
|                    |                  | $F_1$ -Score | 0.046        | 0.026      | 0.032        | 0.028        | 0.027        | 0.020        | 0.017        | 0.040        | 0.015        | 0.008        | 0.021        | 0.013        | 0.008        |
|                    |                  | G-Mean       | 0.035        | 0.042      | 0.024        | 0.016        | 0.016        | 0.008        | 0.012        | 0.022        | 0.003        | -0.002       | 0.009        | 0.003        | 0.023        |
| High Imbalance     | Low Overlap      | AUC          | 0.002        | 0.003      | 0.002        | 0.007        | 0.001        | 0.001        | 0.013        | 0.002        | 0.001        | 0.001        | 0.002        | 0.000        | 0.017        |
|                    |                  | $F_1$ -Score | 0.004        | 0.004      | 0.004        | 0.007        | 0.009        | 0.007        | 0.030        | 0.010        | 0.005        | 0.003        | 0.006        | 0.003        | 0.031        |
|                    |                  | G-Mean       | 0.002        | 0.003      | 0.002        | 0.007        | 0.001        | 0.001        | 0.013        | 0.002        | 0.001        | 0.001        | 0.002        | 0.000        | 0.017        |
|                    | Moderate Overlap | AUC          | 0.021        | 0.020      | 0.017        | 0.008        | 0.002        | -0.002       | 0.001        | 0.009        | -0.002       | -0.021       | -0.003       | -0.004       | -0.004       |
|                    |                  | $F_1$ -Score | 0.036        | 0.017      | 0.031        | 0.041        | <b>0.064</b> | 0.047        | <b>0.052</b> | <b>0.073</b> | 0.048        | 0.006        | 0.023        | 0.029        | 0.025        |
|                    |                  | G-Mean       | 0.023        | 0.021      | 0.018        | 0.008        | -0.000       | -0.003       | -0.001       | 0.008        | -0.003       | -0.022       | -0.003       | -0.005       | -0.005       |
|                    | High Overlap     | AUC          | 0.041        | 0.022      | 0.024        | 0.014        | 0.015        | 0.008        | 0.009        | 0.019        | 0.001        | -0.008       | 0.003        | -0.001       | -0.009       |
|                    |                  | $F_1$ -Score | <b>0.078</b> | 0.005      | <b>0.055</b> | <b>0.100</b> | <b>0.100</b> | <b>0.097</b> | <b>0.099</b> | <b>0.112</b> | <b>0.082</b> | <b>0.067</b> | <b>0.072</b> | <b>0.071</b> | <b>0.061</b> |
|                    |                  | G-Mean       | <b>0.053</b> | 0.047      | 0.031        | 0.007        | 0.006        | 0.000        | 0.004        | 0.011        | -0.007       | -0.017       | -0.005       | -0.009       | 0.001        |

Table 5.3: Performance difference of ACDC versus CCCD and other strong classifiers for imbalanced data across different scenarios of class overlap and evaluation metrics. Positive differences indicate ACDC performed better.

In order to understand the effects of dimensionality on ACDC classifiers, we generated target class observations from the distribution  $F_{\mathcal{X}_0} = \text{Uniform}(0, 1)^{500}$  and non-target class observations from  $F_{\mathcal{X}_1} = \text{Uniform}(0.25, 1.25)^{500}$ , with  $n_0 = 250$  and  $n_1 = 250q = 250(0.25) \approx 63$  observations, respectively. A pilot study with 200 replications was conducted to identify optimal hyperparameters, followed by a main Monte Carlo simulation under identical conditions. Test datasets were generated until the standard error for each evaluation metric fell below 0.0005.

Results are omitted, as ACDC converged to the maximum AUC,  $F_1$ -score, and G-mean within just a few iterations of the main simulation setting following the pilot study. The results from Chapter 3 and now suggest that both CCCD-based classifiers are robust to the “curse of dimensionality”.

To evaluate the practical efficiency of ACDC, we benchmarked the runtime against all selected classifiers under the previous run-time experimental settings. Each configuration was run for 200 iterations, and mean runtimes along with standard deviations were recorded. Results are summarized in Table (5.4). All computations were performed using the Auburn University Easley Cluster. ACDC classifiers had an average runtime smaller than P-CCCD, RW-CCCD, and MP-CCCD. It’s mean runtime was only slightly longer than SV-CCCD. We observe that increasing  $k$  from 1 to  $\sqrt{N}$  increases the model’s runtime but not significantly. ACDC was also significantly faster than other strong ensemble methods.

A quick summary for all Monte Carlo simulation results is provided below,

- (i) Higher values of  $k^*$  are selected across all levels of class imbalance and overlap. Selection of  $\tau^*$  is more important depending on the degree of class imbalance.
- (ii) ACDC performed better under almost all class imbalance and overlap settings than the state-of-the-art classifiers. For  $F_1$ -scores in high class imbalance and overlap settings, the results were practically significant for all but SV-CCCD.
- (iii) ACDC had faster runtimes than all other EasyEnsemble and SMOTE+ENN variants.

| Index             | Mean $\pm$ SD         |
|-------------------|-----------------------|
| ACDC - 1          | 0.0809 $\pm$ 0.0308   |
| ACDC - $\sqrt{N}$ | 0.0900 $\pm$ 0.0328   |
| P-CCCD            | 0.1681 $\pm$ 0.1964   |
| RW-CCCD           | 0.1435 $\pm$ 0.1363   |
| BP-CCCD           | 2.3978 $\pm$ 3.5584   |
| MP-CCCD           | 3.4688 $\pm$ 5.2414   |
| SV-CCCD           | 0.0611 $\pm$ 0.0273   |
| $Sk$ -NN          | 9.4283 $\pm$ 12.1143  |
| SSVM              | 11.7459 $\pm$ 11.8681 |
| SC4.5             | 10.0818 $\pm$ 13.8776 |
| $Ek$ -NN          | 7.7427 $\pm$ 9.2585   |
| ESVM              | 0.8749 $\pm$ 0.9784   |
| EC4.5             | 0.4209 $\pm$ 0.5447   |
| $Ck$ -NN          | 0.0492 $\pm$ 0.0139   |
| CSVM              | 0.0215 $\pm$ 0.0175   |
| C5.0              | 0.0217 $\pm$ 0.0109   |

Table 5.4: Aggregated mean and standard deviations for ACDC and other selected strong classifiers runtimes across simulated data settings.

## 5.5 Real Data Experiments

We now consider the performance of ACDC on real data settings. We use the same datasets and methods considered in Chapters 3 and 4. Tables (5.5), (5.6), and (5.7) present the mean and standard deviation AUC,  $F_1$ -score, and G-mean respectively for ACDC against other comparative CCCD and strong classifier variants. The results of the  $5 \times 2$  CV  $F$ -test are presented in parenthesis below where applicable. Values in blue represent the highest average performer out of CCCD variants, while values in red represent the highest average overall.

ACDC was the highest performing CCCD variant for Yeast5, Yeast4, Yeast1289vs7, Shuttle, Abalone, and Segment datasets only being beaten by SV-CCCD for Vowel and Pageblocks datasets. For both datasets, the difference between ACDC and SV-CCCD were not statistically significant ( $p = 0.4044$  and  $p = 0.6792$  respectively). ACDC was statistically significantly better than MP-CCCD for the Ionosphere ( $p = 0.0562$ ), Yeast4 ( $p = 0.0590$ ),

Segment ( $p = 0.0024$ ) and Pageblocks ( $p = 0.0157$ ) datasets. ACDC was statistically significantly better than SV-CCCD on Ionosphere ( $p = 0.0145$ ), Yeast6 ( $p = 0.0491$ ), Yeast5 ( $p = 0.007$ ), Yeast4 ( $p = 0.0228$ ), Abalone ( $p = 0.0389$ ), and Segment ( $p < 0.0001$ ) datasets. None of the RW-CCCD or P-CCCD variants outscored ACDC. Out of the highest performing strong models, ACDC was only statistically significantly different from  $Ek$ -NN on Abalone ( $p = 0.0173$ ) and EC4.5 on Pageblocks ( $p = 0.0024$ ).

ACDC recorded the highest average  $F_1$ -scores for Ionosphere, Sonar, Yeast6, Yeast5, Yeast4, Yeast1289vs7, and Vowel datasets. For the Abalone dataset ACDC was tied with RW-CCCD, MP-CCCD,  $Ck$ -NN, and C5.0 as the highest scoring model.

ACDC had the highest average  $F_1$ -score for all but the Shuttle, Segment, and Pageblocks datasets. For those three datasets, it was the highest average performing CCCD variant, however. Only Pageblocks had a selected classifier which had a statistically significantly higher average  $F_1$ -score than ACDC. ACDC was significantly outperformed on the Pageblocks dataset by EC4.5 for AUC, but C5.0 for  $F_1$ -score. ACDC outperformed MP-CCCD significantly for Ionosphere, Vowel, and Segment datasets while outperforming SV-CCCD significantly on Ionosphere, Sonar, Yeast6, Yeast5, Abalone, and Segment datasets.

Table 5.7 presents the G-mean results for each of the datasets. ACDC had the highest performing G-mean for only the Ionosphere and Sonar data, but was the highest performing CCCD variant for Yeast6, Yeast5, Yeast4, Yeast1289vs7, Shuttle, and Segment datasets. SV-CCCD had a higher average G-mean on the Vowel and Pageblocks dataset, but was not statistically significant ( $p = 0.4250$  and  $p = 0.7143$ ). Against other strong classifiers, only  $Ek$ -NN performed statistically significantly better on the Abalone dataset ( $p = 0.0051$ ).

The results indicate that ACDC does well at minority class classification without a large amount of false positives. Such points, especially those near the boundary where class overlap is greatest, are likely misclassified during early iterations and are more likely to be stochastically chosen as covering ball centers in subsequent iterations. In terms of performance, ACDC performs better than other CCCD variants, and is at least practically



competitive with other strong models only having significantly different results on Abalone and Pageblocks datasets. Both the Abalone and Pageblocks datasets had the highest number of observations of the selected datasets with  $N = 4174$  and  $N = 5472$  respectively. Abalone had both high global ( $q = 129.40$ ) and local imbalance while Pageblocks had a global imbalance of  $q = 8.79$  with little local imbalance.

It is unlikely, however, that the data set cardinality is the only factor contributing to the performance difference as ACDC was able to perform as well as, or in some instances better than, other strong classifiers on datasets like Segment, Yeast4, Yeast5, Yeast6, Yeast7, Yeast1289vs7 where  $N > 1000$ , and Shuttle with  $N > 2000$ . For the Abalone dataset specifically,  $k$ -NN variants of either  $Ek$ -NN or  $Ck$ -NN were the best performers outside of ACDC for AUC and G-mean. Meanwhile for Pageblocks it was either EC4.5 or C5.0. Given the higher performance consistency being based on a specific underlying model family for both Abalone and Pageblocks, it is likely the underlying weak model was more suited to the latent underlying class structures comparative to ACDC rather than the models superior.

We now provide a brief summary of the results from the real dataset settings,

- (i) ACDC was competitive against all other state-of-the-art classification methods for all evaluation metrics.
- (ii) The only dataset that ACDC was statistically outperformed by a decision tree variant on for each metric was the Pageblocks dataset. However, based on the dataset description, this is probably due to underlying characteristics of the dataset.
- (iii) ACDC does particularly well with minority class classification without over predicting minority class observations, as is evident by the high  $F_1$ -score and G-mean scores.

|                    | Ionosphere         | Sonar              | Yeast6             | Yeast5             | Yeast4             | Yeast1289vs7       | Vowel              | Shuttle            | Abalone            | Sement             | Pageblocks         |
|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|
| ACDC               | <b>0.965±0.019</b> | <b>0.866±0.019</b> | <b>0.891±0.067</b> | <b>0.919±0.045</b> | <b>0.801±0.058</b> | <b>0.682±0.057</b> | 0.982±0.026        | <b>0.998±0.005</b> | <b>0.605±0.049</b> | <b>0.991±0.002</b> | 0.901±0.006        |
| RW-CCCD            | 0.904±0.021        | 0.730±0.055        | 0.834±0.054        | 0.858±0.077        | 0.771±0.056        | 0.657±0.067        | 0.870±0.051        | 0.947±0.105        | 0.622±0.049        | 0.910±0.025        | 0.875±0.019        |
| (p-value vs. ACDC) | (0.0125)           | (0.0193)           | (0.1311)           | (0.6370)           | (0.2886)           | (0.6097)           | (0.0485)           | (0.4789)           | (0.0452)           | (0.0117)           | (0.1647)           |
| P-CCCD             | 0.926±0.028        | 0.795±0.052        | 0.755±0.054        | 0.792±0.095        | 0.598±0.051        | 0.564±0.027        | 0.965±0.031        | 0.988±0.016        | 0.539±0.037        | 0.957±0.01         | 0.869±0.009        |
| (p-value vs. ACDC) | (0.0356)           | (0.2197)           | (0.1676)           | (0.0756)           | (0.0249)           | (0.1178)           | (0.0912)           | (0.5196)           | (0.3086)           | (0.0103)           | (0.0154)           |
| MP-CCCD            | 0.946±0.025        | 0.820±0.046        | 0.766±0.039        | 0.853±0.073        | 0.653±0.048        | 0.638±0.048        | 0.969±0.035        | 0.991±0.013        | 0.517±0.021        | 0.963±0.008        | 0.869±0.009        |
| (p-value vs. ACDC) | (0.0562)           | (0.4668)           | (0.1875)           | (0.1745)           | (0.0590)           | (0.2982)           | (0.3795)           | (0.5335)           | (0.0903)           | (0.0024)           | (0.0157)           |
| SV-CCCD            | 0.906±0.030        | 0.736±0.056        | 0.648±0.048        | 0.713±0.068        | 0.589±0.033        | 0.591±0.058        | <b>0.983±0.033</b> | 0.984±0.012        | 0.506±0.015        | 0.590±0.011        | <b>0.906±0.016</b> |
| (p-value vs. ACDC) | (0.0145)           | (0.0663)           | (0.0491)           | (0.0007)           | (0.0228)           | (0.1802)           | <b>(0.4044)</b>    | (0.3161)           | (0.0389)           | (<0.0001)          | <b>(0.6792)</b>    |
| <i>Sk</i> -NN      | 0.838±0.022        | 0.739±0.063        | 0.882±0.040        | 0.963±0.018        | 0.802±0.049        | 0.636±0.059        | 0.940±0.040        | 0.999±0.001        | 0.573±0.032        | 0.978±0.007        | 0.816±0.012        |
| (p-value vs. ACDC) | (0.0014)           | (0.0218)           | (0.6956)           | (0.2423)           | (0.8077)           | (0.6689)           | (0.0087)           | (0.5673)           | (0.4617)           | (0.0485)           | (0.0004)           |
| SSVM               | 0.948±0.010        | 0.636±0.045        | 0.864±0.032        | 0.947±0.031        | 0.803±0.025        | 0.680±0.043        | <b>0.988±0.016</b> | 0.998±0.001        | 0.522±0.017        | 0.987±0.004        | 0.856±0.011        |
| (p-value vs. ACDC) | (0.2612)           | (0.0033)           | (0.6771)           | (0.4876)           | (0.8139)           | (0.7752)           | <b>(0.4458)</b>    | (0.5305)           | (0.1162)           | (0.2431)           | (0.0006)           |
| SC4.5              | 0.866±0.022        | 0.707±0.033        | 0.764±0.056        | 0.883±0.073        | 0.660±0.065        | 0.595±0.041        | 0.944±0.028        | 0.998±0.003        | 0.503±0.019        | 0.982±0.005        | 0.925±0.012        |
| (p-value vs. ACDC) | (0.0142)           | (0.0056)           | (0.0591)           | (0.1306)           | (0.1725)           | (0.3024)           | (0.2438)           | (0.5528)           | (0.0894)           | (0.1032)           | (0.0844)           |
| <i>Ek</i> -NN      | 0.858±0.021        | 0.817±0.016        | 0.890±0.038        | 0.965±0.004        | <b>0.865±0.022</b> | <b>0.750±0.040</b> | 0.965±0.008        | 0.996±0.004        | <b>0.731±0.031</b> | 0.957±0.004        | 0.914±0.006        |
| (p-value vs. ACDC) | (0.0030)           | (0.0202)           | (0.7362)           | (0.3236)           | <b>(0.1705)</b>    | <b>(0.1012)</b>    | (0.4992)           | (0.5491)           | <b>(0.0173)</b>    | (0.0007)           | (0.0039)           |
| ESVM               | 0.802±0.026        | 0.649±0.049        | 0.854±0.040        | 0.954±0.007        | 0.825±0.029        | 0.619±0.034        | 0.870±0.043        | 0.994±0.004        | 0.640±0.044        | 0.895±0.017        | 0.519±0.006        |
| (p-value vs. ACDC) | (0.0003)           | (0.0014)           | (0.3310)           | (0.4908)           | (0.4100)           | (0.4291)           | (0.0131)           | (0.4475)           | (0.6691)           | (0.0024)           | (<0.0001)          |
| EC4.5              | 0.897±0.013        | 0.783±0.024        | 0.855±0.031        | 0.947±0.010        | 0.833±0.037        | 0.621±0.063        | 0.965±0.009        | 0.999±0.001        | 0.719±0.04         | 0.992±0.003        | <b>0.952±0.006</b> |
| (p-value vs. ACDC) | (0.0315)           | (0.0016)           | (0.4868)           | (0.5139)           | (0.5747)           | (0.4234)           | (0.4642)           | (0.5806)           | (0.0092)           | (0.3576)           | <b>(0.0024)</b>    |
| <i>Ck</i> -NN      | 0.873±0.018        | 0.825±0.025        | 0.876±0.045        | <b>0.970±0.010</b> | 0.823±0.027        | 0.683±0.047        | 0.980±0.033        | 0.999±0.000        | 0.572±0.035        | <b>0.994±0.001</b> | 0.889±0.010        |
| (p-value vs. ACDC) | (<0.0001)          | (0.1150)           | (0.6536)           | <b>(0.1999)</b>    | (0.7375)           | (0.7510)           | (0.5480)           | (0.5660)           | (0.4087)           | <b>(0.4662)</b>    | (0.3199)           |
| CSVM               | 0.951±0.010        | 0.735±0.062        | 0.835±0.042        | 0.948±0.025        | 0.766±0.045        | 0.690±0.062        | 0.985±0.030        | 0.984±0.012        | 0.685±0.057        | 0.590±0.011        | 0.605±0.006        |
| (p-value vs. ACDC) | (0.3009)           | (0.0822)           | (0.5312)           | (0.5550)           | (0.5196)           | (0.8266)           | (0.5800)           | (0.3161)           | (0.1073)           | (<0.0001)          | (<0.0001)          |
| C5.0               | 0.907±0.020        | 0.768±0.042        | 0.730±0.035        | 0.821±0.069        | 0.590±0.042        | 0.567±0.051        | 0.948±0.027        | <b>1.000±0.000</b> | 0.503±0.010        | 0.992±0.003        | 0.926±0.016        |
| (p-value vs. ACDC) | (0.0307)           | (0.0078)           | (0.0647)           | (0.0617)           | (0.0241)           | (0.2947)           | (0.1456)           | <b>(0.5336)</b>    | (0.0371)           | (0.6329)           | (0.1698)           |

Table 5.5: Mean and standard deviation of ACDC and chosen models' AUC across chosen real data. Values highlighted in red indicate the best performer of all models for a specific dataset. Values highlighted in blue indicate the best performer out of CCCD variant models. The results of the relevant 5x2 CV  $F$ -test are presented in parenthesis when relevant.

|                    | Ionosphere  | Sonar       | Yeast6      | Yeast5      | Yeast4      | Yeast1289vs7 | Vowel       | Shuttle     | Abalone     | Segment     | Pageblocks  |
|--------------------|-------------|-------------|-------------|-------------|-------------|--------------|-------------|-------------|-------------|-------------|-------------|
| ACDC               | 0.976±0.012 | 0.858±0.026 | 0.717±0.057 | 0.77±0.071  | 0.474±0.062 | 0.364±0.075  | 0.979±0.027 | 0.997±0.004 | 0.996±0.000 | 0.985±0.005 | 0.796±0.010 |
| RW-CCCD            | 0.924±0.016 | 0.726±0.055 | 0.450±0.065 | 0.574±0.088 | 0.334±0.045 | 0.177±0.048  | 0.628±0.067 | 0.834±0.308 | 0.996±0.000 | 0.706±0.035 | 0.676±0.199 |
| (p-value vs. ACDC) | (0.0031)    | (0.0254)    | (0.0207)    | (0.0573)    | (0.0104)    | (0.0024)     | (0.0013)    | (0.4761)    | (0.1666)    | (0.0004)    | (0.0014)    |
| P-CCCD             | 0.947±0.017 | 0.777±0.057 | 0.556±0.089 | 0.600±0.132 | 0.250±0.101 | 0.165±0.070  | 0.924±0.059 | 0.978±0.020 | 0.994±0.001 | 0.927±0.012 | 0.786±0.011 |
| (p-value vs. ACDC) | (0.0124)    | (0.2408)    | (0.1954)    | (0.0626)    | (0.1288)    | (<0.0001)    | (0.2377)    | (0.4184)    | (0.0578)    | (0.0007)    | (0.4394)    |
| MP-CCCD            | 0.960±0.017 | 0.810±0.048 | 0.582±0.076 | 0.693±0.076 | 0.358±0.102 | 0.303±0.102  | 0.956±0.016 | 0.983±0.016 | 0.996±0.000 | 0.931±0.013 | 0.788±0.010 |
| (p-value vs. ACDC) | (0.0623)    | (0.3694)    | (0.2060)    | (0.0844)    | (0.2092)    | (0.5486)     | (<0.0001)   | (0.3724)    | (0.1766)    | (0.0012)    | (0.4643)    |
| SV-CCCD            | 0.939±0.019 | 0.652±0.091 | 0.393±0.102 | 0.522±0.109 | 0.258±0.073 | 0.274±0.154  | 0.974±0.037 | 0.982±0.012 | 0.993±0.001 | 0.304±0.031 | 0.828±0.017 |
| (p-value vs. ACDC) | (0.0037)    | (0.0761)    | (0.0570)    | (0.0157)    | (0.1088)    | (0.1063)     | (0.5456)    | (0.2097)    | (0.0067)    | (<0.0001)   | (0.1080)    |
| <i>Sk</i> -NN      | 0.905±0.012 | 0.708±0.058 | 0.441±0.038 | 0.682±0.052 | 0.386±0.028 | 0.242±0.082  | 0.902±0.049 | 0.980±0.013 | 0.991±0.003 | 0.934±0.010 | 0.791±0.014 |
| (p-value vs. ACDC) | (0.0057)    | (0.0217)    | (0.0085)    | (0.4319)    | (0.1111)    | (0.1223)     | (0.0510)    | (0.3108)    | (0.1298)    | (0.0004)    | (0.5861)    |
| SSVM               | 0.960±0.006 | 0.448±0.141 | 0.590±0.037 | 0.754±0.039 | 0.397±0.041 | 0.345±0.103  | 0.949±0.032 | 0.976±0.011 | 0.993±0.002 | 0.974±0.012 | 0.719±0.056 |
| (p-value vs. ACDC) | (0.0424)    | (0.0324)    | (0.0525)    | (0.7793)    | (0.0026)    | (0.7578)     | (0.2674)    | (0.0526)    | (0.1244)    | (0.4098)    | (0.2074)    |
| SC4.5              | 0.913±0.016 | 0.663±0.059 | 0.484±0.104 | 0.658±0.068 | 0.309±0.095 | 0.218±0.076  | 0.836±0.045 | 0.969±0.033 | 0.993±0.002 | 0.966±0.012 | 0.835±0.017 |
| (p-value vs. ACDC) | (0.0028)    | (0.0015)    | (0.0550)    | (0.0204)    | (0.1799)    | (0.0204)     | (0.0067)    | (0.5040)    | (0.1597)    | (0.0701)    | (0.0493)    |
| <i>Ek</i> -NN      | 0.919±0.011 | 0.799±0.031 | 0.282±0.023 | 0.461±0.027 | 0.319±0.036 | 0.204±0.043  | 0.741±0.037 | 0.996±0.004 | 0.818±0.050 | 0.806±0.019 | 0.719±0.012 |
| (p-value vs. ACDC) | (0.0027)    | (0.2112)    | (0.0007)    | (0.0075)    | (0.0011)    | (0.0037)     | (0.0008)    | (0.7710)    | (0.0026)    | (0.0002)    | (0.0026)    |
| ESVM               | 0.890±0.017 | 0.643±0.040 | 0.181±0.026 | 0.391±0.030 | 0.268±0.034 | 0.087±0.023  | 0.590±0.046 | 0.968±0.010 | 0.711±0.128 | 0.630±0.025 | 0.686±0.020 |
| (p-value vs. ACDC) | (0.0026)    | (0.0029)    | (0.0007)    | (0.0038)    | (0.0011)    | (0.0005)     | (<0.0001)   | (0.0660)    | (0.0513)    | (<0.0001)   | (0.0047)    |
| EC4.5              | 0.931±0.015 | 0.783±0.044 | 0.247±0.030 | 0.431±0.043 | 0.238±0.029 | 0.095±0.021  | 0.789±0.052 | 0.989±0.012 | 0.793±0.036 | 0.966±0.012 | 0.798±0.02  |
| (p-value vs. ACDC) | (0.0004)    | (0.0232)    | (0.0010)    | (0.0068)    | (0.0040)    | (0.0019)     | (0.0105)    | (0.6532)    | (0.0019)    | (0.1914)    | (0.7873)    |
| <i>Ck</i> -NN      | 0.929±0.010 | 0.810±0.026 | 0.470±0.052 | 0.568±0.046 | 0.456±0.048 | 0.281±0.052  | 0.963±0.054 | 0.988±0.007 | 0.996±0.000 | 0.976±0.006 | 0.798±0.012 |
| (p-value vs. ACDC) | (0.0003)    | (0.0625)    | (0.0005)    | (0.0541)    | (0.1898)    | (0.1590)     | (0.4141)    | (0.1150)    | (0.0666)    | (0.3827)    | (0.7221)    |
| CSVM               | 0.963±0.008 | 0.650±0.100 | 0.525±0.042 | 0.743±0.028 | 0.369±0.074 | 0.205±0.041  | 0.978±0.033 | 0.983±0.012 | 0.994±0.000 | 0.304±0.031 | 0.815±0.009 |
| (p-value vs. ACDC) | (0.0355)    | (0.0907)    | (0.0282)    | (0.6788)    | (0.1527)    | (0.0599)     | (0.3126)    | (0.3662)    | (0.0273)    | (0.0001)    | (0.3123)    |
| C5.0               | 0.944±0.013 | 0.744±0.041 | 0.519±0.054 | 0.683±0.109 | 0.243±0.083 | 0.195±0.128  | 0.916±0.048 | 1.000±0.000 | 0.996±0.001 | 0.987±0.004 | 0.876±0.015 |
| (p-value vs. ACDC) | (0.0411)    | (0.0024)    | (0.0140)    | (0.0585)    | (0.0696)    | (0.1043)     | (0.0574)    | (0.5348)    | (0.7190)    | (0.4069)    | (0.0001)    |

Table 5.6: Mean and standard deviation of ACDC and other chosen models'  $F_1$  scores across chosen real data. Values highlighted in red indicate the best performer of all models for a specific dataset. Values highlighted in blue indicate the best performer out of CCCD variant models. The results of the relevant 5x2 CV  $F$ -test are presented in parenthesis.

|                    | Ionosphere         | Sonar              | Yeast6             | Yeast5             | Yeast4             | Yeast1289vs7       | Vowel              | Shuttle            | Abalone               | Segment               | Pageblocks         |
|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|-----------------------|-----------------------|--------------------|
| ACDC               | <b>0.966±0.015</b> | <b>0.869±0.027</b> | <b>0.882±0.074</b> | <b>0.910±0.066</b> | <b>0.790±0.068</b> | <b>0.642±0.085</b> | 0.979±0.027        | <b>0.999±0.003</b> | 0.541±0.056           | <b>0.991±0.002</b>    | 0.900±0.005        |
| RW-CCCD            | 0.904±0.021        | 0.724±0.057        | 0.393±0.165        | 0.845±0.098        | 0.752±0.070        | 0.599±0.117        | 0.866±0.055        | 0.938±0.131        | <b>0.609±0.059</b>    | 0.908±0.025           | 0.677±0.200        |
| (p-value vs. ACDC) | (0.0038)           | (0.0204)           | (0.0272)           | (0.6875)           | (0.4505)           | (0.6768)           | (0.0689)           | (0.4789)           | ( <b>&lt;0.0001</b> ) | (0.0094)              | (0.0029)           |
| P-CCCD             | 0.925±0.028        | 0.793±0.053        | 0.715±0.078        | 0.758±0.129        | 0.450±0.116        | 0.403±0.093        | 0.964±0.032        | 0.988±0.016        | 0.272±0.229           | 0.956±0.011           | 0.862±0.011        |
| (p-value vs. ACDC) | (0.0614)           | (0.3081)           | (0.2026)           | (0.0821)           | (0.0398)           | (0.1445)           | (0.2613)           | (0.5207)           | (0.4159)              | (0.0165)              | (0.8104)           |
| MP-CCCD            | 0.945±0.025        | 0.824±0.039        | 0.730±0.055        | 0.697±0.096        | 0.573±0.058        | 0.526±0.110        | 0.968±0.036        | 0.991±0.013        | 0.135±0.146           | 0.963±0.008           | 0.863±0.011        |
| (p-value vs. ACDC) | (0.0769)           | (0.5187)           | (0.1951)           | (0.0028)           | (0.0148)           | (0.2700)           | (0.5193)           | (0.5332)           | (0.0030)              | (0.0043)              | (0.0066)           |
| SV-CCCD            | 0.904±0.031        | 0.697±0.073        | 0.540±0.097        | 0.647±0.104        | 0.425±0.075        | 0.398±0.175        | <b>0.983±0.034</b> | 0.984±0.012        | 0.195±0.140           | 0.423±0.025           | <b>0.903±0.017</b> |
| (p-value vs. ACDC) | (0.0039)           | (0.1081)           | (0.0616)           | (0.0032)           | (0.0044)           | (0.0048)           | ( <b>0.4250</b> )  | (0.3376)           | ( <b>&lt;0.0001</b> ) | ( <b>&lt;0.0001</b> ) | ( <b>0.7143</b> )  |
| <i>Sk</i> -NN      | 0.819±0.019        | 0.712±0.073        | 0.875±0.046        | 0.970±0.016        | 0.781±0.060        | 0.545±0.113        | 0.939±0.047        | 0.999±0.001        | 0.392±0.081           | 0.977±0.007           | 0.906±0.014        |
| (p-value vs. ACDC) | (0.0011)           | (0.0105)           | (0.7240)           | (0.3316)           | (0.6429)           | (0.3634)           | (0.2154)           | (0.5591)           | (0.0669)              | (0.0214)              | (0.4771)           |
| SSVM               | 0.947±0.012        | 0.527±0.081        | 0.860±0.045        | 0.948±0.026        | 0.770±0.051        | 0.646±0.057        | <b>0.986±0.017</b> | 0.998±0.001        | 0.204±0.147           | 0.987±0.004           | 0.905±0.010        |
| (p-value vs. ACDC) | (0.066)            | (0.0027)           | (0.6457)           | (0.5547)           | (0.6791)           | (0.6577)           | ( <b>0.4302</b> )  | (0.4285)           | (0.0566)              | (0.3286)              | (0.4634)           |
| SC4.5              | 0.841±0.021        | 0.660±0.052        | 0.748±0.057        | 0.836±0.091        | 0.579±0.140        | 0.436±0.082        | 0.945±0.029        | 0.997±0.003        | 0.075±0.120           | 0.982±0.004           | 0.922±0.013        |
| (p-value vs. ACDC) | (0.0032)           | (0.1081)           | (0.0490)           | (0.2562)           | (0.0303)           | (0.0734)           | (0.1935)           | (0.6082)           | (0.0045)              | (0.0841)              | (0.3662)           |
| <i>Ek</i> -NN      | 0.847±0.024        | 0.817±0.025        | <b>0.888±0.041</b> | 0.964±0.005        | <b>0.858±0.024</b> | <b>0.730±0.037</b> | 0.962±0.009        | 0.996±0.004        | <b>0.710±0.023</b>    | 0.957±0.004           | 0.915±0.003        |
| (p-value vs. ACDC) | (0.0035)           | (0.0105)           | ( <b>0.7886</b> )  | (0.4274)           | ( <b>0.1764</b> )  | ( <b>0.1861</b> )  | (0.3074)           | (0.5543)           | ( <b>0.0051</b> )     | (0.0003)              | (0.3653)           |
| ESVM               | 0.791±0.033        | 0.637±0.038        | 0.846±0.031        | 0.949±0.005        | 0.826±0.038        | 0.616±0.075        | 0.865±0.041        | 0.994±0.004        | 0.477±0.227           | 0.891±0.020           | 0.875±0.018        |
| (p-value vs. ACDC) | (0.0010)           | (0.0012)           | (0.6078)           | (0.5528)           | (0.2621)           | (0.3523)           | (0.0049)           | (0.3522)           | (0.2982)              | (0.0038)              | (0.7261)           |
| EC4.5              | 0.906±0.016        | 0.791±0.043        | 0.860±0.039        | 0.953±0.011        | 0.82±0.033         | 0.616±0.078        | 0.963±0.011        | 0.999±0.001        | 0.718±0.039           | 0.993±0.002           | <b>0.957±0.006</b> |
| (p-value vs. ACDC) | (0.0075)           | (0.0807)           | (0.6590)           | (0.4735)           | (0.6173)           | (0.6303)           | (0.4752)           | (0.5274)           | (0.0041)              | (0.1240)              | ( <b>0.0992</b> )  |
| <i>Ck</i> -NN      | 0.866±0.021        | 0.823±0.025        | 0.872±0.05         | <b>0.970±0.010</b> | 0.813±0.031        | 0.648±0.071        | 0.980±0.034        | 0.999±0.000        | 0.428±0.077           | <b>0.994±0.001</b>    | 0.891±0.014        |
| (p-value vs. ACDC) | (0.0003)           | (0.2400)           | (0.7633)           | ( <b>0.3248</b> )  | (0.5401)           | (0.6412)           | (0.6946)           | (0.5889)           | (0.0514)              | ( <b>0.4630</b> )     | (0.6459)           |
| CSVM               | 0.950±0.010        | 0.696±0.081        | 0.825±0.049        | 0.947±0.026        | 0.750±0.056        | 0.667±0.078        | 0.984±0.032        | 0.984±0.012        | 0.656±0.079           | 0.423±0.025           | 0.952±0.006        |
| (p-value vs. ACDC) | (0.0428)           | (0.1271)           | (0.5156)           | (0.5466)           | (0.2785)           | (0.7359)           | (0.4223)           | (0.3376)           | (0.1257)              | ( <b>&lt;0.0001</b> ) | (0.1138)           |
| C5.0               | 0.904±0.022        | 0.764±0.043        | 0.680±0.050        | 0.798±0.095        | 0.426±0.098        | 0.323±0.195        | 0.946±0.029        | <b>1.000±0.000</b> | 0.025±0.079           | 0.992±0.003           | 0.924±0.017        |
| (p-value vs. ACDC) | (0.0118)           | (0.0056)           | (0.0539)           | (0.1077)           | (0.0117)           | (0.1076)           | (0.2006)           | ( <b>0.5349</b> )  | ( <b>&lt;0.0001</b> ) | (0.5689)              | (0.2619)           |

Table 5.7: Mean and standard deviation of ACDC and other chosen models' G-means across chosen real data. Values highlighted in red indicate the best performer of all models for a specific dataset. Values highlighted in blue indicate the best performer out of CCCD variant models. The results of the relevant 5x2 CV  $F$ -test are presented in parenthesis when relevant.

## 5.6 Conclusions and Discussions

In this chapter, we discussed a novel CCCD framework called “FlexiBalls”. FlexiBalls are constructed such that the upper bound of the  $k$ -selected covering balls’ radii are the distance to their closest non-target observation(s). The radius is scaled by  $\tau$  where lower values of  $\tau$  now can change the structure of the digraph to allow a more flexible generalization ability. We present theoretical findings on the VC-dimension of the special case of  $k = 1$  FlexiBall classifiers (denoted as a “single-ball classifier”), and extend it to show that for any  $k$  the VC-dimension is equal to  $2k$ . We provided a theoretical upper bound on the generalization of FlexiBall classifiers. We also show that FlexiBall classifiers find a (probably) improper, pure class cover of  $\mathcal{C}_{\mathcal{X}_j}$  in  $\mathcal{O}(kd(n + m))$  time. The AdaCover digraph classifier (ACDC) acts as the boosted variant of FlexiBall classifiers where in each iteration,  $k$  is stochastically selected such that points that are misclassified have a higher probability of being selected as covering balls.

Monte Carlo simulation studies were performed to assess ACDC performance under both class overlap and class imbalance settings. ACDC performed as well as or better than CCCD and other strong classifier variants in all settings. As class overlap increases, higher values of  $k$  have a more frequent selection rate for each of the evaluation metrics. Class imbalance plays a role in tuning  $\tau$  for ACDC. ACDC was shown to be robust to the curse of dimensionality.

In real data settings, ACDC was only statistically significantly outperformed on the Abalone and Pageblocks datasets. However, it is likely that this is due to specific data intrinsic characteristics of these datasets rather than specific model performance. ACDC was either the top performing classifier, or among the top performing classifiers for all datasets and metrics. Therefore, both synthetic studies and real data experiments imply that ACDC is as effective as other strong models in the classification of imbalanced and overlapping datasets.

ACDC runtimes were faster than P-CCCD, RW-CCCD, MP-CCCD, other ensemble methods, and other SMOTE+ENN methods. The combination of quicker runtimes and performance on noisy, heterogeneous data makes ACDC an appeal alternative to other CCCD and strong classifier variants. Ultimately, we advocate for the use of ACDC over the other compared models for classification tasks, especially when incorrect classification of the minority class is costly.

The conclusion of this chapter finishes our graph-based methods for improving classification performance. In the next chapter, we shift our focus to a new, yet complementary, challenge in statistical learning, reducing data complexity through prototypes. Chapter 6 introduces a novel topological-based prototype selection method and discusses its potential merits.

## Part III

# Topological Prototype Selection

## Chapter 6

### Topological Data Analysis as a Tool for Prototype Selection

The exponential growth of data complexity and dimensionality in modern applications has necessitated the development of novel prototype frameworks. Topological data analysis (TDA) has emerged as a method that leverages the mathematical machinery of algebraic topology to extract robust, coordinate-invariant features from complex dataset topologies [20]. In this chapter, we explore using TDA as a methodology for selecting a prototype set.

#### 6.1 Topological Data Analysis and the Prototype Selection Problem

Despite the rich topological information gained from TDA, it has not been used in prototype selection. Rather, TDA is used primarily as a visualization tool [20], an additional input into another learning model [24, 143], or as a standalone classification strategy [105, 60]. In this chapter, we argue for TDA being a potential solution to the *prototype selection problem*, defined below.

**Definition 6.1.1.** Given a training data  $\mathcal{X} = \{(x_1, y_1), \dots, (x_n, y_n)\}$  where class labels  $y_i \in \mathcal{Y}$  and  $\mathcal{Y} = 1, \dots, K$ , the objective of the prototype selection problem is to find  $\mathcal{S} \subset \mathcal{X}$  that minimizes  $|\mathcal{S}|$  subject to maintaining classification performance on a validation set.

The prototype selection problem arises from several practical considerations such as computational complexity where classification becomes prohibitive for large datasets, storage requirements where storing the entire dataset in memory is infeasible, and / or for noise sensitivity or removing redundancies within the data set [45, 91, 56]. Prototype algorithms are typically classified either by *objectives* or by *approaches*. When classifying by objectives, prototype selection algorithms are seen as either



1. *Condensation Methods.* Condensation methods try to remove redundant class observations. An example of condensation method algorithms is condensed nearest neighbor (CNN) which incrementally builds  $\mathcal{S}$  by adding misclassified examples. However, condensation method performance are not robust to order-dependencies and tend to retain unnecessary prototypes near class centers.
2. *Edition Methods.* Edition methods try to remove noisy or mislabeled points. Approaches like edited nearest neighbor (ENN) construct  $\mathcal{S}$  by removing points from  $\mathcal{X}$  that disagree with their neighborhood majority. While effective for noise removal, these methods often achieve limited reduction rates and suffer from accuracy loss.
3. or *Hybrid Methods.* Hybrid methods combine condensation and edition, which typically achieve better performance but at increased computational cost [92].

Alternatively, prototype selection algorithms are classified by approaches as either,

1. *Heuristic Methods.* These methods are based on simple rules or distance criteria. Both CNN and ENN are examples of heuristic methods.
2. *Optimization Methods.* These methods formulate prototype selection as an optimization problem, such as the selecting prototypes using optimal transport (SPOT) algorithm, which formulates prototype selection as an optimal transport optimization problem. When the optimization problem is explicitly tied to the performance of a downstream model, these methods are called *wrapper methods*. Examples of wrapper methods might be forward or backward selection of prototypes under  $k$ -NN while optimizing AIC.
3. *Clustering-Based Methods.* Cluster-based methods use clustering such as k-means or k-medoids to find representative centroids which then act as a prototype set.

$$\begin{array}{ccccccc}
\emptyset & \longrightarrow & \mathcal{K}_{0,0} & \longrightarrow & \mathcal{K}_{1,0} & \longrightarrow & \mathcal{K}_{2,0} \\
& & \downarrow & & \downarrow & & \downarrow \\
& & \mathcal{K}_{0,1} & \longrightarrow & \mathcal{K}_{1,1} & \longrightarrow & \mathcal{K}_{2,1} \longrightarrow \mathcal{K}
\end{array}$$

Figure 6.1: An example of a commutative diagram for a bifiltration of  $\mathcal{K}$  across two parameters. Each  $\mathcal{K}_{i,j}$  represents a sub-complex.

4. or *Information-Theoretic and Learning-Based Methods*. These strategies are *filtered methods* which select prototypes based on intrinsic properties of the data without explicitly evaluating model performance [90].

The central principle shared among all prototype methods is the *boundary-proximity principle*, which states that patterns near classification boundaries are considered to have higher classification contribution [90, 94]. From a topological viewpoint, classification boundaries can be characterized as *submanifolds* separating different classes. Bifiltrations that capture both geometric and density information provide a promising framework for characterizing these submanifolds. Additionally, since homological features persist across scales, topological based prototype selection enables a more robust prototype based on genuine structural features comparative to other geometric selection strategies, which can be affected by outliers.

## 6.2 Topological Prototype Selector (TPS)

The rest of the chapter is devoted to describing the topological prototype selector (TPS) algorithm and its performance on both synthetic and real data. A bifiltered simplicial complex  $\mathcal{K}$  is built over training data  $\mathcal{X}$  to generate data relationships. Therefore, we can define sub-complex  $\mathcal{K}_{i,j} \subset \mathcal{K}$ , where  $i$  and  $j$  represent indices for respective bifiltration parameters. Figure (6.1) illustrates an example of a bifiltration commutative diagram for  $i = 0, 1, 2$  and  $j = 0, 1$ .

Assume a training data  $\mathcal{X} = \{(x_i, y_i)\}_{i=1}^n$  is training data with class labels  $\mathcal{Y} = 1, \dots, K$ . We define a bifiltered simplicial complex  $\mathcal{K}$  using *radius filtration parameter*  $\epsilon_i$ , and *neighbor filtration parameter*  $\epsilon_j$ . Therefore, both parameters  $\epsilon_i$  and  $\epsilon_j$  correspond to the indices  $i$  and  $j$  of the sub-complex  $\mathcal{K}_{i,j}$  respectively. Ultimately, our goal is to find a meaningful representation of our data using a bifiltration. We briefly discuss the two scale parameters.

Define target class  $y \in \mathcal{Y}$ . Then parameter  $\epsilon_i$  corresponds to the distance scales of the filtration along observations  $\mathcal{X}_y = \{(x_i, y_i) \in \mathcal{X} | y_i = y\}$ . That is the filtration created using the scales  $\{\epsilon_i\}$  shows the representative topology of the underlying target class.

The neighbor scale filtration parameter  $\epsilon_j$  is used to build the filtration across total distances of the target class points to non-target class points. We will limit this filtration to the total distance to  $k$  non-target class points instead of all non-target class points, where  $k$  will be user-defined prior. In this way, topological information of the boundary between target and non-target class(es) are retained. Then the filtration created by  $\{\epsilon_j\}$  is used to indicate points that are further away from the other classes and thus ultimately the decision boundary.

To accomplish our goal of finding a prototype set, we use the union of all  $q$ -simplices' vertex-sets that exist in chosen sub-complex  $\mathcal{K}_{i,j}$ . The new objective is to find and recover a meaningful sub-complex  $\mathcal{K}_{i,j}$ . We discuss recovering the meaningful sub-complex in depth below.

### 6.2.1 Recovering a meaningful sub-complex

Simplicial complex  $\mathcal{K}$  provides a large quantity of multiscale data relationships. Therefore, we would like to choose a sub-complex that best approximates the actual structure of the data and its relation to other classes. Under a single-parameter filtration, each topological feature in a homology group of a given dimension is represented by its *persistence interval*. However, in a bifiltration, it is not always possible to assign clean descriptions

directly to features because the persistence module does not always decompose nicely as in single-parameter filtration [114].

Bifiltrations can be viewed as a 2D grid where each axis represents the different filter parameters [20]. We can therefore slice a bifiltration along lines of a fixed non-negative scale parameter slope to generate a collection of barcodes called *fibered barcodes*. Figure 6.2 provides an example of the 2D grid visualization of a bifiltration over a radius and curve scale parameter, and the resultant sliced sub-complexes for fixed radius and curvature parameters for an example dataset.

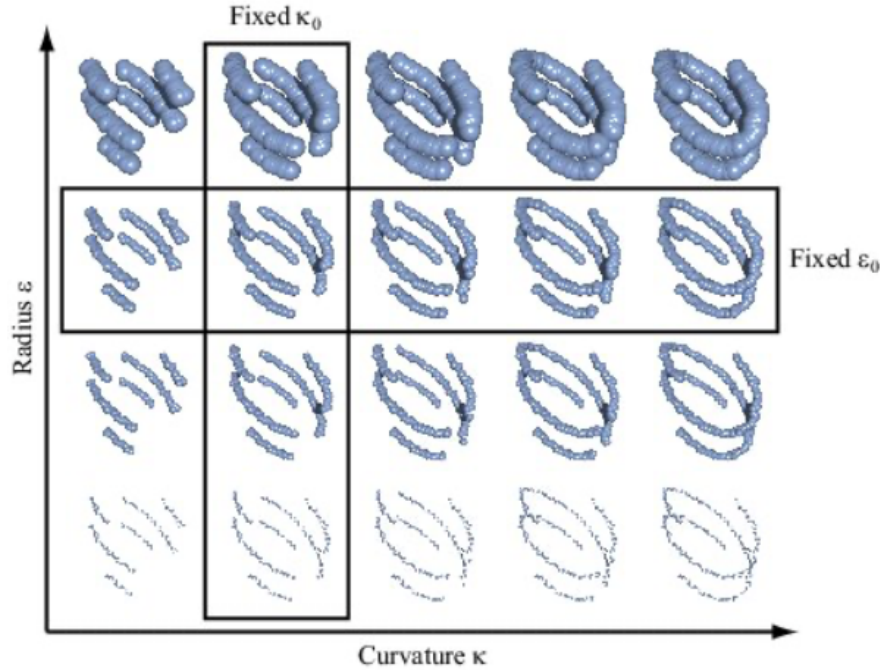


Figure 6.2: Graphical example of representing a bifiltration in a 2D grid. Radius and curvature scale parameters change the underlying sub-complex.

Slicing across one parameter allows recovery of topological information from the opposite parameter’s filtration. Under the boundary-proximity principle, it is less important to identify points further away from the boundary. Thus, we are less interested in how the topology induced by the neighbor filtration changes after a certain threshold.

Therefore, in recovering a meaningful sub-complex, we can slice along the neighbor filtration scale  $\{\epsilon_j\}$  for some fixed  $j$  and recover just the topological changes induced by

the radius filtration scale  $\{\epsilon_i\}$ . Since  $j$  will be fixed, our new objective is the selection of a fixed  $\epsilon_i$ . The meaningful sub-complex  $\mathcal{K}_{i,j}$  will be the simplicial complex at the intersection between the sliced fibered barcode and the selected  $\epsilon_i$ .

Let  $\{D_{i,j}\}$  be the set of all (birth, death) pairs of non-trivial simplicial complexes along fixed slice  $j$ . Persistence intervals with infinite death values are truncated to the maximum value in the collection. Define the  $\text{int}(\cdot)$  function which measures the lifetime of the topological feature associated with each  $d_i \in D_{i,j}$  as

$$\text{int}(\cdot) = \min\{d_i[\text{death}], \max(d_i[\text{death}])\} - d_i[\text{birth}]. \quad (6.1)$$

The collection of all lifetime values across varying  $\epsilon_i$  is denoted as  $\{\text{int}(d_i)\}$  for all  $d_i \in D_{i,j}$ . We can define the average lifetime as

$$\text{avg}(D_{i,j}) = \frac{1}{|\{\text{int}(d_i)\}|} \sum \text{int}(d_i) \quad (6.2)$$

We define

$$\text{AvgInt}(D_{i,j}) = \arg \min_{d_i \in D_{i,j}} (|\text{int}(D_{i,j}) - \text{avg}(D_{i,j})|) \quad (6.3)$$

as the interval lifetime that is closest to the average of all lifetimes. Ties of  $\text{AvgInt}(D_{i,j})$  are settled by using the persistence interval with higher birth. Therefore,  $\epsilon_j$  will be fixed, and  $\epsilon_i$  corresponds to the sub-complex that contains the average interval lifetime, and the meaningful sub-complex  $\mathcal{K}_{i,j}$  is the sub-complex at the intersection of these two filtration scales.

### 6.2.2 Algorithm Details

We present necessary pseudocode for TPS in Algorithm (6). Algorithm (7) is the pseudocode for the bifiltration prototype selection (BPS) algorithm that selects a meaningful sub-complex. For computational efficiency, we fix a maximal dimension  $2 \leq q \ll |\mathcal{X}|$  to control the exponential growth of simplicial complexes. Additionally, we reduce the number

---

**Algorithm 6** Topological Prototype Selector (TPS)

---

**Input:** Dataset  $\mathcal{D}$ , Bifiltration Prototype Selection required parameters  $\theta$

**Output:** Prototype dictionary  $\mathcal{P}_{dict}$

```
 $\mathcal{C} \leftarrow$  unique classes in  $\mathcal{D}$ 
 $\mathcal{P}_{dict} \leftarrow \emptyset$ 
for each class  $c \in \mathcal{C}$  do
     $\mathcal{P}_c \leftarrow \text{BPS}(\mathcal{D}, c, \theta)$ 
     $\mathcal{P}_{dict}[c] \leftarrow \mathcal{P}_c$ 
end for
return  $\mathcal{P}_{dict}$ 
```

---

of simplices while preserving the original persistence information of the complex by applying the edge collapsing method where carefully chosen edges in a complex are replaced with a single vertex. This process reduces the number of vertices, edges, and faces in the structure, leading to simplification without changing persistence information. Full details are in [15]. Implementation of the bifiltration is done using `RipsComplex` in the GUDHI library in Python3 [80]. The algorithm is structured in sequentially, and therefore the runtime is determined by the worst-case complexity of a single persistent homology, which is  $\mathcal{O}(N^{q+2})$  where  $n$  is the number of observations and  $q$  is the maximum simplex dimension [12, 119].

The rationale for using the mean persistence lifetime interval for radius scale selection is the mean is more representative of the overall underlying class topology distribution. However, additional functions for radius persistence selection such as a quantile-based selection, or those discussed in [60], could also be considered instead of the average.

Tuning  $j$  is important to the prototype selection problem, and ultimately determines the cardinality of the resultant prototype set. Currently, TPS fixes  $j$  by corresponding to the lifetime persistence interval closest to a user-defined quantile hyperparameter.

TPS can be performed over any  $H_k(\mathcal{X})$  homology group. We will simplify the notation as  $H_k$ . In the following sections, we provide prototype selection from both simulated and real data settings. Selection in these sections are done over over the  $H_0$  or  $H_1$  homology groups. We acknowledge that for specialized data with high dimensions (such as medical

---

**Algorithm 7** Bifiltration Prototype Selection (BPS) Algorithm

---

**Input:** Class labels  $y_i \in \mathcal{Y} = \{1, \dots, C\}$ , target class  $c \in \mathcal{Y}$ , target class training data  $\mathcal{X} = \{(x_i, y_i)\}_{i=1}^n$ , number of neighbors  $k$ , homology group dimension  $d$ , minimum persistence  $\tau$ , neighbor quantile  $q$

**Output:** Class  $c$  prototype indices  $\mathcal{P}$

```

 $\mathcal{X}_c \leftarrow \{(\mathbf{x}_i, y_i) \in \mathcal{X} : y_i = c\}$  ▷ Target class points
 $\mathcal{X}_{\neg c} \leftarrow \{(\mathbf{x}_i, y_i) \in \mathcal{X} : y_i \neq c\}$  ▷ Non-target class points
for each  $\mathbf{x}_i \in \mathcal{X}_c$  do
     $\mathcal{N}_k(\mathbf{x}_i) \leftarrow k$  nearest neighbors of  $\mathbf{x}_i$  in  $\mathcal{X}_{\neg c}$ 
     $\delta_i \leftarrow \sum_{\mathbf{x}_j \in \mathcal{N}_k(\mathbf{x}_i)} \|\mathbf{x}_i - \mathbf{x}_j\|$ 
end for
Compute pairwise distances  $D = \{d_{ij}\}$  for all points in  $\mathcal{X}_c$ 
 $\{r_{steps}\}, \{n_{steps}\} \leftarrow$  adaptively select filtration step sizes ▷ Optionally, user-defined
 $r_{max} \leftarrow \max(D)$ ,  $\delta_{max} \leftarrow \max(\{\delta_i\})$ 
 $\mathcal{R} \leftarrow \{r_{steps}\} = \{r_1, \dots, r_{r_{max}}\}$ 
 $\mathcal{N} \leftarrow \{n_{steps}\} = \{n_1, \dots, n_{n_{max}}\}$ 
 $\mathcal{H}_r \leftarrow \mathcal{H}_n \leftarrow \emptyset$  ▷ Persistence intervals
for each  $n \in \mathcal{N}$  do
     $\mathcal{V}_n \leftarrow \{\mathbf{x}_i \in \mathcal{X}_c : \delta_i \leq n\}$  ▷ Surviving vertices
    if  $|\mathcal{V}_n| \geq 2$  then
         $\mathcal{K}_n \leftarrow$  Rips complex with radius  $n$  on  $\mathcal{V}_n$ 
         $\mathcal{H}_n \leftarrow \mathcal{H}_n \cup \{\text{intervals from } \mathcal{K}_n \text{ with persistence} > \tau\}$ 
    end if
end for
for each  $r \in \mathcal{R}$  do
     $\mathcal{K}_r \leftarrow$  Rips complex with radius  $r$  on  $\mathcal{X}_c$ 
     $\mathcal{H}_r \leftarrow \mathcal{H}_r \cup \{\text{intervals from } \mathcal{K}_r \text{ with persistence} > \tau\}$ 
end for
 $\bar{p}_r \leftarrow$  Mean persistence of intervals in  $\mathcal{H}_r$ 
 $\bar{f}_r \leftarrow$  Mean filtration value of intervals in  $\mathcal{H}_r$ 
 $p_n^q \leftarrow q$ -quantile of persistence in  $\mathcal{H}_n$ 
 $f_n^q \leftarrow q$ -quantile of filtration values in  $\mathcal{H}_n$ 
 $r^* \leftarrow \arg \min_{r \in \mathcal{R}} |r - \bar{f}_r|$ 
 $n^* \leftarrow \arg \min_{n \in \mathcal{N}} |n - f_n^q|$ 
 $\mathcal{V}_r \leftarrow$  vertices in Rips complex at radius  $r^*$ 
 $\mathcal{V}_n \leftarrow$  vertices with  $\delta_i \leq n^*$ 
 $\mathcal{P} \leftarrow \mathcal{V}_r \cap \mathcal{V}_n$  ▷ Vertices of intersection of bifiltration
return  $\mathcal{P}$ 

```

---

| Synthetic Data                      | Number of Observations | Classes | Class Ratio     | Imbalance | Baseline Accuracy |
|-------------------------------------|------------------------|---------|-----------------|-----------|-------------------|
| Well Separated Blobs                | 600                    | 3       | 200:200:200     | 1.00      | 100.000%          |
| Overlapping Blobs                   | 800                    | 4       | 200:200:200:200 | 1.00      | 86.250%           |
| Overlapping Clusters                | 600                    | 3       | 200:200:200     | 1.00      | 70.556%           |
| Two Moons (moderate noise)          | 500                    | 2       | 250:250         | 1.00      | 100.000%          |
| Two Moons (high noise)              | 500                    | 2       | 250:250         | 1.00      | 87.333%           |
| Concentric Circles (moderate noise) | 500                    | 2       | 250:250         | 1.00      | 96.667%           |
| Concentric Circles (high noise)     | 500                    | 2       | 250:250         | 1.00      | 86.667%           |
| Imbalanced, Overlapping Blobs       | 500                    | 2       | 400:100         | 4.00      | 85.333%           |
| Mixed Multi-Class Blobs             | 800                    | 3       | 400:300:100     | 4.00      | 90.417%           |

Table 6.1: Descriptions of the nine synthetic datasets used for testing prototype selection performance of TPS.

imaging, voids in porous media for material sciences, volumetric physical simulations, etc.) exploration of higher homology groups should be considered.

### 6.3 Performance Evaluation on Synthetic Data

We select nine artificial datasets, each constructed with different underlying distributions and intrinsic data characteristics, to evaluate the reduction performance of TPS. All relevant information on the synthetic datasets is provided in Table (6.1). All synthetic datasets are simulated in  $\mathbb{R}^2$  to allow for visualization of prototype selected. Data was pre-processed by using a 70%/30% training and testing split. A baseline 1-NN model was trained on the entire training split. Next, prototypes were selected from the training data split and a new 1-NN model was trained only on the resultant prototype set for comparison. Both 1-NN models were evaluated on accuracy of classifying the testing data.

Four configurations of TPS are applied to the training data used in the baseline classification model, the  $H_0$  homology group with  $q = 0.10$  and  $q = 0.25$ , and the  $H_1$  homology group with  $q = 0.10$  and  $q = 0.25$ . In this way, we can examine the effects of both homology dimension and neighbor quantile on TPS. We classify the best configuration as the one that achieves the highest accuracy. In cases of tied accuracy between configurations, the highest data reduction percentage is selected. Neighbor filtration parameter  $k$  was set to be equal



| Dataset                                  | Best Config   | Baseline Acc. (%) | # Prototypes | Prototype Ratio | TPS Acc. (%) | Accuracy Diff. (%) | Reduction (%) | TPS Time (s) |
|------------------------------------------|---------------|-------------------|--------------|-----------------|--------------|--------------------|---------------|--------------|
| Well-Separated Blobs (3 Classes)         | $H_1, q=0.10$ | 100.00            | 220          | 68:50:102       | 100.00       | +0.00              | 47.6          | 5.03         |
| Overlap. Blobs (4 Classes)               | $H_1, q=0.25$ | 86.25             | 449          | 107:115:115:112 | 79.58        | -6.67              | 19.8          | 8.77         |
| Overlap. Clusters (3 Classes)            | $H_1, q=0.25$ | 70.56             | 378          | 119:125:134     | 66.11        | -4.44              | 10.0          | 8.13         |
| Two Moons (Moderate Noise)               | $H_0, q=0.10$ | 100.00            | 153          | 86:67           | 100.00       | +0.00              | 56.3          | 7.57         |
| Two Moons (High Noise)                   | $H_0, q=0.25$ | 87.33             | 271          | 135:136         | 87.33        | +0.00              | 22.6          | 8.98         |
| Concentric Circles (Moderate Noise)      | $H_1, q=0.10$ | 96.67             | 217          | 109:108         | 96.67        | +0.00              | 38.0          | 5.62         |
| Concentric Circles (High Noise)          | $H_0, q=0.25$ | 86.67             | 263          | 136:127         | 86.67        | +0.00              | 24.9          | 7.18         |
| Imbalanced Classes (80/20)               | $H_0, q=0.25$ | 85.33             | 283          | 235:48          | 82.67        | -2.67              | 19.1          | 29.63        |
| Mixed Multi-Class (3 Classes, 40/30/10%) | $H_0, q=0.25$ | 90.42             | 421          | 201:170:50      | 90.42        | +0.00              | 24.8          | 37.67        |
| Average                                  | -             | 89.25             | -            | -               | 87.72        | -1.53              | 29.23         | 13.18        |

Table 6.2: TPS performance comparison across the selected synthetic datasets. The best configurations for TPS recorded were the ones that achieved the highest accuracy.

to 10% of the total training dataset. We required a minimum persistence lifetime threshold of  $\tau = 0.01$  for inclusion across all configurations.

Graphical representations of the synthetic datasets and resultant prototypes are provided as figures in the Appendix. Figures (6.3) and (6.4) show the accuracy difference relative to the baseline 1-NN and data reduction percentage across all configurations and datasets respectively. Results are presented in Table (6.2). Under optimal configurations, TPS achieved an average data reduction of 27.7% with an average accuracy decrease of only 1.53%, demonstrating its potential to reduce dataset size without significant loss in classification performance.

Using  $H_0$  consistently results in greater data reduction for a given  $q$  value when compared to  $H_1$  homology groups, with the sole exception occurring in the case of well-separated blobs, where both perform equally. Representing a corresponding 1-dimensional hole in  $H_1$  requires at least three vertices, but often more in practice. Meaning that  $H_1$ -based prototype sets must retain more points by design, thus limiting reduction capabilities [39]. This relationship generalizes to higher-dimensional homology groups. For any  $H_k$ , a minimum of  $k+2$  vertices is required to represent a  $k$ -dimensional hole, therefore, selecting higher  $H_k$  groups will limit data reduction comparatively.

Among the datasets, at least one of the  $H_0$  configurations was selected as the best-performing in two moons (with both moderate and high noise), concentric circles with high

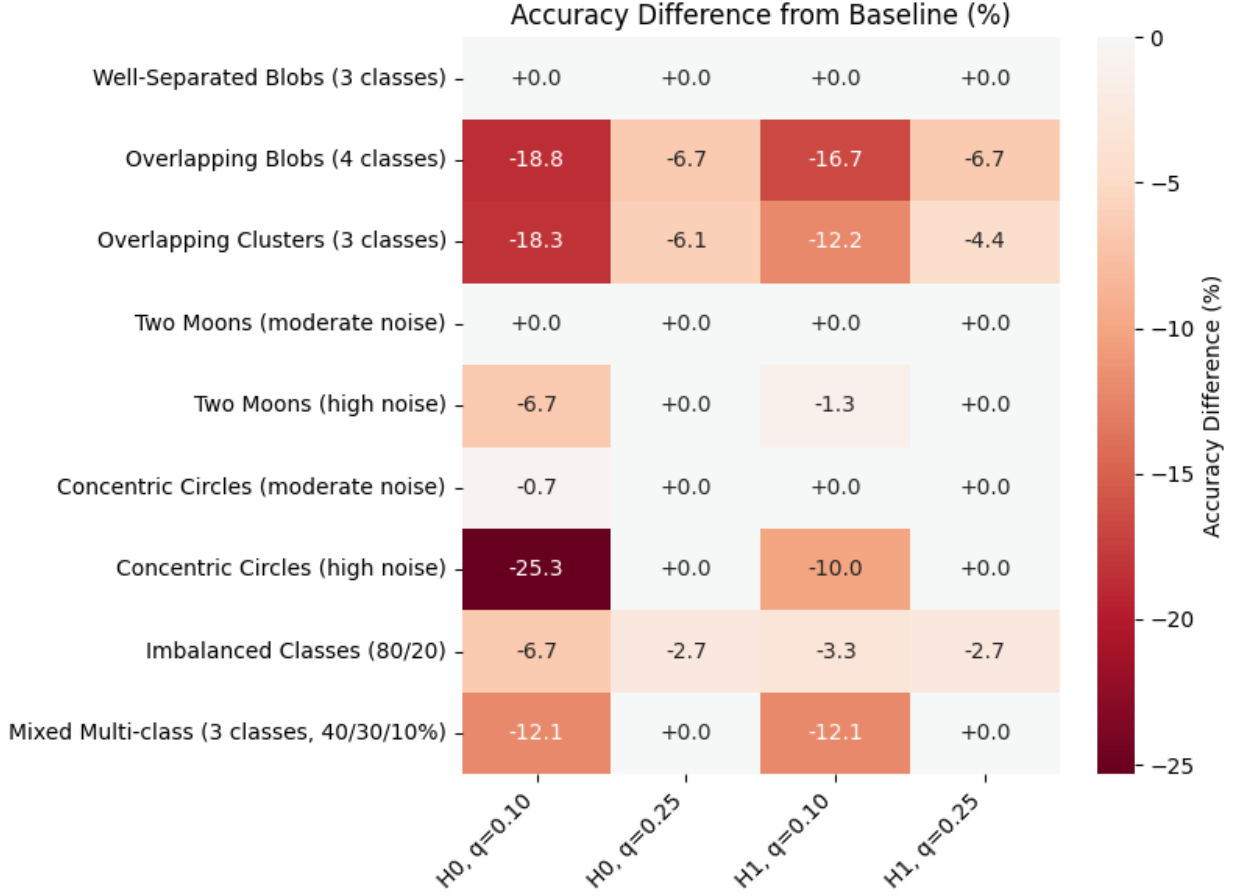


Figure 6.3: Accuracy difference (in percent) of baseline 1-NN against 1-NN after each configuration of TPS is applied for each synthetic dataset.

noise, imbalanced classes, and mixed multi-class datasets. However, in all instances, the classification accuracy of the chosen  $H_0$  configuration matched that of the corresponding  $H_1$  configuration. This suggests that the reduction gain in using  $H_0$  configurations does not necessarily come at the cost of predictive performance.

Comparing  $q = 0.10$  and  $q = 0.25$  under  $H_0$  configurations, we observe that increasing  $q$  leads to a noticeably lower reduction percentage, and thus more prototypes. This is consistent with the role of  $q$ , which reflects the proportion of non-target class points within a neighborhood. A lower  $q$  threshold focuses on regions closest to the decision boundary, and does not include points further away. Geometrically, lower  $q$  values introduce a form

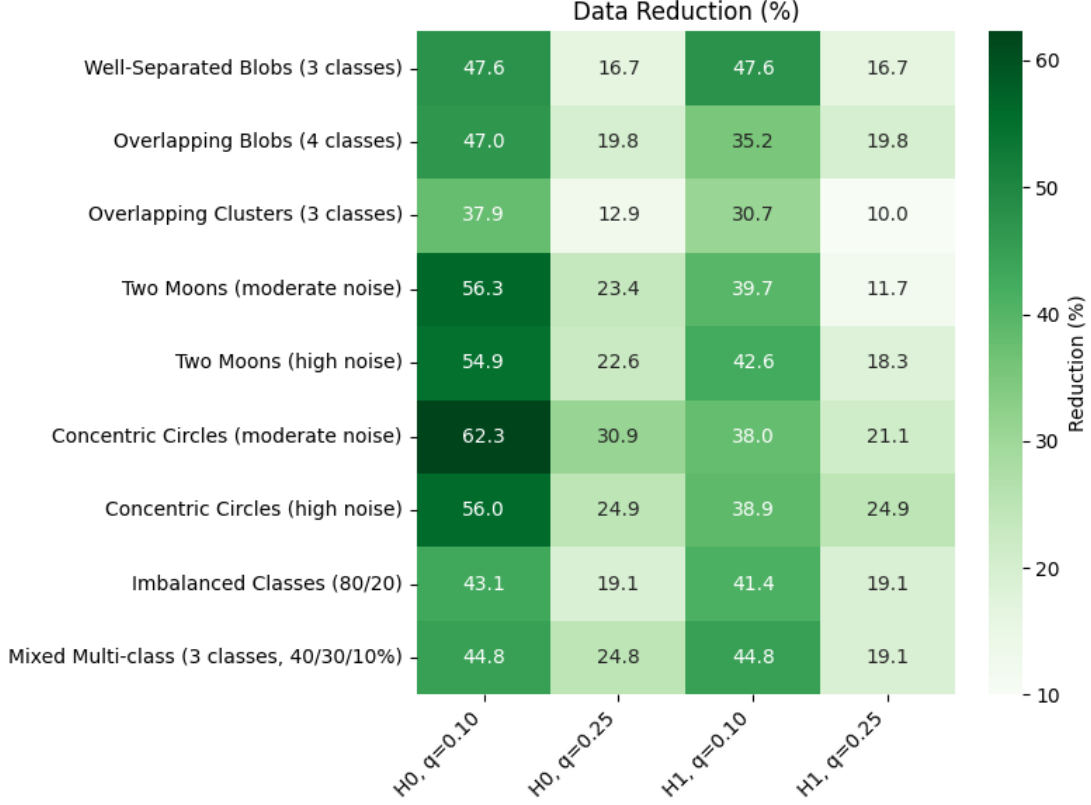


Figure 6.4: Dataset reduction (in percent) training data split after each configuration of TPS is applied for each synthetic dataset.

of regularization that suppresses local fluctuations and topological noise, smoothing the underlying density estimate [106].

For instance, in the concentric circles dataset with high noise, the reduction percentage drops from 56.0% with  $q = 0.10$  to just 24.9% with  $q = 0.25$  under  $H_0$  homology, with a classification accuracy difference of  $-25.30\%$  and  $0.00\%$  respectively. This emphasizes a key practical insight, a higher  $q$  can achieve similar predictive performance to baseline models with more strategically located prototypes, but at the expense of data reduction making tuning  $q$  a data-dependent problem.

Figure (6.5) visually compares the selected prototypes for both  $q$  values under  $H_0$ . While both configurations capture the global structure, the version using  $q = 0.25$  results in a cleaner, more stable decision boundary and less accuracy loss. We note that the accuracy

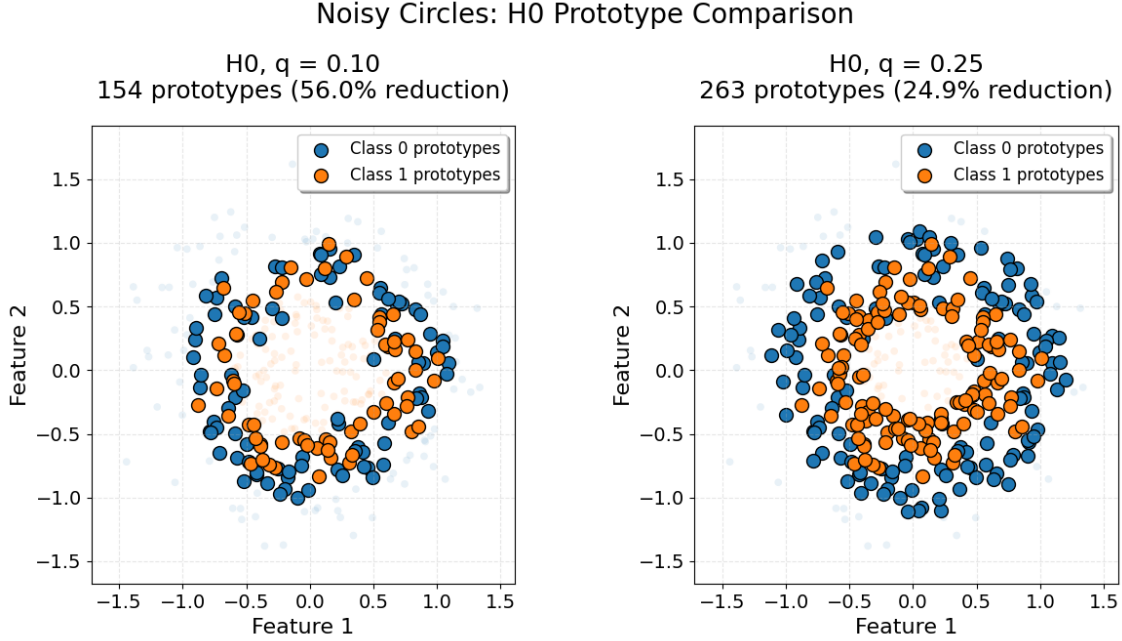


Figure 6.5: Example of prototype selection for  $q = 0.10$  (left) and  $q = 0.25$  (right) under  $H_0$  configurations on the noisy circles synthetic dataset. Lower  $q$  values have a greater reduction percentage.

difference percentage decrease from  $q = 0.10$  to  $q = 0.25$  is lessened under the  $H_1$  homology group with only a 14% difference between reduction, which might indicate that tuning  $q$  becomes less important for higher homology groups.

A key computational bottleneck remains in the calculation of persistent homology, due to the combinatorial nature [12]. This opens a potential criticism to TDA-based prototype selection. For example, on the mixed multi-class dataset with 800 observations, the time required to compute the bifiltration under the best configuration was 37.67 seconds. From a practical standpoint, this underscores that TPS may not yet offer end-to-end computational savings in all contexts, despite ongoing research into more efficient and stable algorithms for computing persistence [30, 137]. Preprocessing data with TPS could be a limiting factor, particularly when TPS is applied before relatively low-cost classification tasks.

In particular, we believe that TPS is better suited for applications focused on dataset compression for storage, memory efficiency, and noise filtration. From a practical standpoint,

| Dataset   | Number of Classes | Number of Observations | Features | Class Distribution                      | Imbalance | Avg. CV | Baseline Acc. |
|-----------|-------------------|------------------------|----------|-----------------------------------------|-----------|---------|---------------|
| Iris      | 3                 | 150                    | 4        | 50:50:50                                | 1.00      | 0.346   | 0.933         |
| Wine      | 3                 | 178                    | 13       | 59:71:48                                | 1.48      | 0.294   | 0.963         |
| Wisconsin | 2                 | 569                    | 30       | 212:357                                 | 1.68      | 0.486   | 0.959         |
| Glass     | 6                 | 214                    | 9        | 70:76:17:13:9:29                        | 8.44      | 0.773   | 0.769         |
| Pima      | 2                 | 768                    | 8        | 500:268                                 | 1.87      | 0.618   | 0.732         |
| BUPA      | 2                 | 345                    | 6        | 145:200                                 | 1.38      | 0.558   | 0.606         |
| Cleveland | 2                 | 303                    | 13       | 164:139                                 | 1.18      | 0.752   | 0.791         |
| Digits    | 10                | 1797                   | 64       | 178:182:177:183:181:182:181:179:174:180 | 1.05      | 4.291   | 0.974         |

Table 6.3: Descriptions of the real datasets used for testing prototype selection performance of TPS.

$H_0$ -based TPS configurations can offer substantial storage savings without sacrificing classification accuracy or underlying data class structure. TPS can still be useful for classification prototype selection if the classifier’s computational complexity outweighs the combinatorial nature of bifiltrations.

## 6.4 Performance Evaluation on Real Data

We benchmark the performance of TPS on eight datasets obtained from the UCI Machine Learning Repository [38]. Detailed characteristics of each dataset, including number of features, classes, and imbalance ratios, are provided in Table (6.3). The selected datasets are: Iris, Wine, Breast Cancer Wisconsin (Wisconsin), Glass Identification (Glass), Pima Indians Diabetes (Pima), Liver Disorders (BUPA), Cleveland Heart Disease (Cleveland), and Optical Recognition of Handwritten Digits (Digits). These datasets were chosen for their diversity in size, number of classes, feature dimensionality, class distribution, and real-world domain representation.

Each dataset was randomly partitioned into a 70%/30% training and testing split. Similar to the synthetic studies, a baseline 1-NN classifier was trained on the entire training split. TPS was performed on the training data and another 1-NN classifier was trained on the resultant prototypes. Both models were evaluated on the testing set and the accuracy performance difference between the two 1-NN models was recorded. For consistency with the synthetic study, we apply the same four TPS configurations. The neighbor filtration

| Dataset        | Best Config     | Baseline Acc. (%) | # Prototypes | Prototype Ratio                         | TPS Acc. (%) | Accuracy Diff. (%) | Reduction (%) | TPS Time (s) |
|----------------|-----------------|-------------------|--------------|-----------------------------------------|--------------|--------------------|---------------|--------------|
| Iris           | $H_0, q = 0.1$  | 93.33             | 57           | 18:21:18                                | 93.33        | +0.00              | 45.7          | 0.08         |
| Wine           | $H_1, q = 0.1$  | 96.30             | 77           | 22:39:16                                | 96.30        | +0.00              | 37.9          | 0.12         |
| Wisconsin      | $H_0, q = 0.25$ | 95.91             | 350          | 109:241                                 | 95.91        | +0.00              | 12.1          | 24.16        |
| Glass          | $H_1, q = 0.25$ | 76.92             | 124          | 37:48:9:7:5:18                          | 72.31        | -4.62              | 16.8          | 0.21         |
| Pima           | $H_0, q = 0.25$ | 73.16             | 489          | 332:157                                 | 72.29        | -0.87              | 8.9           | 68.94        |
| BUPA           | $H_0, q = 0.1$  | 60.58             | 184          | 82:102                                  | 61.54        | +0.96              | 23.7          | 3.49         |
| Cleveland      | $H_0, q = 0.1$  | 79.12             | 143          | 81:62                                   | 76.92        | -2.20              | 32.5          | 1.45         |
| Digits         | $H_0, q = 0.25$ | 97.41             | 1184         | 106:115:122:127:122:112:121:116:120:123 | 97.22        | -0.19              | 5.8           | 17.31        |
| <b>Average</b> | -               | 84.09             | -            | -                                       | 83.23        | -0.86              | 22.9          | 14.47        |

Table 6.4: Performance of TPS across the selected real datasets. The best configurations for TPS recorded were the ones that achieved the highest accuracy.

parameter  $k$  was set to 10% of the size of the training dataset, and a minimum persistence threshold of  $\tau = 0.01$  was required for inclusion in all configurations. Figures (6.6) and (6.7) respectively show the accuracy difference relative to the 1-NN baseline and the data reduction percentage using the TPS-selected prototypes across all configurations and datasets. Again, the best configuration for each dataset is defined as the one achieving the highest classification accuracy, with data reduction percentage used as a tiebreaker. A summary of results is presented in Table (6.4).

Across all real-world datasets, TPS achieved an average accuracy decrease of only 0.86%, while reducing the size of the training data by an average of 22.9%. This highlights the practical utility of TPS in significantly compressing datasets with minimal impact on classification performance. Consistent with trends observed in the synthetic experiments, the  $H_0$  homology group consistently produced greater data reduction compared to  $H_1$  when the filtration threshold  $q$  was held constant. Furthermore, increasing the value of  $q$  within the same homology group led to a noticeable decrease in the reduction percentage, aligning with the interpretation of higher  $q$  values as promoting coarser, more conservative representations that filter out topological noise but retain more points overall.

For the Iris, Wisconsin, Pima, BUPA, Cleveland, and Digits datasets, there was no observed difference in accuracy decrease between the  $H_0$  and  $H_1$  configurations, suggesting

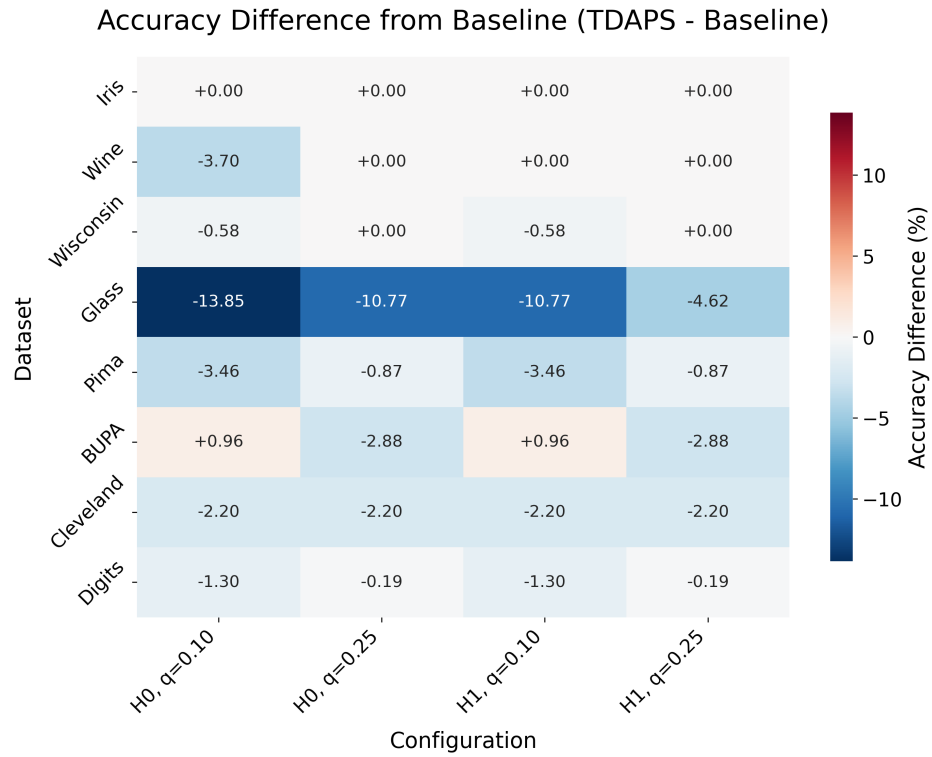


Figure 6.6: Accuracy difference (in percent) of baseline 1-NN against 1-NN after each configuration of TPS is applied for each real dataset.

that homology group choice may have limited impact on predictive performance for certain structured datasets. Notably, in the BUPA dataset, selecting  $q = 0.10$  led to a positive accuracy gain of 0.96% compared to the baseline, whereas increasing  $q$  to 0.25 resulted in a significant accuracy drop of 2.88%. This outcome underscores the importance of tuning  $q$ , as lower thresholds may better eliminate noisy or mislabeled observations in certain cases. The  $H_0$  configuration was selected as optimal in 6 out of the 8 datasets, with the optimal  $q$  value evenly split between 0.10 and 0.25. Each of the four configurations was chosen as the best option for at least one dataset, highlighting the adaptability of TPS to diverse data landscapes, but emphasizing the necessity of empirical validation for parameter selection.

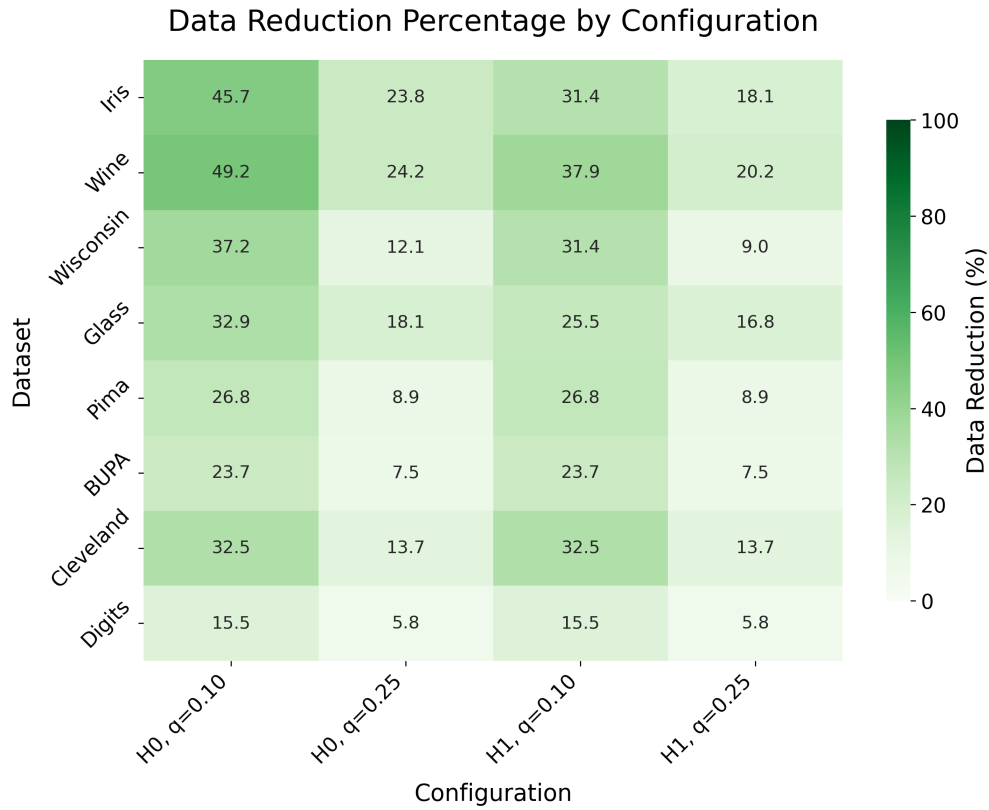


Figure 6.7: Dataset reduction (in percent) training data split after each configuration of TPS is applied for each of the selected real dataset.

Under the best configurations, the Iris, Wine, and Wisconsin datasets showed no accuracy difference between the baseline and TPS-reduced datasets, while achieving training data reductions of 45.7%, 39.9%, and 12.1%, respectively. The Pima and Digits datasets



experienced less than a 1% decrease in accuracy, though both saw under 10% reduction in training data size. The Pima dataset had the highest prototype selection time of 68.94 seconds with the Iris having the lowest at 0.08 seconds. Wisconsin and Digits also had higher computational times of 24.16 and 17.31 seconds respectively while Wine and Glass had times of 0.12 and 0.21 seconds respectively.

For completeness, we compare TPS accuracy and reduction difference against two of the more commonly used prototype selection methods, CNN+ENN and the greedy version of SPOT. The results are presented in Table (6.5). Positive values indicate that TPS performed better, while negative values indicate the compared algorithm performed better. We note that, on average, TPS performed better than CNN+ENN for the accuracy and reduction percentage differences. However, while TPS had a higher average accuracy than SPOT, the reduction percentage difference was much lower for the compared method.

One advantage that TPS has over other prototype selection algorithms is its application across different domains. SPOT is designed for vector data in Euclidean spaces [48]. Applications such as prototype selection of text would require the original domain space to be embedded into Euclidean space and preprocessed prior. The only requirement for TPS is that the space be equipped with a metric, such as cosine similarity for text-analysis [93]. In this way, TPS is inherently more flexible than most other prototype selection methods as it does not require preprocessing the original data before prototype selection.

In this work we compared all classifiers built using accuracy. Given looking at the prototype ratios of synthetic and real datasets, we see that TPS is able to keep much of the corresponding imbalance structure. We believe that TPS should provide prototypes that are effective reductions of original data without fundamentally changing the underlying confusion matrix. We leave the comparison of classifiers after using TPS and other prototype selectors across AUC,  $F_1$ -score, and G-mean as well as across different metrizable spaces as future work.

| Dataset   | TPS vs CNN+ENN |                     | TPS vs SPOTgreedy |                     |
|-----------|----------------|---------------------|-------------------|---------------------|
|           | Acc. Diff (%)  | Reduction Diff. (%) | Acc. Diff (%)     | Reduction Diff. (%) |
| BUPA      | +7.7           | +21                 | +0.0              | -31                 |
| Cleveland | -27.5          | +41                 | -27.5             | -4                  |
| Digits    | -1.9           | -16                 | +5.4              | -47                 |
| Glass     | +4.6           | -9                  | +13.8             | -40                 |
| Iris      | +42.2          | -30                 | +6.7              | -42                 |
| Pima      | -13.0          | +18                 | -6.1              | -25                 |
| Wine      | +7.4           | -6                  | +14.8             | -29                 |
| Wisconsin | -1.8           | +46                 | -0.6              | -22                 |
| Average   | +2.2           | +8                  | +0.8              | -30                 |

Table 6.5: TPS performance comparison against CNN+ENN and SPOT across the selected real datasets.

## 6.5 Conclusions and Discussions

In this chapter, we introduced TPS, a topological prototype selector, and outlined its construction and practical merits. TPS operates in two primary phases

- (1) constructing a bifiltration on the input data, and
- (2) extracting a representative sub-complex whose vertices serve as the prototype set.

The bifiltration combines a radius filtration based on the equipped metric and a neighbor filtration determined by the summed distances to a user-defined number of  $k$  nearest non-target class points. This allows TPS to preserve both intrinsic class topology and inter-class boundary structure when selecting prototypes.

TPS is flexible, being able to be applied to any data embedded in a metric space and necessary computational homology dimensions. While the majority of experiments in this work use  $H_0$  and  $H_1$ , the method extends to higher-dimensional homology groups such as  $H_2$  provided the data exhibits sufficient topological complexity (i.e., voids or cavities).

Synthetic data experiments showed that  $H_0$  lead to greater data reduction comparative to  $H_1$  but often at the expense of worse accuracy. The difference is due to higher homology groups requiring more points to make the higher dimensional holes. The hyperparameter  $q$  acts as a pseudo-geometric regularization on the data set. Lower values of  $q$  selects prototypes

closest to the boundary points, but at the expense of potential accuracy decrease compared to baseline classifiers. We see in real data settings that higher values of  $q$  lead to a significantly more prototypes than using different homology groups.

There does exist a key computational bottleneck in calculating prototype sets with TPS. The high computational cost of persistent homology calculations can potentially outweigh the gains in classification efficiency, particularly when TPS is applied prior to low-complexity classification tasks. We recommend TPS in scenarios where preprocessing overhead is justified, such as tasks emphasizing dataset compression for storage, memory efficiency, or creating prototypes robust to noise. In these settings, TPS excels at distilling robust, structurally meaningful prototypes from noisy or high-dimensional observations, making it especially advantageous for long-term storage or downstream applications requiring compact and interpretable data representations.

## Part IV

# Conclusions

## Chapter 7

### Summary and Discussions

In statistical learning, many modern methods represent data using graphs to capture structure and relationships. Among these, class cover catch digraphs (CCCDs) were originally introduced to address the class cover problem (CCP) and have since been applied to classification and clustering tasks. CCCDs were shown to be efficient classifiers robust to the class imbalance problem, but not necessarily the class overlap problem [99, 76]. This work addresses two distinct, yet complementary, challenges in statistical learning,

- (i) classification performance degradation under class imbalance and class overlap, and
- (ii) reducing data complexity through a novel, principled prototype selection method.

We propose algorithmically modified CCCD variants that improve robustness and generalization in imbalanced and overlapped class settings while preserving the geometric intuition of the original CCCD framework. These contributions enhance the practical utility of CCCD classifiers. In addition, we introduce a topological data analysis (TDA)-based framework for selecting representative subsets (prototypes) from large datasets.

The first of these modifications was incorporation of cardinality-based scoring into the classification decision rule called modified pure class cover catch digraphs (MP-CCCD). Theoretical results showed such MP-CCCD classifiers were consistent under strictly separable class supports. Simulation studies showed that bagged versions of P-CCCD, called BP-CCCD, classifiers performed better than MP-CCCD and other CCCD variants on balanced, overlapping data. Both MP-CCCD and BP-CCCD performed better than other CCCD variants when both class overlap and imbalance was considered, although the differences were practically small. Simulation studies showed that both were particularly effective under

$F_1$ -score evaluation, but added significant computational runtime. Real data experiments showed MP-CCCD had enhanced performance across noisy, heterogeneous data compared to the other CCCD variants for all metrics.

The second modification was a hybrid classifier between SVM and RW-CCCD called support vector class cover catch digraphs (SV-CCCD). In balanced, overlapping data there was no practical significance between SV-CCCD, CCCD variants including MP-CCCD, and other strong state-of-the-art classifiers. SV-CCCD were shown to have runtimes much quicker than MP-CCCD, P-CCCD, RW-CCCD, ensembled classifiers, and other SMOTE+ENN strong classifiers. However, SV-CCCD was characterized by particularly poor performance on real data settings, often having high  $F_1$ -scores but low G-mean. SV-CCCD performance is dependent on the initialization of SVM and characterization of the overlapping regions. Despite its quicker practical runtimes, SV-CCCD should only be used in settings where precision of minority classes is the most important metric for evaluation.

The last algorithmic modification is the construction of a novel family of CCCDs called FlexiBalls. Theoretical results provide a bound on the VC-dimension of P-CCCD and FlexiBalls classifiers respectively. Empirical Rademacher complexity motivates the selection of FlexiBalls as the base learner for a AdaBoosted ensemble learner called AdaCover Digraph Classifier (ACDC). Simulation studies showed improvement over other CCCD and strong classifier variants. In real data settings, ACDC was among the top performers for all datasets across all metrics. ACDC is also characterized by a low practical runtime, making it an appealing alternative to current state-of-the-art classifiers for settings of high class overlap and imbalance. Figure (7.1) details the interaction effects of dimension, overlap shift parameter  $\Delta$ , and imbalance have on the computational runtime for ACDC- $\sqrt{N}$ . Runtimes increase as both dimension and imbalance ratio increase, consistent with theoretical results on the worst case complexity for FlexiBall classifiers.

Additionally, we introduced a topological method for prototype selection called TPS. TPS is applicable to any point cloud in a metric space and supports prototype selection across

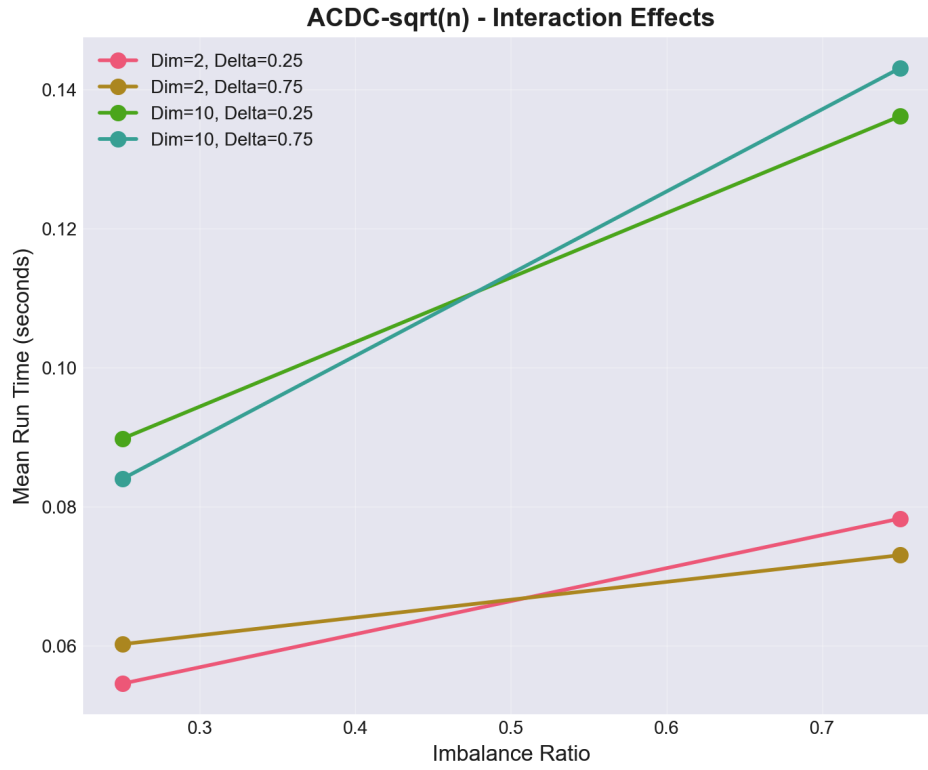


Figure 7.1: Plot of interaction effects for ACDC- $\sqrt{N}$  classifier runtimes across different dimension, overlap shift parameter  $\Delta$ , and imbalance ratio. Increasing dimension greatly increases runtimes.

multiple homology dimensions. Empirical results on both synthetic and real datasets show that TPS achieves substantial data reduction without compromising classification accuracy. TPS consists of two main stages:

- (1) constructing a bifiltration on the input point cloud that simultaneously captures intra-class structure and inter-class boundaries, and
- (2) extracting a representative sub-complex whose vertices define the prototype set.

The bifiltration integrates a standard radius filtration with a neighbor filtration based on the summed distances to the  $k$  nearest non-target class points, thereby preserving both class-intrinsic topology and inter-class boundary structure. We now present areas of future research related to the algorithms detailed throughout this work.

## 7.1 Future Work

We close with a discussion of future research questions that are immediate due to results presented throughout this work. Potential open questions are,

1. For P-CCCDs we believe that both VC dimension bounds presented are not sharp. One potential research direction is the sharpening of both bounds for the VC-dimension of P-CCCD, which would provide immediate generalization bounds and potentially other algorithmic modification avenues.
2. Similar to P-CCCD, RW-CCCD are data-dependent classifiers. Is it possible to provide bounds for the VC-dimension of RW-CCCD? If so, what implications do those bounds have on future improvements to RW-CCCD?
3. Since we were able to incorporate cardinality-based scoring into the P-CCCD classification rule, is it possible to incorporate the cover volume tuning into the RW-CCCD cover construction? Put another way, could variables like  $\tau$  be introduced when constructing RW-CCCD covers to improve their performance?



4. It was shown that the construction of  $\delta$ -separable spaces affects the underlying classification ability of SV-CCCD. Using the largest misclassified points on each class to construct the  $\delta$  separable regions can potentially overfit the hard-overlapping region and lead to poor generalization. To this end, a potential open problem is to formulate the construction of the  $\delta$  separable region as an optimization problem to find an optimal  $\delta$  hyperparameter selection for SV-CCCD.
5. Alternative hybrid-CCCD classification methods that do not rely on tuning regions of designated class overlap could reduce some of the performance problems seen in SV-CCCD. One such example is either the use of class convex hulls or SVDD-based methods. Points within the overlapping region can be classified using RW-CCCD (or ACDC) and points outside the overlapping region are classified using P-CCCD (or MP-CCCD). Another potential advantage is that the overlap region is defined without the need for a user-defined hyperparameter such as  $\delta$ .
6. Our last open problem is the modification of TPS to a prototype classification method, similar to CCCDs. Metric covering balls centered at the barycenter of relevant simplicial complexes can act as a class cover. The radii of these covering balls can be adjusted based on filtration scales, simplicial complex coverings, or adaptively using information of the covering ball.

## Part V

### Appendix & References

## Chapter 8

### Appendix

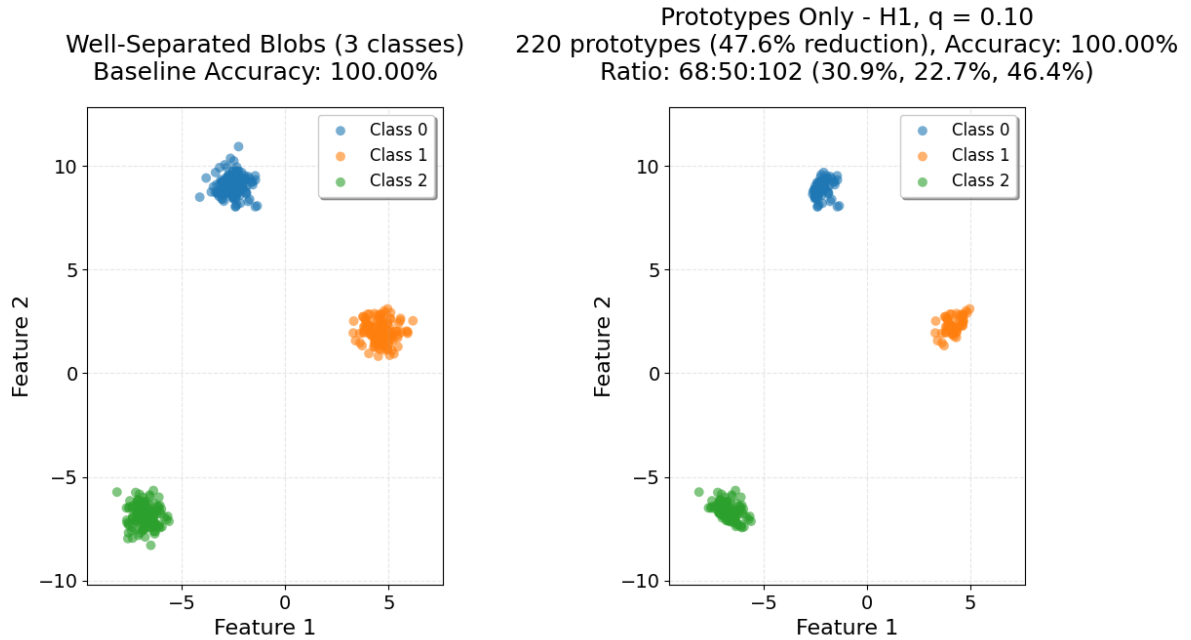


Figure 1: Example of TPS prototype selection on well-separated blobs. The left shows the original dataset and 1-NN accuracy, while the right is the selected prototypes, data reduction percentage, and comparison 1-NN accuracy.

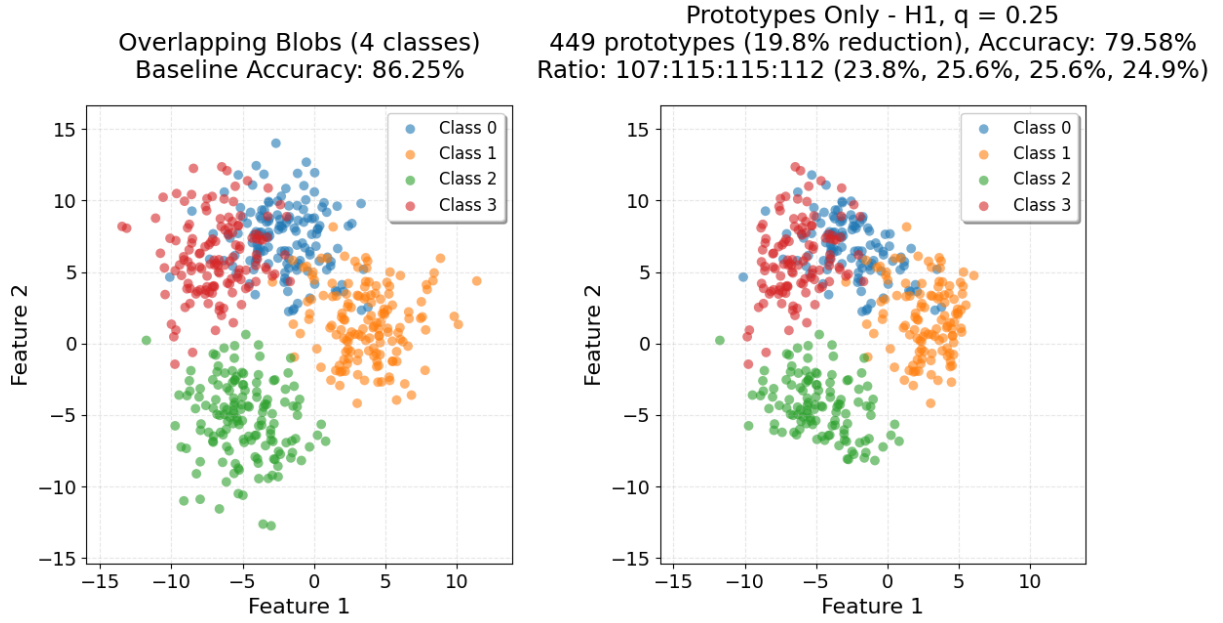


Figure 2: Example of TPS prototype selection on class overlapping blobs. The left shows the original dataset and 1-NN accuracy, while the right is the selected prototypes, data reduction percentage, and comparison 1-NN accuracy.

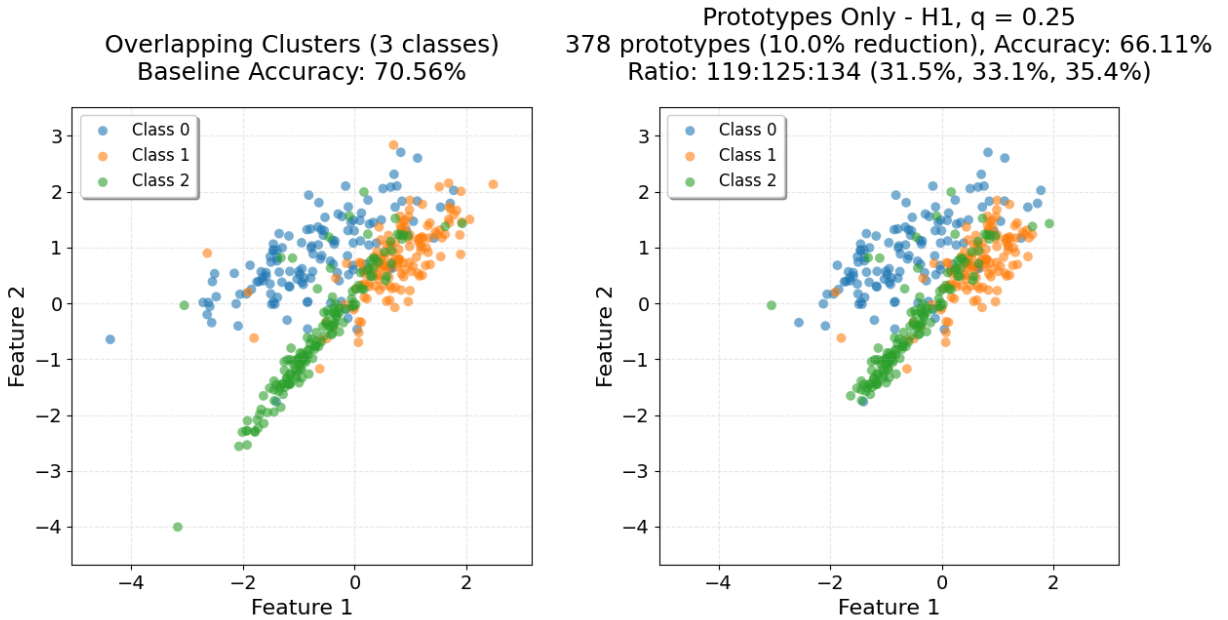


Figure 3: Example of TPS prototype selection on class overlapping clusters. The left shows the original dataset and 1-NN accuracy, while the right is the selected prototypes, data reduction percentage, and comparison 1-NN accuracy.

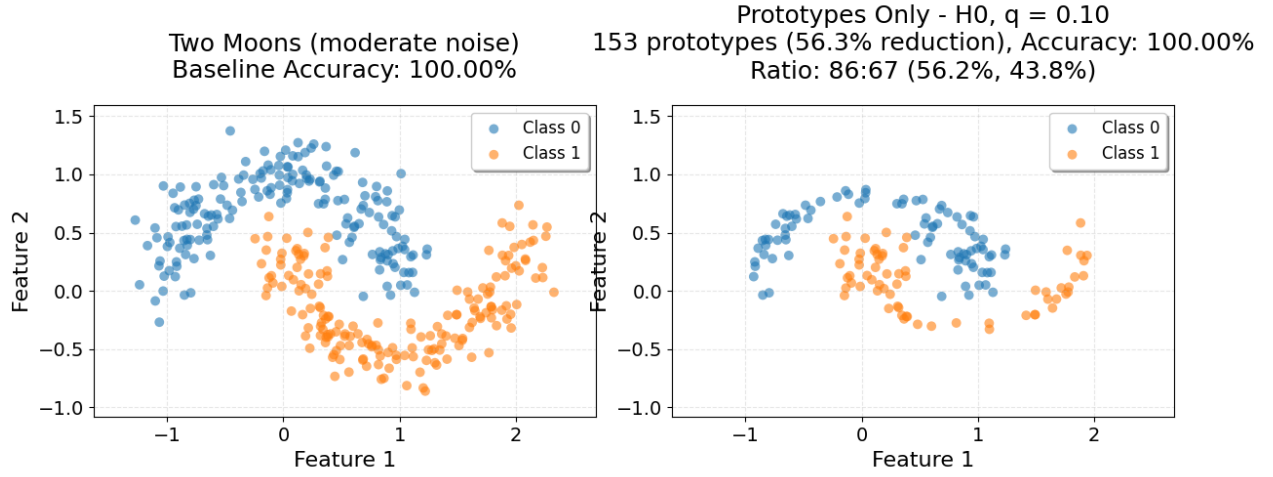


Figure 4: Example of TPS prototype selection on two moons. The left shows the original dataset and 1-NN accuracy, while the right is the selected prototypes, data reduction percentage, and comparison 1-NN accuracy.

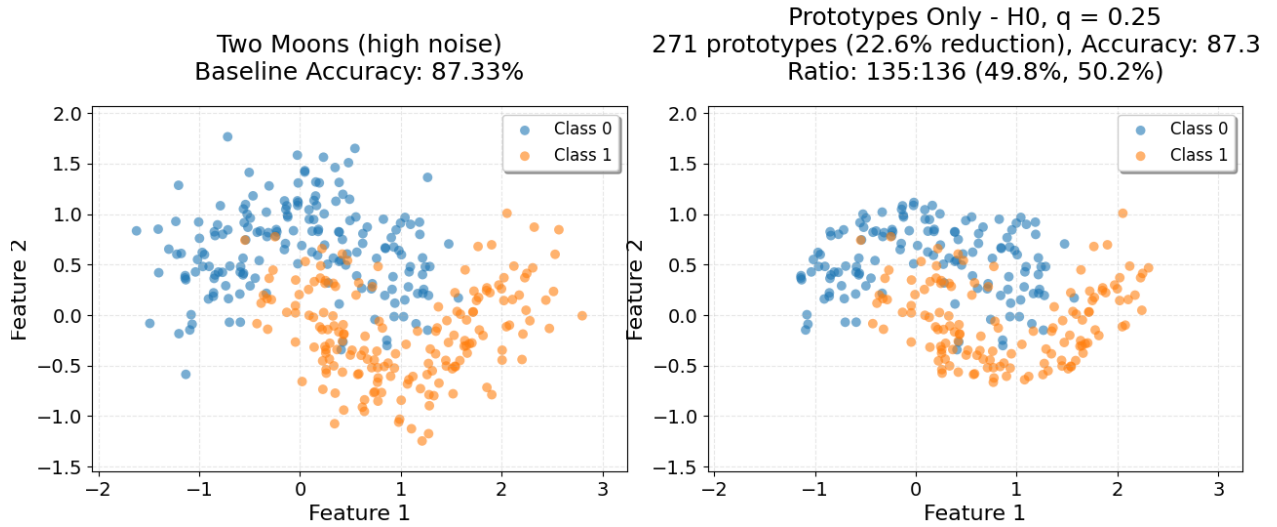


Figure 5: Example of TPS prototype selection on noisy two moons. The left shows the original dataset and 1-NN accuracy, while the right is the selected prototypes, data reduction percentage, and comparison 1-NN accuracy.

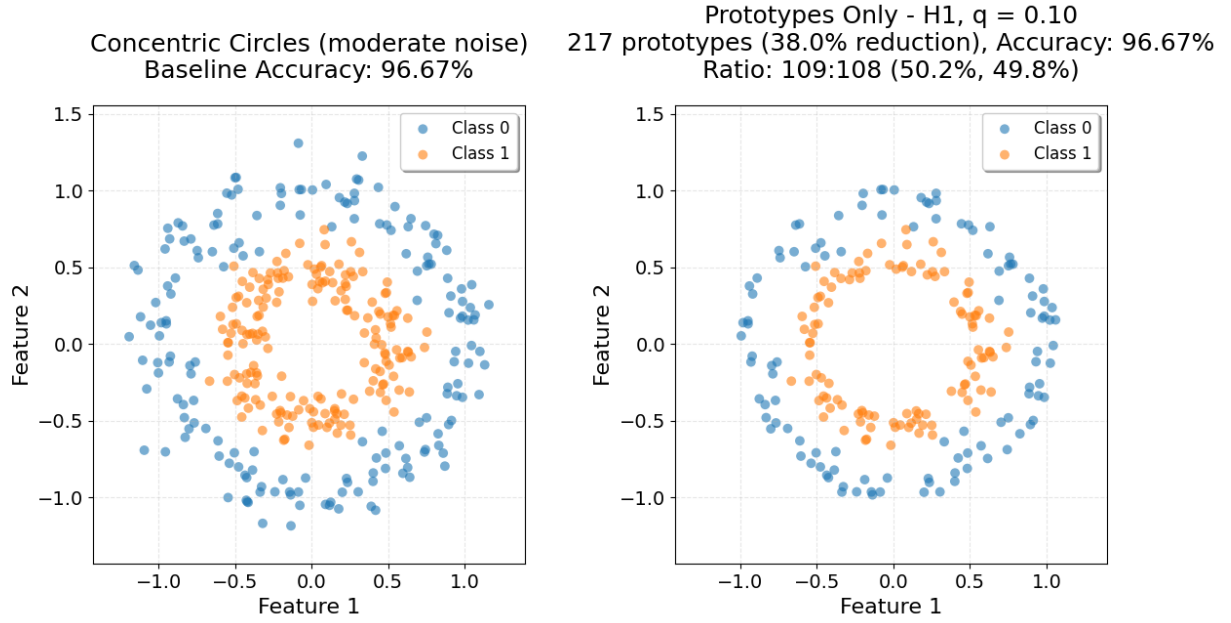


Figure 6: Example of TPS prototype selection on concentric circles. The left shows the original dataset and 1-NN accuracy, while the right is the selected prototypes, data reduction percentage, and comparison 1-NN accuracy.

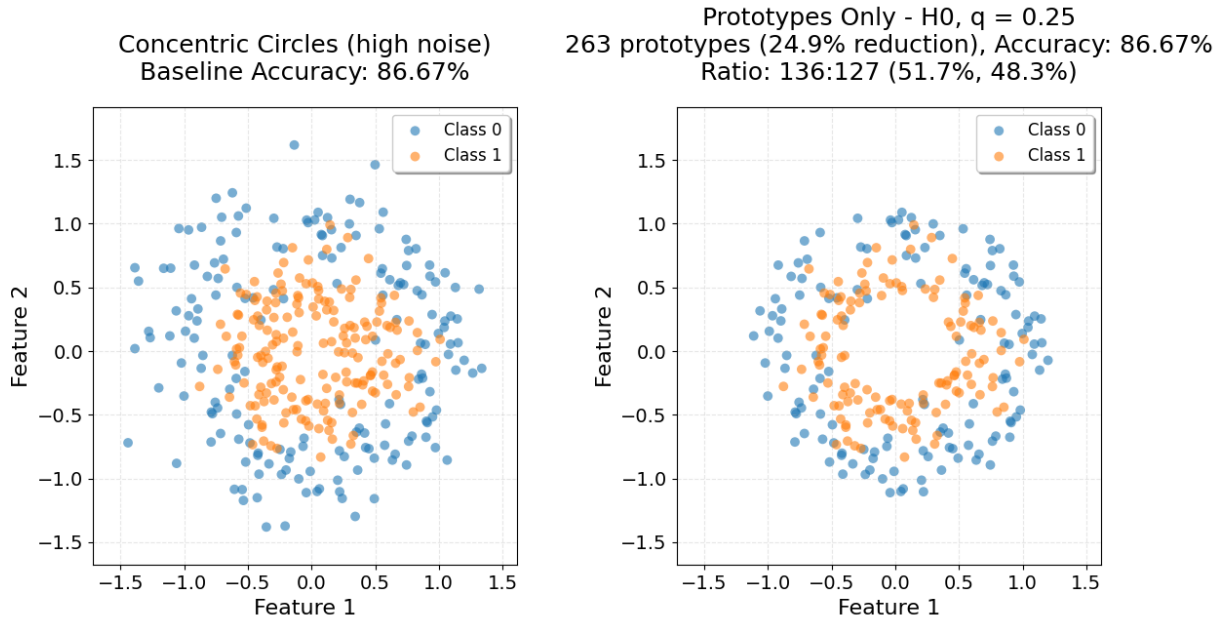


Figure 7: Example of TPS prototype selection on noisy concentric circles. The left shows the original dataset and 1-NN accuracy, while the right is the selected prototypes, data reduction percentage, and comparison 1-NN accuracy.

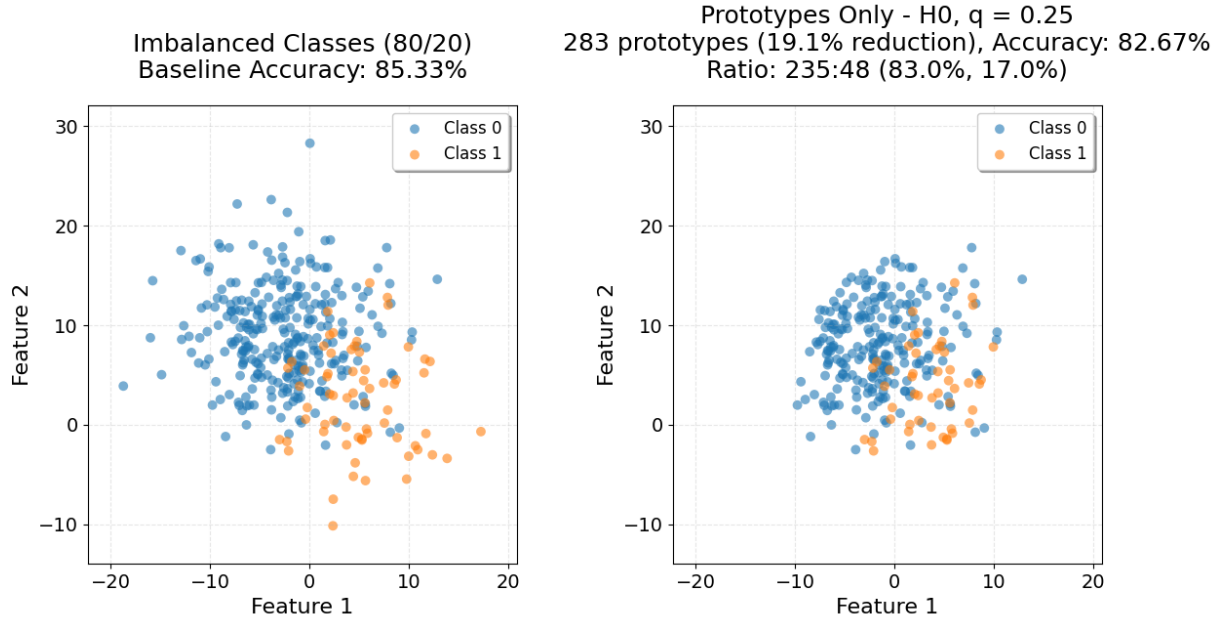


Figure 8: Example of TPS prototype selection on imbalanced classes. The left shows the original dataset and 1-NN accuracy, while the right is the selected prototypes, data reduction percentage, and comparison 1-NN accuracy.

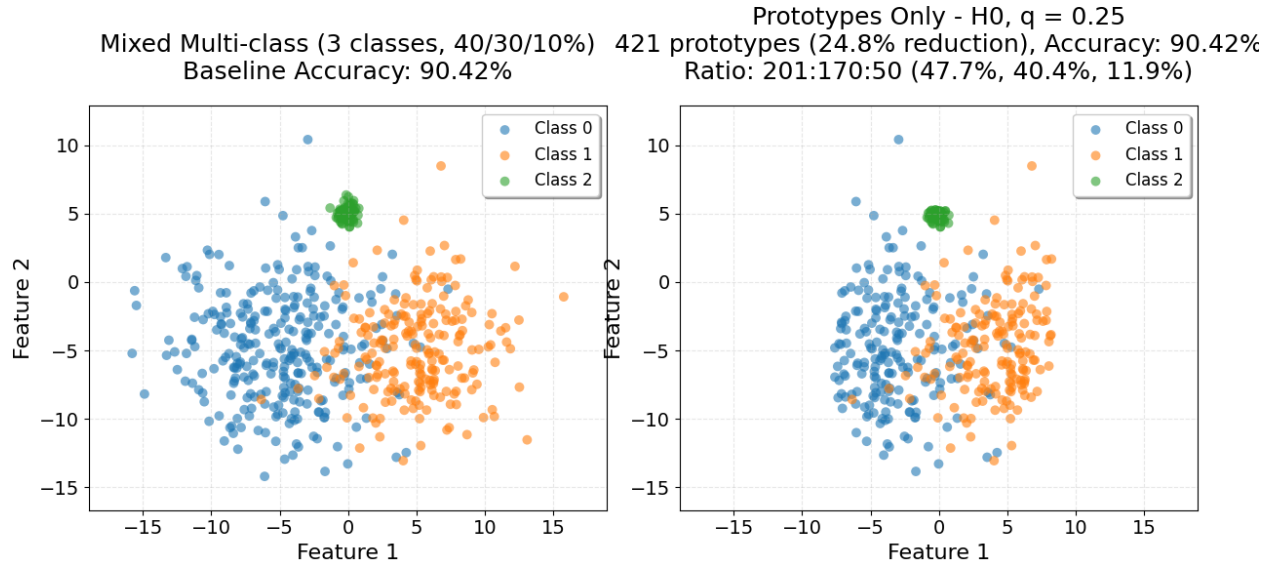


Figure 9: Example of TPS prototype selection on mixed multi-class blobs. The left shows the original dataset and 1-NN accuracy, while the right is the selected prototypes, data reduction percentage, and comparison 1-NN accuracy.

## Bibliography

- [1] Jesus Alcala-Fdez et al. “KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework”. In: *Journal of Multiple-Valued Logic and Soft Computing* 17 (Jan. 2010), pp. 255–287.
- [2] Madjid Allili et al. “Acyclic Partial Matchings for Multidimensional Persistence: Algorithm and Combinatorial Interpretation”. In: *Journal of Mathematical Imaging and Vision* 61.2 (Feb. 2019), pp. 174–192. ISSN: 1573-7683. DOI: 10.1007/s10851-018-0843-8. URL: <https://doi.org/10.1007/s10851-018-0843-8>.
- [3] Ethem Alpaydin. “Combined 5 x 2 cv F Test for Comparing Supervised Classification Learning Algorithms”. In: *Neural computation* 11 8 (1999), pp. 1885–92. URL: <https://api.semanticscholar.org/CorpusID:207731295>.
- [4] Sigurd Assing and Xin Zhi. *Optimal Stopping with Trees: The Details*. eprint: 2210.04645. 2023.
- [5] Mariette Awad and Rahul Khanna. *Efficient learning machines: theories, concepts, and applications for engineers and system designers*. English. Springer nature, 2015. ISBN: 1-4302-5990-6.
- [6] B. Das, N. C. Krishnan, and D. J. Cook. “Handling Class Overlap and Imbalance to Detect Prompt Situations in Smart Homes”. In: *2013 IEEE 13th International Conference on Data Mining Workshops*. Journal Abbreviation: 2013 IEEE 13th International Conference on Data Mining Workshops. Dec. 2013, pp. 266–273. ISBN: 2375-9259. DOI: 10.1109/ICDMW.2013.18.



- [7] Małgorzata Bach, Aleksandra Werner, and Mateusz Palt. “The Proposal of Under-sampling Method for Learning from Imbalanced Datasets”. In: *Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 23rd International Conference KES2019* 159 (Jan. 2019), pp. 125–134. ISSN: 1877-0509. DOI: 10.1016/j.procs.2019.09.167. URL: <https://www.sciencedirect.com/science/article/pii/S1877050919313456>.
- [8] R. Barandela et al. “Strategies for learning in class imbalance problems”. In: *Pattern Recognition* 36.3 (2003), pp. 849–851. ISSN: 0031-3203. DOI: [https://doi.org/10.1016/S0031-3203\(02\)00257-1](https://doi.org/10.1016/S0031-3203(02)00257-1). URL: <https://www.sciencedirect.com/science/article/pii/S0031320302002571>.
- [9] Peter L. Bartlett and Shahar Mendelson. “Rademacher and gaussian complexities: risk bounds and structural results”. In: *J. Mach. Learn. Res.* 3.null (Mar. 2003). Publisher: JMLR.org, pp. 463–482. ISSN: 1532-4435.
- [10] Gustavo E. A. P. A. Batista, Ronaldo C. Prati, and Maria Carolina Monard. “A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data”. In: *SIGKDD Explor. Newsl.* 6.1 (June 2004). Place: New York, NY, USA Publisher: Association for Computing Machinery, pp. 20–29. ISSN: 1931-0145. DOI: 10.1145/1007730.1007735. URL: <https://doi.org/10.1145/1007730.1007735>.
- [11] Rukshan Batuwita and Vasile Palade. “FSVM-CIL: Fuzzy Support Vector Machines for Class Imbalance Learning”. In: *IEEE Transactions on Fuzzy Systems* 18.3 (2010), pp. 558–571. DOI: 10.1109/TFUZZ.2010.2042721.
- [12] Ulrich Bauer. “Ripser: efficient computation of Vietoris–Rips persistence barcodes”. In: *Journal of Applied and Computational Topology* 5.3 (Sept. 2021), pp. 391–423. ISSN: 2367-1734. DOI: 10.1007/s41468-021-00071-5. URL: <https://doi.org/10.1007/s41468-021-00071-5>.

- [13] S. Bereg et al. “The class cover problem with boxes”. en. In: *Computational Geometry* 45.7 (Aug. 2012), pp. 294–304. ISSN: 09257721. DOI: 10.1016/j.comgeo.2012.01.014. URL: <https://linkinghub.elsevier.com/retrieve/pii/S0925772112000375> (visited on 11/14/2024).
- [14] Stephan Bloehdorn and Andreas Hotho. “Boosting for Text Classification with Semantic Features”. In: *Advances in Web Mining and Web Usage Analysis*. Ed. by Bamshad Mobasher et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 149–166. ISBN: 978-3-540-47128-8.
- [15] Jean-Daniel Boissonnat and Siddharth Pritam. “Edge Collapse and Persistence of Flag Complexes”. In: *36th International Symposium on Computational Geometry (SoCG 2020)*. Ed. by Sergio Cabello and Danny Z. Chen. Vol. 164. Leibniz International Proceedings in Informatics (LIPIcs). ISSN: 1868-8969. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020, 19:1–19:15. ISBN: 978-3-95977-143-6. DOI: 10.4230/LIPIcs.SoCG.2020.19. URL: <https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.SoCG.2020.19>.
- [16] Leo Breiman. “Bagging predictors”. In: *Machine Learning* 24.2 (Aug. 1996), pp. 123–140. ISSN: 1573-0565. DOI: 10.1007/BF00058655. URL: <https://doi.org/10.1007/BF00058655>.
- [17] Leo Breiman et al. “Cart”. In: *Classification and regression trees* (1984). Publisher: Wadsworth and Brooks/Cole Monterey, CA, USA.
- [18] Christopher J.C. Burges. “A Tutorial on Support Vector Machines for Pattern Recognition”. In: *Data Mining and Knowledge Discovery* 2.2 (June 1998), pp. 121–167. ISSN: 1573-756X. DOI: 10.1023/A:1009715923555. URL: <https://doi.org/10.1023/A:1009715923555>.
- [19] Adam H. Cannon and Lenore J. Cowen. “Approximation Algorithms for the Class Cover Problem”. In: *Annals of Mathematics and Artificial Intelligence* 40.3 (Mar.

- 2004), pp. 215–223. ISSN: 1573-7470. DOI: 10.1023/B:AMAI.0000012867.03976.a5. URL: <https://doi.org/10.1023/B:AMAI.0000012867.03976.a5>.
- [20] Gunnar Carlsson. “Topology and data”. en. In: *Bulletin of the American Mathematical Society* 46.2 (Jan. 2009), pp. 255–308. ISSN: 0273-0979. DOI: 10.1090/S0273-0979-09-01249-X. URL: <http://www.ams.org/journal-getitem?pii=S0273-0979-09-01249-X> (visited on 06/03/2025).
- [21] Jair Cervantes et al. “A comprehensive survey on support vector machine classification: Applications, challenges and trends”. In: *Neurocomputing* 408 (Sept. 2020), pp. 189–215. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2019.10.118. URL: <https://www.sciencedirect.com/science/article/pii/S0925231220307153>.
- [22] Elvan Ceyhan. “An investigation of proximity catch digraphs in Delaunay tessellations”. PhD Thesis. Baltimore, MD: Johns Hopkins University, 2005. URL: <https://www.proquest.com/openview/b65816de82d760cf17062b97ee0cc55f/1?pq-origsite=gscholar&cbl=18750&diss=y>.
- [23] Nitesh V Chawla et al. “SMOTE: synthetic minority over-sampling technique”. In: *Journal of artificial intelligence research* 16 (2002), pp. 321–357. ISSN: 1076-9757.
- [24] Frédéric Chazal and Bertrand Michel. *An introduction to Topological Data Analysis: fundamental and practical aspects for data scientists*. 2021. URL: <https://arxiv.org/abs/1710.04019>.
- [25] V. Chvatal. “A Greedy Heuristic for the Set-Covering Problem”. In: *Mathematics of Operations Research* 4.3 (1979). Publisher: INFORMS, pp. 233–235. ISSN: 0364765X, 15265471. URL: <http://www.jstor.org/stable/3689577> (visited on 04/26/2023).
- [26] David A. Cieslak and Nitesh V. Chawla. “Learning Decision Trees for Unbalanced Data”. In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by Walter Daelemans, Bart Goethals, and Katharina Morik. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 241–256. ISBN: 978-3-540-87479-9.

- [27] L. Cleofas-Sánchez et al. “Associative learning on imbalanced environments: An empirical study”. In: *Expert Systems with Applications* 54 (July 2016), pp. 387–397. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2015.10.001. URL: <https://www.sciencedirect.com/science/article/pii/S0957417415006880>.
- [28] Vinícius G Costa and Carlos E Pedreira. “Recent advances in decision trees: An updated survey”. In: *Artificial Intelligence Review* 56.5 (2023). Publisher: Springer, pp. 4765–4800. ISSN: 0269-2821.
- [29] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge: Cambridge University Press, 2000. ISBN: 978-0-521-78019-3. DOI: 10.1017/CB09780511801389. URL: <https://www.cambridge.org/core/product/A6A6F4084056A4B23F88648DDBFDD6FC>.
- [30] Matija Čufar and Žiga Virk. “Fast computation of persistent homology representatives with involuted persistent homology”. In: *Foundations of Data Science* 5.4 (2023). Publisher: American Institute of Mathematical Sciences (AIMS), pp. 466–479. ISSN: 2639-8001. DOI: 10.3934/fods.2023006. URL: <http://dx.doi.org/10.3934/fods.2023006>.
- [31] Misha Denil and Thomas Trappenberg. “Overlap versus Imbalance”. In: *Advances in Artificial Intelligence*. Ed. by Atefeh Farzindar and Vlado Kešelj. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 220–231. ISBN: 978-3-642-13059-5.
- [32] Jason G. DeVinney. “The Class Cover Problem and its Application in Pattern Recognition”. PhD Thesis. Baltimore, MD: Johns Hopkins University, 2003. URL: [http://home.ku.edu.tr/pcds/public\\_html/References/JD\\_Defense.pdf](http://home.ku.edu.tr/pcds/public_html/References/JD_Defense.pdf).
- [33] Jason G. DeVinney et al. “Random walks and catch digraphs in classification”. In: *Computing Science and Statistics* 34 (July 2002), pp. 107–117. URL: [http://home.ku.edu.tr/pcds/public\\_html/References/RandWalkDeVinney.pdf](http://home.ku.edu.tr/pcds/public_html/References/RandWalkDeVinney.pdf).

- [34] Luc Devroye, László Györfi, and Gábor Lugosi. *A Probabilistic Theory of Pattern Recognition*. Vol. 31. Stochastic Modelling and Applied Probability. Springer, 1996. ISBN: 978-1-4612-0711-5.
- [35] Luc Devroye, László Györfi, and Gábor Lugosi. “The Bayes Error”. In: *A Probabilistic Theory of Pattern Recognition*. Ed. by Luc Devroye, László Györfi, and Gábor Lugosi. New York, NY: Springer New York, 1996, pp. 9–20. ISBN: 978-1-4612-0711-5. DOI: 10.1007/978-1-4612-0711-5\_2. URL: [https://doi.org/10.1007/978-1-4612-0711-5\\_2](https://doi.org/10.1007/978-1-4612-0711-5_2).
- [36] Tamal Krishna Dey and Yusu Wang. *Computational Topology for Data Analysis*. Cambridge University Press, 2022.
- [37] Pedro Domingos. “Metacost: A general method for making classifiers cost-sensitive”. In: 1999, pp. 155–164.
- [38] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017. URL: <http://archive.ics.uci.edu/ml>.
- [39] Herbert Edelsbrunner and John Harer. *Computational topology: an introduction*. American Mathematical Soc., 2010.
- [40] Bradley Efron and Robert Tibshirani. *An Introduction to the Bootstrap*. English. 1st. New York, NY: Chapman and Hall/CRC, 1994. ISBN: 978-0-429-24659-3. URL: <https://www.taylorfrancis.com/books/mono/10.1201/9780429246593/introduction-bootstrap-bradley-efron-tibshirani>.
- [41] J. P. Egan. *Signal detection theory and ROC analysis*. Series in Cognition and Perception. New York, NY: Academic Press, 1975.
- [42] Charles Elkan. “The Foundations of Cost-Sensitive Learning”. In: *Proceedings of the Seventeenth International Conference on Artificial Intelligence: 4-10 August 2001; Seattle 1* (May 2001).

- [43] Alberto Fernández et al. “Data Intrinsic Characteristics”. In: *Learning from Imbalanced Data Sets*. Ed. by Alberto Fernández et al. Cham: Springer International Publishing, 2018, pp. 253–277. ISBN: 978-3-319-98074-4. DOI: 10.1007/978-3-319-98074-4\_10. URL: [https://doi.org/10.1007/978-3-319-98074-4\\_10](https://doi.org/10.1007/978-3-319-98074-4_10).
- [44] Alberto Fernández et al. “SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary”. In: *J. Artif. Int. Res.* 61.1 (Jan. 2018). Place: El Segundo, CA, USA Publisher: AI Access Foundation, pp. 863–905. ISSN: 1076-9757.
- [45] Salvador Garcia et al. “Prototype Selection for Nearest Neighbor Classification: Taxonomy and Empirical Study”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.3 (2012), pp. 417–435. DOI: 10.1109/TPAMI.2011.142.
- [46] V. García, R. A. Mollineda, and J. S. Sánchez. “On the k-NN performance in a challenging scenario of imbalance and overlapping”. In: *Pattern Analysis and Applications* 11.3 (Sept. 2008), pp. 269–280. ISSN: 1433-755X. DOI: 10.1007/s10044-007-0087-5. URL: <https://doi.org/10.1007/s10044-007-0087-5>.
- [47] Robert Ghrist. “Barcodes: The persistent topology of data”. In: *BULLETIN (New Series) OF THE AMERICAN MATHEMATICAL SOCIETY* 45 (Feb. 2008). DOI: 10.1090/S0273-0979-07-01191-3.
- [48] Karthik S. Gurumoorthy, Pratik Jawanpuria, and Bamdev Mishra. *SPOT: A framework for selection of prototypes using optimal transport*. eprint: 2103.10159. 2021. URL: <https://arxiv.org/abs/2103.10159>.
- [49] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. “Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning”. In: *Advances in Intelligent Computing*. Ed. by De-Shuang Huang, Xiao-Ping Zhang, and Guang-Bin Huang. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 878–887. ISBN: 978-3-540-31902-3.

- [50] T. Hastie, R. Tibshirani, and J.H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer series in statistics. Springer, 2009. ISBN: 978-0-387-84884-6. URL: <https://books.google.com/books?id=eBSgoAEACAAJ>.
- [51] Trevor Hastie et al. “Multi-class adaboost”. In: *Statistics and its Interface* 2.3 (2009). Publisher: International Press of Boston, pp. 349–360.
- [52] Haibo He and E.A. Garcia. “Learning from Imbalanced Data”. In: *Knowledge and Data Engineering, IEEE Transactions on* 21 (Oct. 2009), pp. 1263–1284. DOI: 10.1109/TKDE.2008.239.
- [53] Kurt Hornik, Christian Buchta, and Achim Zeileis. “Open-Source Machine Learning: R Meets Weka”. In: *Computational Statistics* 24.2 (2009), pp. 225–232. DOI: 10.1007/s00180-008-0119-7.
- [54] Jin Huang and C.X. Ling. “Using AUC and accuracy in evaluating learning algorithms”. In: *IEEE Transactions on Knowledge and Data Engineering* 17.3 (2005), pp. 299–310. DOI: 10.1109/TKDE.2005.50.
- [55] Abiodun M. Ikotun et al. “K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data”. In: *Information Sciences* 622 (Apr. 2023), pp. 178–210. ISSN: 0020-0255. DOI: 10.1016/j.ins.2022.11.139. URL: <https://www.sciencedirect.com/science/article/pii/S0020025522014633>.
- [56] Jacob Bien and Robert Tibshirani. “Prototype selection for interpretable classification”. In: *The Annals of Applied Statistics* 5.4 (Dec. 2011), pp. 2403–2424. DOI: 10.1214/11-A0AS495. URL: <https://doi.org/10.1214/11-A0AS495>.
- [57] Nathalie Japkowicz and Shaju Stephen. “The Class Imbalance Problem: A Systematic Study”. In: *Intell. Data Anal.* 6.5 (Oct. 2002). Place: NLD Publisher: IOS Press, pp. 429–449. ISSN: 1088-467X.

- [58] Thorsten Joachims. *Making large-scale SVM learning practical*. Tech. rep. Technical report, 1998.
- [59] Shehroz S. Khan and Michael G. Madden. “A Survey of Recent Trends in One Class Classification”. In: *Artificial Intelligence and Cognitive Science*. Ed. by Lorcan Coyle and Jill Freyne. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 188–197. ISBN: 978-3-642-17080-5.
- [60] Rolando Kindelan et al. “A topological data analysis based classifier”. en. In: *Advances in Data Analysis and Classification* 18.2 (June 2024), pp. 493–538. ISSN: 1862-5347, 1862-5355. DOI: 10.1007/s11634-023-00548-4. URL: <https://link.springer.com/10.1007/s11634-023-00548-4> (visited on 06/03/2025).
- [61] S. B. Kotsianti and D. Kanellopoulos. “Combining Bagging, Boosting and Dagging for Classification Problems”. In: *Knowledge-Based Intelligent Information and Engineering Systems*. Ed. by Bruno Apolloni, Robert J. Howlett, and Lakhmi Jain. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 493–500. ISBN: 978-3-540-74827-4.
- [62] Max Kuhn and Kjell Johnson. “Classification Trees and Rule-Based Models”. In: *Applied Predictive Modeling*. Ed. by Max Kuhn and Kjell Johnson. New York, NY: Springer New York, 2013, pp. 369–413. ISBN: 978-1-4614-6849-3. DOI: 10.1007/978-1-4614-6849-3\_14. URL: [https://doi.org/10.1007/978-1-4614-6849-3\\_14](https://doi.org/10.1007/978-1-4614-6849-3_14).
- [63] Sanjeev R. Kulkarni, Gabor Lugosi, and Santosh S. Venkatesh. “Learning pattern classification-A survey”. English (US). In: *IEEE Transactions on Information Theory* 44.6 (1998). Publisher: Institute of Electrical and Electronics Engineers Inc., pp. 2178–2206. ISSN: 0018-9448. DOI: 10.1109/18.720536.
- [64] Anil Kumar, Dinesh Singh, and Rama Shankar Yadav. “Class overlap handling methods in imbalanced domain: A comprehensive survey.” eng. In: *Multimedia Tools & Applications* 83.23 (July 2024). Publisher: Springer Nature, pp. 63243–63290. ISSN:



- 1380-7501. DOI: 10.1007/s11042-023-17864-8. URL: <https://research.ebsco.com/linkprocessor/plink?id=39d0fbcc-8a40-3276-9483-6e561d47c0e3>.
- [65] Ludmila I. Kuncheva et al. "Instance selection improves geometric mean accuracy: a study on imbalanced data classification". In: *Progress in Artificial Intelligence* 8.2 (June 2019), pp. 215–228. ISSN: 2192-6360. DOI: 10.1007/s13748-019-00172-4. URL: <https://doi.org/10.1007/s13748-019-00172-4>.
- [66] Daniel T. Larose and Chantal D. Larose. "k Nearest Neighbor Algorithm". In: *Discovering Knowledge in Data: An Introduction to Data Mining*. 2014, pp. 149–164. DOI: 10.1002/9781118874059.ch7.
- [67] Han Kyu Lee and Seoung Bum Kim. "An overlap-sensitive margin classifier for imbalanced and overlapping data". In: *Expert Systems with Applications* 98 (May 2018), pp. 72–83. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2018.01.008. URL: <https://www.sciencedirect.com/science/article/pii/S0957417418300071>.
- [68] Zachary Chase Lipton, Charles Elkan, and Balakrishnan Narayanaswamy. *Thresholding Classifiers to Maximize F1 Score*. eprint: 1402.1892. 2014. URL: <https://arxiv.org/abs/1402.1892>.
- [69] Cheng-Lin Liu. "Partial discriminative training for classification of overlapping classes in document analysis". In: *IJDAR* 11 (Nov. 2008), pp. 53–65. DOI: 10.1007/s10032-008-0069-1.
- [70] Wei Liu, Yoshihisa Suzuki, and Shuyi Du. "Ensemble learning algorithms based on easyensemble sampling for financial distress prediction". In: *Annals of Operations Research* 346.3 (Mar. 2025), pp. 2141–2172. ISSN: 1572-9338. DOI: 10.1007/s10479-025-06494-y. URL: <https://doi.org/10.1007/s10479-025-06494-y>.
- [71] Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. "Exploratory Undersampling for Class-Imbalance Learning". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 39.2 (2009), pp. 539–550. DOI: 10.1109/TSMCB.2008.2007853.

- [72] Victoria López et al. “An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics”. In: *Information sciences* 250 (2013). Publisher: Elsevier, pp. 113–141. ISSN: 0020-0255.
- [73] Ana C. Lorena et al. *How Complex is your classification problem? A survey on measuring classification complexity*. eprint: 1808.03591. 2020. URL: <https://arxiv.org/abs/1808.03591>.
- [74] Ulrike von Luxburg and Bernhard Schölkopf. “Statistical Learning Theory: Models, Concepts, and Results”. In: *Inductive Logic*. Ed. by Dov M. Gabbay, Stephan Hartmann, and John Woods. Vol. 10. Handbook of the History of Logic. ISSN: 1874-5857. North-Holland, 2011, pp. 651–706. DOI: <https://doi.org/10.1016/B978-0-444-52936-7.50016-1>. URL: <https://www.sciencedirect.com/science/article/pii/B9780444529367500161>.
- [75] Hossin M and Sulaiman M.N. “A Review On Evaluation Metrics For Data Classification Evaluations”. In: *International Journal of Data Mining & Knowledge Management Process* 5 (2015), pp. 01–11. URL: <https://api.semanticscholar.org/CorpusID:61877559>.
- [76] Artür Manukyan and Elvan Ceyhan. “Classification of Imbalanced Data with a Geometric Digraph Family”. In: *The Journal of Machine Learning Research* 17.1 (Oct. 2019), pp. 6504–6543. DOI: <https://doi.org/10.48550/arXiv.1904.04564>. URL: <https://www.jmlr.org/papers/volume17/15-604/15-604.pdf>.
- [77] Artür Manukyan and Elvan Ceyhan. “Classification using proximity catch digraphs”. In: *Machine Learning* 109.4 (Apr. 2020), pp. 761–811. ISSN: 1573-0565. DOI: 10.1007/s10994-020-05878-4. URL: <https://doi.org/10.1007/s10994-020-05878-4>.
- [78] David Marchette. “Class cover catch digraphs”. In: *WIREs Computational Statistics* 2.2 (Mar. 2010). Publisher: John Wiley & Sons, Ltd, pp. 171–177. ISSN: 1939-5108.

- DOI: 10.1002/wics.70. URL: <https://doi.org/10.1002/wics.70> (visited on 05/10/2023).
- [79] David J. Marchette. *cccd: Class Cover Catch Digraphs*. 2022. URL: <https://CRAN.R-project.org/package=cccd>.
  - [80] Clément Maria et al. “Rips complex”. In: *GUDHI User and Reference Manual*. 3.11.0. GUDHI Editorial Board, 2025. URL: [https://gudhi.inria.fr/doc/3.11.0/group\\_\\_rips\\_\\_complex.html](https://gudhi.inria.fr/doc/3.11.0/group__rips__complex.html).
  - [81] Maciej A. Mazurowski et al. “Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance”. In: *Advances in Neural Networks Research: IJCNN '07* 21.2 (Mar. 2008), pp. 427–436. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2007.12.031. URL: <https://www.sciencedirect.com/science/article/pii/S0893608007002407>.
  - [82] David Meyer et al. *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*. 2023. URL: <https://CRAN.R-project.org/package=e1071>.
  - [83] Satwik Mishra. “Handling imbalanced data: SMOTE vs. random undersampling”. In: *Int. Res. J. Eng. Technol* 4.8 (2017), pp. 317–320.
  - [84] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012. ISBN: 0-262-01825-X.
  - [85] James N Morgan and John A Sonquist. “Problems in the analysis of survey data, and a proposal”. In: *Journal of the American statistical association* 58.302 (1963). Publisher: Taylor & Francis, pp. 415–434. ISSN: 0162-1459.
  - [86] Mary M Moya, Mark W Koch, and Larry D Hostetler. “One-class classifier networks for target recognition applications”. In: *NASA STI/Recon Technical Report N 93* (1993), p. 24043.

- [87] James R. Munkres. *Topology*. 2nd ed. Published: Hardcover. Prentice Hall, Inc., Jan. 2000. ISBN: 0-13-181629-2. URL: <http://www.worldcat.org/isbn/0131816292>.
- [88] N. Thai-Nghe, Z. Gantner, and L. Schmidt-Thieme. “Cost-sensitive learning methods for imbalanced data”. In: *The 2010 International Joint Conference on Neural Networks (IJCNN)*. Journal Abbreviation: The 2010 International Joint Conference on Neural Networks (IJCNN). July 2010, pp. 1–8. ISBN: 2161-4407. DOI: 10.1109/IJCNN.2010.5596486.
- [89] Sihem Nouas, Lamia Oukid, and Fatima Boumahdi. “Enhancing imbalanced text classification: an overlap-based refinement approach”. In: *Data Science and Management* (2025). ISSN: 2666-7649. DOI: <https://doi.org/10.1016/j.dsm.2025.03.001>. URL: <https://www.sciencedirect.com/science/article/pii/S2666764925000177>.
- [90] José Arturo Olvera López, Jesús Ariel Carrasco Ochoa, and José Francisco Martínez Trinidad. “Prototype selection methods”. In: *Computación y Sistemas* 13.4 (2010). Publisher: Instituto Politécnico Nacional, Centro de Investigación en Computación, pp. 449–462.
- [91] J. Arturo Olvera-López, J. Ariel Carrasco-Ochoa, and J. Fco. Martínez-Trinidad. “Prototype Selection Via Prototype Relevance”. en. In: *Progress in Pattern Recognition, Image Analysis and Applications*. Ed. by José Ruiz-Shulcloper and Walter G. Kropatsch. Vol. 5197. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 153–160. ISBN: 978-3-540-85919-2 978-3-540-85920-8. DOI: 10.1007/978-3-540-85920-8\_19. URL: [http://link.springer.com/10.1007/978-3-540-85920-8\\_19](http://link.springer.com/10.1007/978-3-540-85920-8_19) (visited on 06/03/2025).
- [92] J. Arturo Olvera-López et al. “A review of instance selection methods”. In: *Artificial Intelligence Review* 34.2 (Aug. 2010), pp. 133–143. ISSN: 1573-7462. DOI: 10.1007/s10462-010-9165-y. URL: <https://doi.org/10.1007/s10462-010-9165-y>.

- [93] Kwangil Park, June Seok Hong, and Wooju Kim. “A methodology combining cosine similarity with classifier for text classification”. In: *Applied Artificial Intelligence* 34.5 (2020). Publisher: Taylor & Francis, pp. 396–411.
- [94] Elżbieta Pekalska, Robert P.W. Duin, and Pavel Paclík. “Prototype selection for dissimilarity-based classifiers”. In: *Part Special Issue: Complexity Reduction* 39.2 (Feb. 2006), pp. 189–208. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2005.06.012. URL: <https://www.sciencedirect.com/science/article/pii/S0031320305002633>.
- [95] Barbara Pes and Giuseppina Lai. “Cost-sensitive learning strategies for high-dimensional and imbalanced data: a comparative study”. In: *PeerJ Computer Science* 7 (Dec. 2021). Ed. by Davide Chicco, e832. ISSN: 2376-5992. DOI: 10.7717/peerj-cs.832. URL: <https://doi.org/10.7717/peerj-cs.832>.
- [96] Clifton Phua, Daminda Alahakoon, and Vincent Lee. “Minority report in fraud detection: classification of skewed data”. In: *SIGKDD Explor. Newsl.* 6.1 (June 2004). Place: New York, NY, USA Publisher: Association for Computing Machinery, pp. 50–59. ISSN: 1931-0145. DOI: 10.1145/1007730.1007738. URL: <https://doi.org/10.1145/1007730.1007738>.
- [97] Ronaldo Prati, Gustavo Batista, and Maria-Carolina Monard. *Data mining with imbalanced class distributions: Concepts and methods*. Journal Abbreviation: Paper presented at the IICAI Pages: 376 Publication Title: Paper presented at the IICAI. Jan. 2009.
- [98] Carey E. Priebe, Jason G. DeVinney, and David J. Marchette. “On the distribution of the domination number for random class cover catch digraphs”. In: *Statistics & Probability Letters* 55.3 (Dec. 2001), pp. 239–246. ISSN: 0167-7152. DOI: 10.1016/S0167-7152(01)00129-8. URL: <https://www.sciencedirect.com/science/article/pii/S0167715201001298>.

- [99] Carey E. Priebe et al. “Class cover catch digraphs for latent class discovery in gene expression monitoring by DNA microarrays”. In: *Computational Statistics & Data Analysis* 43.4 (2003), pp. 621–632. URL: <https://EconPapers.repec.org/RePEc:eee:csdana:v:43:y:2003:i:4:p:621-632>.
- [100] Carey E. Priebe et al. “Classification Using Class Cover Catch Digraphs”. In: *Journal of Classification* 20.1 (May 2003), pp. 003–023. ISSN: 1432-1343. DOI: 10.1007/s00357-003-0003-7. URL: <https://doi.org/10.1007/s00357-003-0003-7>.
- [101] Mini Prince and PM Prathap. “An Imbalanced Dataset and Class Overlapping Classification Model for Big Data.” In: *Computer Systems Science & Engineering* 44.2 (2023).
- [102] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014. ISBN: 0-08-050058-7.
- [103] R. Quinlan. *C5.0*. 2004. URL: [www.rulequest.com](http://www.rulequest.com).
- [104] R Core Team. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing, 2023. URL: <https://www.R-project.org/>.
- [105] Henri Riihimäki et al. “A topological data analysis based classification method for multiple measurements”. en. In: *BMC Bioinformatics* 21.1 (Dec. 2020), p. 336. ISSN: 1471-2105. DOI: 10.1186/s12859-020-03659-3. URL: <https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-020-03659-3> (visited on 06/03/2025).
- [106] Alessandro Rinaldo and Larry Wasserman. “Generalized density clustering”. In: *The Annals of Statistics* 38.5 (Oct. 2010). Publisher: Institute of Mathematical Statistics. ISSN: 0090-5364. DOI: 10.1214/10-aos797. URL: <http://dx.doi.org/10.1214/10-AOS797>.

- [107] S. D. Abdul Bujang et al. “Imbalanced Classification Methods for Student Grade Prediction: A Systematic Literature Review”. In: *IEEE Access* 11 (2023), pp. 1970–1989. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2022.3225404.
- [108] Omer Sagi and Lior Rokach. “Ensemble learning: A survey”. In: *WIREs Data Mining and Knowledge Discovery* 8.4 (July 2018). Publisher: John Wiley & Sons, Ltd, e1249. ISSN: 1942-4787. DOI: 10.1002/widm.1249. URL: <https://doi.org/10.1002/widm.1249> (visited on 07/12/2023).
- [109] Miriam Seoane Santos et al. “A unifying view of class overlap and imbalance: Key concepts, multi-view panorama, and open avenues for research”. In: *Information Fusion* 89 (2023), pp. 228–253. ISSN: 1566-2535. DOI: <https://doi.org/10.1016/j.inffus.2022.08.017>. URL: <https://www.sciencedirect.com/science/article/pii/S1566253522001099>.
- [110] Taisho Sasada et al. “A Resampling Method for Imbalanced Datasets Considering Noise and Overlap”. In: *Procedia Computer Science* 176 (2020), pp. 420–429. ISSN: 1877-0509. DOI: <https://doi.org/10.1016/j.procs.2020.08.043>. URL: <https://www.sciencedirect.com/science/article/pii/S1877050920318676>.
- [111] Sara Scaramuccia et al. “Computing multiparameter persistent homology through a discrete Morse-based approach”. In: *Computational Geometry* 89 (2020), p. 101623. ISSN: 0925-7721. DOI: <https://doi.org/10.1016/j.comgeo.2020.101623>. URL: <https://www.sciencedirect.com/science/article/pii/S0925772120300171>.
- [112] Robert E. Schapire. “The Boosting Approach to Machine Learning: An Overview”. In: *Nonlinear Estimation and Classification*. Ed. by David D. Denison et al. New York, NY: Springer New York, 2003, pp. 149–171. ISBN: 978-0-387-21579-2. DOI: 10.1007/978-0-387-21579-2\_9. URL: [https://doi.org/10.1007/978-0-387-21579-2\\_9](https://doi.org/10.1007/978-0-387-21579-2_9).
- [113] Robert E. Schapire and Yoav Freund. *Boosting: Foundations and Algorithms*. The MIT Press, May 2012. ISBN: 978-0-262-30118-3. DOI: 10.7551/mitpress/8291.001.

0001. URL: <https://doi.org/10.7551/mitpress/8291.001.0001> (visited on 09/29/2023).
- [114] H. Schenck. *Algebraic Foundations for Applied Topology and Data Analysis*. Mathematics of Data. Springer International Publishing, 2022. ISBN: 978-3-031-06663-4. URL: <https://books.google.com/books?id=4IQkzwEACAAJ>.
- [115] Bernhard Schölkopf et al. “Estimating the Support of a High-Dimensional Distribution”. In: *Neural Computation* 13.7 (July 2001), pp. 1443–1471. ISSN: 0899-7667. DOI: 10.1162/089976601750264965. URL: <https://doi.org/10.1162/089976601750264965> (visited on 04/25/2023).
- [116] Paolo Serafini. “Classifying negative and positive points by optimal box clustering”. In: *Discrete Applied Mathematics* 165 (2014), pp. 270–282. ISSN: 0166-218X. DOI: <https://doi.org/10.1016/j.dam.2013.05.003>. URL: <https://www.sciencedirect.com/science/article/pii/S0166218X1300228X>.
- [117] Shaukat Ali Shahee and Usha Ananthakumar. “An effective distance based feature selection approach for imbalanced data”. In: *Applied Intelligence* 50.3 (Mar. 2020), pp. 717–745. ISSN: 1573-7497. DOI: 10.1007/s10489-019-01543-z. URL: <https://doi.org/10.1007/s10489-019-01543-z>.
- [118] Victor S Sheng and Charles X Ling. “Cost-sensitive learning”. In: *Encyclopedia of Data Warehousing and Mining, Second Edition*. IGI Global, 2009, pp. 339–345.
- [119] Jacek Skryzalin and Pawin Vongmasa. “The computational complexity of multidimensional persistence”. In: *Proposed Journal Article, unpublished* 2017.SAND-2017-12740J (2017). Publisher: Sandia National Lab.(SNL-NM), Albuquerque, NM (United States).
- [120] Diego A. Socolinsky et al. “Fast Face Detection with a Boosted CCCD Classifier”. In: 2003.



- [121] Jake A. Soloff, Rina Foygel Barber, and Rebecca Willett. *Bagging Provides Assumption-free Stability*. arXiv:2301.12600 [cs, math, stat]. Apr. 2023. URL: <http://arxiv.org/abs/2301.12600> (visited on 07/20/2023).
- [122] Paria Soltanzadeh, M. Reza Feizi-Derakhshi, and Mahdi Hashemzadeh. “Addressing the class-imbalance and class-overlap problems by a metaheuristic-based under-sampling approach”. In: *Pattern Recognition* 143 (Nov. 2023), p. 109721. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2023.109721. URL: <https://www.sciencedirect.com/science/article/pii/S0031320323004193>.
- [123] Yan-Yan Song and LU Ying. “Decision tree methods: applications for classification and prediction”. In: *Shanghai archives of psychiatry* 27.2 (2015). Publisher: Shanghai Mental Health Center, p. 130.
- [124] Bo Sun and Haiyan Chen. “A survey of k nearest neighbor algorithms for solving the class imbalanced problem”. In: *Wireless Communications and Mobile Computing* 2021 (2021). Publisher: Hindawi Limited, pp. 1–12. ISSN: 1530-8677.
- [125] T. Brito-Sarracino et al. “Explainable Machine Learning for Breast Cancer Diagnosis”. In: *2019 8th Brazilian Conference on Intelligent Systems (BRACIS)*. Journal Abbreviation: 2019 8th Brazilian Conference on Intelligent Systems (BRACIS). Oct. 2019, pp. 681–686. ISBN: 2643-6264. DOI: 10.1109/BRACIS.2019.00124.
- [126] Ichigaku Takigawa, Mineichi Kudo, and Atsuyoshi Nakamura. “Convex sets as prototypes for classifying patterns”. In: *Engineering Applications of Artificial Intelligence* 22.1 (2009), pp. 101–108. ISSN: 0952-1976. DOI: <https://doi.org/10.1016/j.engappai.2008.05.012>. URL: <https://www.sciencedirect.com/science/article/pii/S0952197608001589>.
- [127] Wenyin Tang et al. “Classification for overlapping classes using optimized overlapping region detection and soft decision”. In: *2010 13th international conference on information fusion*. IEEE, 2010, pp. 1–8.

- [128] David MJ Tax and Robert PW Duin. “Support vector data description”. In: *Machine learning* 54 (2004). Publisher: Springer, pp. 45–66. ISSN: 0885-6125.
- [129] Kai Ming Ting and Ian H. Witten. *Stacking bagged and dagged models*. en. Working Paper. 97/09. 1997. URL: <https://hdl.handle.net/10289/1072>.
- [130] L. Torgo. *Data Mining with R, learning with case studies, 2nd edition*. Chapman and Hall/CRC, 2016. URL: <http://ltorgo.github.io/DMwR2>.
- [131] V. N. Vapnik and A. Ya. Chervonenkis. “On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities”. In: *Theory of Probability & Its Applications* 16.2 (1971). \_eprint: <https://doi.org/10.1137/1116025>, pp. 264–280. DOI: 10.1137/1116025. URL: <https://doi.org/10.1137/1116025>.
- [132] Vladimir Vapnik. “The support vector method of function estimation”. In: *Nonlinear modeling: Advanced black-box techniques*. Springer, 1998, pp. 55–85.
- [133] Oliver Vipond et al. “Multiparameter persistent homology landscapes identify immune cell spatial patterns in tumors”. In: *Proceedings of the National Academy of Sciences* 118.41 (2021). \_eprint: <https://www.pnas.org/doi/pdf/10.1073/pnas.2102166118>, e2102166118. DOI: 10.1073/pnas.2102166118. URL: <https://www.pnas.org/doi/abs/10.1073/pnas.2102166118>.
- [134] Piyanoot Vorraboot et al. “Improving classification rate constrained to imbalanced data between overlapped and non-overlapped regions by hybrid algorithms”. In: *Neurocomputing* 152 (Mar. 2015), pp. 429–443. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2014.10.007. URL: <https://www.sciencedirect.com/science/article/pii/S0925231214012788>.
- [135] Vladimir Vovk, Harris Papadopoulos, and Alexander Gammerman. *Measures of Complexity*. Springer, 2015.

- [136] Pattaramon Vuttipittayamongkol, Eyad Elyan, and Andrei Petrovski. “On the class overlap problem in imbalanced data classification”. In: *Knowledge-Based Systems* 212 (Jan. 2021), p. 106631. ISSN: 0950-7051. DOI: 10.1016/j.knosys.2020.106631. URL: <https://www.sciencedirect.com/science/article/pii/S0950705120307607>.
- [137] Hubert Wagner, Chao Chen, and Erald Vuçini. “Efficient Computation of Persistent Homology for Cubical Data”. In: *Topological Methods in Data Analysis and Visualization II: Theory, Algorithms, and Applications*. Ed. by Ronald Peikert et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 91–106. ISBN: 978-3-642-23175-9. DOI: 10.1007/978-3-642-23175-9\_7. URL: [https://doi.org/10.1007/978-3-642-23175-9\\_7](https://doi.org/10.1007/978-3-642-23175-9_7).
- [138] Douglas B. West. *Introduction to Graph Theory*. English. 2nd ed. New Jersey, USA: Prentice Hall, Sept. 2000. ISBN: 0-13-014400-2. URL: <https://www.bibsonomy.org/bibtex/27f7055c34dd93bc157fbe31a0c1e40de/cgray>.
- [139] Dennis L. Wilson. “Asymptotic Properties of Nearest Neighbor Rules Using Edited Data”. In: *IEEE Trans. Syst. Man Cybern.* 2 (1972), pp. 408–421. URL: <https://api.semanticscholar.org/CorpusID:6699477>.
- [140] Kelin Xia and Guo-Wei Wei. “Multidimensional persistence in biomolecular data”. In: *Journal of Computational Chemistry* 36.20 (2015). \_eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/jcc.23953> pp. 1502–1520. DOI: <https://doi.org/10.1002/jcc.23953>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.23953>.
- [141] Haitao Xiong, Junjie Wu, and Lu Liu. “Classification with ClassOverlapping: A Systematic Study”. In: (Dec. 2010). ISSN: 978-90-78677-40-6. DOI: 10.2991/icebi.2010.43.

- [142] Yuanwei Zhu et al. “EHSO: Evolutionary Hybrid Sampling in overlapping scenarios for imbalanced learning”. In: *Neurocomputing* 417 (2020), pp. 333–346. ISSN: 0925-2312. DOI: <https://doi.org/10.1016/j.neucom.2020.08.060>. URL: <https://www.sciencedirect.com/science/article/pii/S0925231220313412>.
- [143] Ali Zia et al. “Topological deep learning: a review of an emerging paradigm”. In: *Artificial Intelligence Review* 57.4 (Feb. 2024), p. 77. ISSN: 1573-7462. DOI: [10.1007/s10462-024-10710-9](https://doi.org/10.1007/s10462-024-10710-9). URL: <https://doi.org/10.1007/s10462-024-10710-9>.