Statistical Learning with Proximity Catch Digraphs

by

Artür Manukyan

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Doctor of Philosophy

in

Computational Science and Engineering

KOÇ
UNIVERSITY

September 15, 2017

Statistical Learning with Proximity Catch Digraphs

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a doctoral dissertation by

Artür Manukyan

and have found that it is complete and satisfactory in all respects,

and that any and all revisions required by the final

examining committee have been made.

Committee Members:

Assoc. Prof. Dr. Mine Çağlar

Prof. Dr. Selda Küçükçifçi

Prof. Dr. Atilla Gürsoy

Asst. Prof. Dr. Özgür Asar

Asst. Prof. Dr. Ümit Işlak

Date:

To Mom and Dad

ABSTRACT

In the field of statistical learning, a significant portion of methods model data as graphs. Proximity graphs, in particular, offer solutions to many challenges in supervised and unsupervised statistical learning. Among these graphs, *class cover catch digraphs* (CCCDs) have been introduced first to investigate the *class cover problem* (CCP), and then employed in classification and clustering. However, this family of digraphs can be improved further to construct better classifiers and clustering algorithms. The purpose of this thesis is to tackle popular problems in statistical learning like robustness, prototype selection and determining the number of clusters with *proximity catch digraphs* (PCD). PCDs are generalized versions of CCCDs and have been proven useful in spatial data analysis. We will investigate the performance of CCCDs and PCDs in both supervised and unsupervised statistical learning, and discuss how these digraph families address real life challenges. We show that CCCD classifiers perform relatively well when one class is more frequent than the others, an example of the *class imbalance problem.* Later, by using barycentric coordinate system and by extending the Delaunay tessellations to partition $\mathbb{R}^d$, we establish PCD based classifiers and clustering methods that are both robust to the class imbalance problem and have computationally tractable prototype sets, making them both appealing and fast. In addition, our clustering algorithms are parameter-free clustering adaptations of an unsupervised version of CCCDs, namely cluster catch digraphs (CCDs). We partition data sets by incorporating spatial data analysis tools based on Ripley's K function, and we also define cluster ensembles based on PCDs for boosting the performance. Such methods are crucial for real life practices where domain knowledge is often infeasible.

ÖZETÇE

İstatistiksel öğrenme alanındaki yöntemlerin anlamlı bir çoğunluğu veriyi çizgeler olarak modellemektedir. Yakınlık çizgeleri gözetimli ve gözetimsiz istatistiksel öğrenme alanlarındaki pek çok probleme çözümler sunmaktadırlar. Bu çizgeler arasında *sınıf örtüsü yakalama yönlü çizgeleri* (SÖYYÇ) sınıf örtüsü problemini (SÖP) çözmek için tanıtılmıştır. SÖYYÇler sınıflama ve kümeleme için de kullanılabilir. Ancak, bu yönlü çizgeler daha iyi sınıflama ve kümeleme yöntemleri geliştirmek için de genelleştirilebilirler. Bu tezin amacı, istatistiksel öğrenme alanındaki popüler sorunlara *yakınlık yakalama yönlü çizgeleri* (YYYÇ) ile çözümler sunmaktır. Bu sorunlar arasında; gürbüzlük, prototip seçimi ve küme sayısının tespiti gibi sorunlar yer almaktadır. YYYÇler esasında SÖYYÇlerin genelleştirilmiştir halleridir ve YYYÇler daha önce uzaysal veri analizi problemlerinde de kullanılmışlardır. Biz SÖYYÇlerin ve YYYÇlerin gözetimli ve gözetimsiz istatistiksel öğrenme alanındaki performanısı inceleyecek, bu çizgelerin gerçek yaşam problemlerin nasıl değinebileceğini tartışacağız. İlk olarak SÖYYÇ tabanlı sınıflayıcıların, veri setlerindeki sınıflardan herhangi birininin diğer sınıflardakinden daha çok gözleme sahip olduğunda, diğer sınıflayıcılara göre göreceli olarak iyi performans gösterdiğini vurgulayacağız. Bu probleme *sınıf dengesizliği problemi* ismi verilmektedir. Daha sonrasında, barisentrik kordinat sistemlerini kullanarak ve Delaunay mozaiklemelerini $\mathbb{R}^d$ yi mozaikleyecek şekilde genişleterek, YYYÇ tabanlı sınıflayıcılar ve kümeleme yöntemleri geliştireceğiz. Bu yöntemler, sınıf dengesizliklerine karşı gürbüz olacak ve hesapsal olarak takip edilebilen prototip setlerine sahip, cazip ve hızlı yöntemler olacaklardır. Özellikle kümeleme algoritmalarımız, parametrelerden bağımsız olarak tanımlanmış ve SÖYYÇlerin gözetimsiz halleri olan, *küme yakalama yönlü çizgelerine* (KYYÇ) dayalıdır. Biz veri setlerini, uzaysal veri analizinde kullanılan Ripley'nin K fonksiyonuna dayalı araçlar ile böleceğiz ve ayrıca

YYYÇlere dayalı küme toplulukları tanımlayıp kümeleme yöntemlerini destekleyen algoritmalar geliştireceğiz. Bu tür yöntemler ise veri setlerine mahsus olan alan bilgisini elde etmenin zor olduğu gerçek yaşam problemlerinde önemini göstereceklerdir.

ACKNOWLEDGMENTS

It was the graduation ceremony for Bachelor's in Statistics at Yıldız Technical University, July 2010. Although the weather was extremely hot and the venue was filled with hundreds of relatives and loved ones, somehow my family were able to make it to the ceremony. However, the most memorable thing was what my father told me: "Son, you are the first college graduate of our family, and I am proud of you". Over the years, these words have flourished into this dissertation. No one other than my family was this supportive and helpful to my academic journey and, therefore, I would like to dedicate my life's work to mom and dad. I will forever be grateful for all the sacrifices they made.

As a statistician, I am honoured to be given the opportunity to work with brilliant mathematicians who had tremendous impact on this dissertation. These people; Elvan Ceyhan, Selim Bahadır, Polat Charyyev and Çiğdem Ak, guided me to a better understanding of mathematical thinking. For that, I would like to first thank Prof. Elvan Ceyhan for his support, as an advisor and as a colleague. I was lucky 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 made my day.

Finally, I would like to thank the members of the 'Cool Office' for making this last four years of my life joyful and meaningful. It is indeed a remarkable gift to have your friends by your side when you need them the most. I hope that they have enjoyed my company as much as I enjoyed theirs. Some of the members I can list are; Haroon Qureshi (my boxing buddy), Bilal Tetik, Esra Ünsal, Müge Atış (my yoga buddy), Büşra Harmanda and Abdullah Kahraman (who happens to dislike the way I cook). I will always remember you and I pray for a future that we will never lose touch with each other, and possibly have more barbecue parties.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

PCD	Proximity catch digraph
CCCD	Class cover catch digraph
P-CCCD	Pure class CCCD
RW-CCCD	Random walk CCCD
PE	Proportional-edge
CS	Central-similarity
PE-PCD	Proportional-edge PCD
CS-PCD	Central-similarity PCD
CCP	Class cover problem
MDS	Minimum dominating set
SVM	Support vector machine
RBF	Radial basis function
AUC	Area under curve
CCR	Correct Classification Rate
k-**NN**	k nearest neighbor
C45-LP	C4.5 with Laplace smoothing and pruning
C45-LNP	C4.5 with Laplace smoothing and no pruning
CV	Cross validation
PCA	Principal Components Analysis
SVDD	Support vector data description
PE-k**NN**	Hybrid PE-PCD classifier with alternative classifier k-NN
CS-k**NN**	Hybrid CS-PCD classifier with alternative classifier k-NN
PE-SVM	Hybrid PE-PCD classifier with alternative classifier SVM
CS-SVM	Hybrid CS-PCD classifier with alternative classifier SVM

PE-CCCD	Hybrid PE-PCD classifier with alternative classifier CCCD
CS-CCCD	Hybrid CS-PCD classifier with alternative classifier CCCD
CSR	Complete Spatial Randomness
CCD	Cluster Catch Digraph
K-S	Kolmogorov-Smirnov
KS-CCD	K-S statistic based CCD
R-CCD	Ripley's K based CCD
pdfC	pdfCluster algorithm
FCmeans	fuzzy c-means
DBSCAN	Density-based spatial clustering of applications with noise

Part I

Preliminaries

Chapter 1

INTRODUCTION

1.1 Motivation

Machine learning methods based on set covering algorithms attracted particular attention recently. In this work, we approach learning data sets with methods that solve the class cover problem (CCP), where the goal is to find a region that encapsulates all the members of the data set or a class of interest. This particular region can be viewed as a *cover*; hence the name *class cover* (Cannon and Cowen, 2004). This problem is closely related to another problem in statistics, namely *support estimation*: estimating the support of a particular random variable defined in a measurable space (Schölkopf et al., 2001). Here, class cover are viewed as estimations of associated class conditional support.

Classification is the most relevant application of class covers. Since a cover is realized as an approximation of the class support, a new given point is labelled as the member of a particular class if the point is in its cover (Cannon and Cowen, 2004). In addition, by investigating the shape and structure of a cover, latent subclasses of a class may be revealed which makes these covers effective tools of exploratory data analysis (Priebe et al., 2003b). In statistical learning, numerous algorithms are based on graphs, and in particular, on *proximity graphs*. The family of class covering methods we discuss in this thesis utilize subsets of the data sets, i.e. *prototype sets*, which are equivalent to the minimum dominating sets (MDSs) of a particular type of proximity graph, called *proximity catch digraphs* (PCDs). Methods based on PCDs are examples of prototype selection methods and serve many purposes, such as pruning data sets, increasing classification speed, clustering of noisy data sets, etc.

1.2 Previous Work on the Class Cover Problem

The first examples of class covers estimated the support of a class with a collection of covering balls that a subset of balls are chosen with approximation algorithms (Cannon and Cowen, 2004). However, class covers have been extended to allow the use of different type of regions to cover a class of interest. Serafini (2014) uses sets of boxes to find covers, of classes, and also defines the maximum redundancy problem. This is an optimization problem of covering as many points as possible by each box where the total number of boxes are kept to a (approximately) minimum. Hammer et al. (2004) investigates CCP using boxes with applications to the logical data analysis. Moreover, Bereg et al. (2012) extend covering boxes to rectilinear polygons to cover classes, and they report on the complexity of the CCP algorithms using such polygonal covering regions. Takigawa et al. (2009) incorporate balls and establish classifiers similar to the ones based on CCCDs, and they also use sets of convex hulls. Ceyhan (2005) uses sets of triangles relative to the tessellation of the opposite class to analytically compute the minimum number of triangles required to establish a class cover. In this thesis, we study class covers with particular triangular regions (simplical regions in higher dimensions). We give some examples in Figure 1.1.

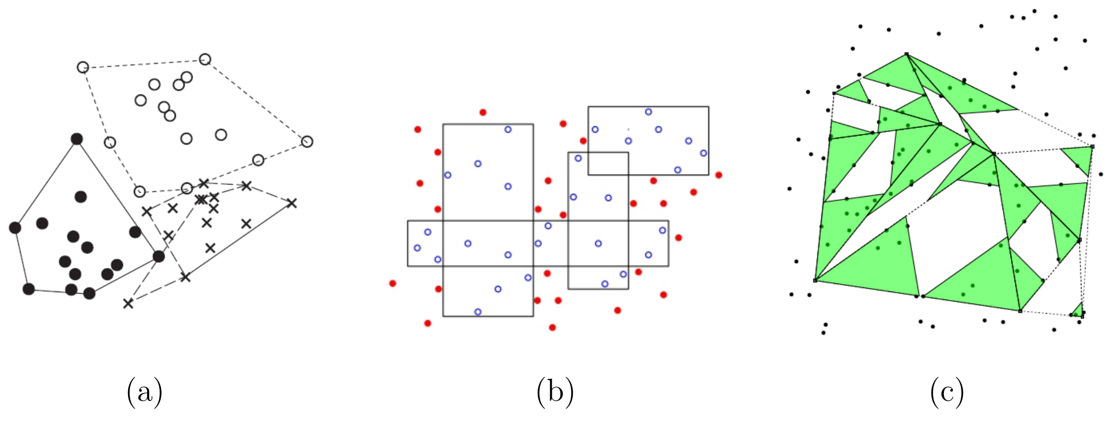

(a) (b) (c)

Figure 1.1: Examples of class covers with various covering regions. (a) convex hulls, (b) axis parallel strips and (c) triangles (Bereg et al., 2012; Ceyhan, 2005; Takigawa et al., 2009)

The class cover catch digraphs (CCCD) provide graph theoretic solutions to the CCP where the class covers are characterized with the minimum dominating sets (Priebe et al., 2001). The approximate minimum dominating sets of CCCDs (which were obtained by a greedy algorithm) and radii of the covering balls have been used to establish efficient classifiers. They introduced CCCDs to find graph theoretic solutions to the CCP problem, and provided some results on the minimum dominating sets and the distribution of the domination number of such digraphs for one dimensional data sets. DeVinney et al. (2002) defined random walk CCCDs (RW-CCCDs) where balls of class covers establish classifiers less prone to overfitting. CCCDs have been applied in face detection and latent class discovery in gene expression data sets (Priebe et al., 2003b; Socolinsky et al., 2003).

1.3 A Brief Introduction to Proximity Catch Digraphs

Proximity catch digraph (PCD) $D = (\mathcal{V}, \mathcal{A})$ with vertex set $\mathcal{V} = \mathcal{X}$ and arc set $\mathcal{A}$ is defined by $(u, v) \in \mathcal{A} \iff \{u, v\} \subset \mathcal{X}$ and $v \in \mathcal{N}(u)$. Each vertex $u \in \mathcal{V}$ is associated with a proximity region $\mathcal{N}(u)$ characterized by the proximity map $\mathcal{N}(\cdot)$. Here, $v \in \mathcal{N}(u)$ is viewed as the notion of u *catching* v since it is the within the *proximity* of u. Hence, the name *proximity catch digraph*.

PCDs are closely related to Class Cover Catch Digraphs (CCCDs) introduced by Priebe et al. (2001), and are vertex-random digraphs (directed graphs) defined by the relationship between either unlabeled or class-labeled observations. Ceyhan (2005) defined PCDs and introduced three families of PCDs to analytically compute the distribution of the domination number of such digraphs in a two class setting. Domination number and, another graph invariant, the arc density (the ratio of number of arcs in a digraph to the total number of arcs possible) of these PCDs have been used for testing spatial patterns of segregation and association (Ceyhan and Priebe, 2005; Ceyhan et al., 2007, 2006).

In this thesis, we investigate PCDs in both classification and clustering. We assess the performance of CCCDs (which are families of PCDs with spherical proximity maps) in the class imbalance problem, offer tools for the construction of prototype-based classifiers with computational tractable exact prototype sets, establish algorithms that partition data sets with no priori parameters, and offer cluster ensembles based on PCDs. In all applications, we use the estimates of the supports, i.e. covers, to model the data set as a digraph, and then, we find the (approximate) *minimum dominating sets* of these digraphs or other graph variants.

1.4 Contributions

We mainly discuss the CCP problem and the algorithms to solve them. CCCDs and PCDs provide prototype-based algorithms to tackle some challenges in machine learning which are topics of extensive research. PCDs are particularly appealing because they address the computational complications associated with the earlier CCCDs. The cover of the data set or the class cover reveals the inherent distribution of the data set, and hence, we use the covers to build discriminant regions or to detect regions of high density (or clusters). We give a list of various uses of PCD covers below:

(i) **Class imbalance problem** occurs when size of one class is substantially different than the other class. The majority class (i.e., the class with larger size) confounds the detection of objects which are originally from the minority class (i.e., the class with fewer points). We show that CCCD classifiers are robust to the class imbalance problem; that is, these classifiers are not affected by the abundance of one or more classes. We have first discovered the robustness property of CCCDs by investigating a data set with two classes that are imbalanced in an unusual way. The data set exhibits what we call a *local imbalance*, usually occuring around the area where two classes overlap. These data sets may have equal number of observations in both classes, hence it may not be possible to detect the imbalance at first sight. However, CCCDs does local pruning of the data sets, mitigating the effects of local imbalances as we show in Chapter 4. On the other hand, CCCDs substantially undersample from the majority class while preserving the information on the discarded points during the undersampling process. Many state-of-the-art methods, however, keep this information by means of ensemble classifiers, but CCCDs yield only a single classifier with the same property, making it both appealing and fast.

(ii) **Prototype selection** deals with the problem of selecting members of a data set so as to attain various tasks including reducing, condensing or summarizing a data set. Many learning methods aim to carry out more than one of these tasks, thereby building efficient learning algorithms (Bien and Tibshirani, 2011; Pękalska et al., 2006). PCDs help reducing the data set to a subset called *prototype set* that reduce the model complexity of class covers. However, PCDs are constructed with respect to the Delaunay tessellation of a subset of the data set which does not apply to the entire $\mathbb{R}^d$. We offered a partitioning scheme based on Delaunay tessellation using a construction of Deng and Zhu (1999) that partitions the entire space. Hence, we defined a unique tessellation of $\mathbb{R}^d$ with respect to the Delaunay tessellation of a set, which is a topic of Chapter 2, and we built prototype-based classifiers which reduce the number of observations of the entire data set in polynomial time. One common property of most prototype selection methods is that these algorithms are NP-hard, and exact solutions are mostly provided by approximation algorithms (Vazirani, 2001). However, Ceyhan (2010) showed that CCCDs find minimum prototype sets in polynomial time given the data set reside in $\mathbb{R}$, and its probabilistic behaviour is mathematically tractable. Fortunately, PCDs have the same property in higher dimensions which makes them appealing in selecting prototypes since the minimum set is computationally tractable regardless of the dimensions. In Chapter 5, we define several types of PCD based classifiers that are both computational tractable and robust to the class imbalance problem.

(iii) **Parameter-free clustering** is an intriguing line of research since algorithms that provides the (estimated) number of clusters without any input parameter are still appealing even though there exist methods to validate and compare the quality of the partitioning of a data set. Human eye is proficient in detecting the second-order properties of spatial objects in 2 dimensional (or in 3 dimensional) domains (Julesz, 1975). Ripley (1977) showed that the second-order moments of a point pattern could be reduced to a function $K(t)$ such that it can be used to devise test for possible clusterings in the domain. We offer density-based clustering methods that combine $K(t)$ with recently introduced unsupervised adaptations of CCCDs. Many existing clustering algorithms require either the assumed number of clusters, or a parameter representing some threshold for the local density of a possibly existing cluster. The latter type of algorithms rely on parameters viewed as the spatial intensity of the data set, i.e. expected number of objects in a unit area. Choice of such parameters are often challenging since different values of such parameters may drastically change the result. Our methods locate the clusters without any parameters via using the $\hat{K}(t)$ (the estimate of the K) that estimates the spatial intensity as we show it in Chapter 6.

(iv) **Ensemble-based clustering** are known to boost the performance of traditional clustering methods. Especially, bagging (bootstrap aggregating) is a traditional method for improving accuracy (Strehl and Ghosh, 2002). We offer clustering algorithms based on ensembles of PCDs. These methods are associated with two seperate proximity maps, called proportional-edge and central-similarity proximity maps. However, PCDs associated with these proximity maps are poorly defined for unlabeled data sets. Hence, in Chapter 7, we show that ensembles of randomly labeled data sets can be used to establish PCD based clustering methods for partitioning an unlabeled data set.

Chapter 2

PRELIMINARY TOOLS

2.1 Introduction

Let $(\Omega, \mathcal{M})$ be a measurable space, and let the training data set $\mathcal{X}$ be a set of Ω-valued random variables with class conditional distribution F, with support $s(F)$, and with sample size $n := |\mathcal{X}|$. For $\Omega = \mathbb{R}^d$, we develop rules to define proximity catch digraphs for the data set $\mathcal{X}$ (Ceyhan, 2005). We also focus on two PCD families where we assume that $\mathcal{X}$ is composed of two non-empty sets, $\mathcal{X}_0$ and $\mathcal{X}_1$, which are sets of $\mathbb{R}^d$-valued random variables with class conditional distributions F_0 and F_1, with supports $s(F_0)$ and $s(F_1)$, and with sample sizes $n_0 := |\mathcal{X}_0|$ and $n_1 := |\mathcal{X}_1|$, respectively. We define PCDs for the class of interest, i.e. *target class*, $\mathcal{X}_j$, for $j = 0, 1$, with respect to the *Delaunay tessellation* of the class of non-interest, i.e. *non-target class* $\mathcal{X}_{1-j}$.

A tessellation in $\mathbb{R}^d$ is a collection of non-intersecting (actually intersecting possibly only on boundaries) convex d-polytopes such that their union covers a region. We partition $\mathbb{R}^d$ into non-intersecting d-simplices and d-polytopes to construct PCDs that tend to have multiple disconnected components. We show that such a partitioning of the domain provides digraphs with computationally tractable minimum dominating sets for PCDs. In addition, we use the barycentric coordinate system to characterize the points of the target class with respect to the Delaunay tessellation of the non-target class. Such a coordinate system simplifies the definitions of many tools associated with PCDs in $\mathbb{R}^d$; such as, vertex and edge regions, minimum dominating sets and convex distance functions.

Finally, we discuss a spatial data analysis tool using Ripley's K function, testing possible clusterings in the domain. We use Ripley's K function with unsupervised adaptations of CCCDs, namely cluster catch digraphs, to establish density-based clustering methods that find the optimal partitioning of unlabelled data sets, without the definition of any priori parameter. We also combine the barycentric coordinate system and K function to characterize the minimum dominating sets of PCDs to construct ensemble-based clustering algorithms.

2.2 *Delaunay Tessellation of* $\mathbb{R}^d$

The convex hull of the non-target class $C_H(\mathcal{X}_{1-j})$ can be partitioned into *Delaunay cells* through the Delaunay tessellation of $\mathcal{X}_{1-j} \subset \mathbb{R}^2$. The Delaunay tessellation becomes a triangulation which partitions $C_H(\mathcal{X}_{1-j})$ into non intersecting triangles. For the points in the general position, the triangles in the Delaunay triangulation satisfy the property that the circumcircle of a triangle contain no points from $\mathcal{X}_{1-j}$ except for the vertices of the triangle. In higher dimensions, Delaunay cells are d-simplices (for example, a tetrahedron in $\mathbb{R}^3$). Hence, the $C_H(\mathcal{X}_{1-j})$ is the union of a set of disjoint d-simplices $\{\mathfrak{S}_k\}_{k=1}^K$ where K is the number of d-simplices, or Delaunay cells. Each d-simplex has $d+1$ non-coplanar vertices where none of the remaining points of $\mathcal{X}_{1-j}$ are in the interior of the circumsphere of the simplex (except for the vertices of the simplex which are points from $\mathcal{X}_{1-j}$). Hence, simplices of the Delaunay tessellations are more likely to be acute (simplices with no substantially small inner angles). Note that Delaunay tessellation is the dual of the *Voronoi diagram* of the set $\mathcal{X}_{1-j}$. A Voronoi diagram is a partitioning of $\mathbb{R}^d$ into convex polytopes such that the points inside each polytope is closer to the point associated with the polytope than any other point in $\mathcal{X}_{1-j}$. Hence, a polytope $\mathcal{V}(\mathsf{y})$ associated with a point $\mathsf{y} \in \mathcal{X}_{1-j}$ is defined as

$$\mathcal{V}(\mathsf{y}) = \{v \in \mathbb{R}^d : \|v - \mathsf{y}\| \leq \|v - z\| \text{ for all } z \in \mathcal{X}_{1-j} \setminus \{\mathsf{y}\}\}.$$

Here, $\|\cdot\|$ stands for the usual Euclidean norm. Observe that the Voronoi diagram is unique for a fixed set of points $\mathcal{X}_{1-j}$. A Delaunay graph is constructed by joining the pairs of points in $\mathcal{X}_{1-j}$ whose boundaries of voronoi polytopes are intersecting. The edges of the Delaunay graph constitute a partitioning of $C_H(\mathcal{X}_{1-j})$, hence the

Delaunay tessellation. By the uniqueness of the Voronoi diagram, the Delaunay tessellation is also unique (except for cases where $d+1$ or more points lie on the same circle of hypersphere). An illustration of the Voronoi diagram and the corresponding Delaunay triangulation in $\mathbb{R}^2$ are given in Figure 2.1(a) and (b).

A Delaunay tessellation partitions only $C_H(\mathcal{X}_{1-j})$ and do not offer a partitioning of the complement $\mathbb{R}^d \setminus C_H(\mathcal{X}_{1-j})$ unlike the Voronoi diagrams. As we will see in the following sections, this drawback makes the definition of our semi-parametric classifiers more difficult. Let *facets* of $C_H(\mathcal{X}_{1-j})$ be the simplices on the boundary of $C_H(\mathcal{X}_{1-j})$. To partition $\mathbb{R}^d \setminus C_H(\mathcal{X}_{1-j})$, we define unbounded regions associated with each facet of $C_H(\mathcal{X}_{1-j})$, namely *outer simplices* in $\mathbb{R}^d$ or *outer triangles* in $\mathbb{R}^2$. Each outer simplex is constructed by a single facet of $C_H(\mathcal{X}_{1-j})$, denoted by $\mathcal{F}_l$ for $l = 1, \cdots, L$. Here, L is the number of boundary facets and, note that, each facet is a $(d-1)$-simplex. Let $\{p_1, p_2, \cdots, p_N\} \subseteq \mathcal{X}_{1-j}$ be the set of points on the boundary of $C_H(\mathcal{X}_{1-j})$, and let $C_M := \sum_{i=1}^{N} p_i/N$ be the center of mass of $C_H(\mathcal{X}_{1-j})$. We use the bisector rays of Deng and Zhu (1999) as frameworks for constructing outer simplices, however such rays are not well defined for convex hulls in $\mathbb{R}^d$ for $d > 2$. Let the ray emanating from C_M through p_i be denoted as $\overrightarrow{C_M p_i}$. Hence, we define the outer simplices by rays emanating from each boundary points p_i to outside of $C_H(\mathcal{X}_{1-j})$ in the direction of $\overrightarrow{C_M p_i}$. Each facet $\mathcal{F}_l$ has d boundary points adjacent to it, and the rays associated with these boundary points establish an unbounded region together with the facet $\mathcal{F}_l$. Such a region can be viewed as an infinite "drinking glass" with $\mathcal{F}_l$ being the bottom while top of the glass reaching infinity, similar to intervals in $\mathbb{R}$ with infinite endpoints. Let $\mathscr{F}_l$ denote the outer simplex associated with the facet $\mathcal{F}_l$. An illustration of outer triangles in $\mathbb{R}^2$ has been given in Figure 2.1(c) where the $C_H(\mathcal{X}_{1-j})$ has six facets, hence $\mathbb{R}^2 \setminus C_H(\mathcal{X}_{1-j})$ is partitioned into six disjoint unbounded regions.

2.3 Barycentric Coordinate System

The *barycentric coordinate system* was introduced by A.F. Möbius in his book "The Barycentric Calculus" in 1837. The idea is to define weights w_1, w_2 and w_3 associated

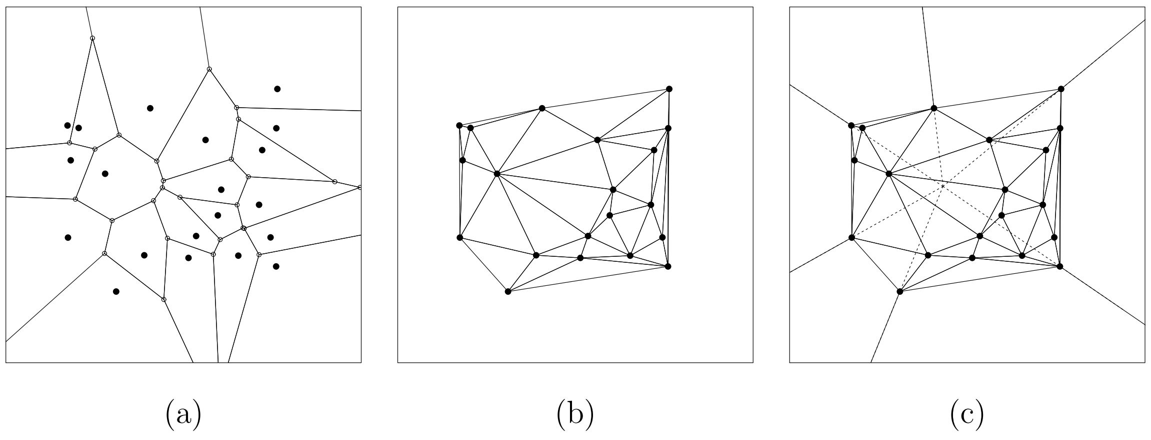


Figure 2.1: (a) A Voronoi diagram of points $\mathcal{X}_{1-j} \subset \mathbb{R}^2$ and (b) the associated the Delaunay triangulation, partitioning $C_H(\mathcal{X}_{1-j})$. (c) The Delaunay tessellation of $\mathcal{X}_{1-j}$ with rays $\overrightarrow{C_M p_i}$ for $i = 1, \ldots, 6$ that yield a partitioning of $\mathbb{R}^2 \setminus C_H(\mathcal{X}_{1-j})$. The dashed lines illustrate the direction of these rays where they meet at the point C_M, center of mass of $C_H(\mathcal{X}_{1-j})$.

with points y_1, y_2 and y_3 which constitute a triangle T in $\mathbb{R}^2$, respectively (Ungar, 2010). Hence the center of mass, or the *barycenter*, for $w_1 + w_2 + w_3 \neq 0$ is given by

$$P = \frac{w_1 \mathsf{y}_1 + w_2 \mathsf{y}_2 + w_3 \mathsf{y}_3}{w_1 + w_2 + w_3}. \tag{2.1}$$

Similarly, let $\mathfrak{S} = \mathfrak{S}(\mathcal{Y})$ be a d-simplex defined by the non-coplanar points $\mathcal{Y} = \{\mathsf{y}_1, \mathsf{y}_2, \cdots, \mathsf{y}_{d+1}\} \subset \mathbb{R}^d$ with weights $(w_1, w_2, \cdots, w_{d+1})$. Thus, the barycenter $W \in \mathbb{R}^d$ is given by

$$W = \frac{\sum_{i=1}^{d+1} w_i \mathsf{y}_i}{\sum_{i=1}^{d+1} w_i} \quad \text{with} \quad \sum_{i=1}^{d+1} w_i \neq 0. \tag{2.2}$$

The $(d+1)$-tuple $\mathbf{w} = (w_1, w_2, \cdots, w_{d+1})$ can also be viewed as a set of coordinates of W with respect to the set $\mathcal{Y} = \{\mathsf{y}_1, \mathsf{y}_2, \cdots, \mathsf{y}_{d+1}\}$ for $d > 0$. Hence, the name *barycentric coordinates*. Observe that W in Equation (2.1) is scale invariant (i.e. invariant under scaling of the weights of W). Therefore, the set of barycentric coordinates, also denoted as $(w_1 : w_2 : \ldots : w_{d+1})$, are homogeneous, i.e., for any $\lambda \in \mathbb{R}_+$,

$$(w_1 : w_2 : \ldots : w_{d+1}) = (\lambda w_1 : \lambda w_2 : \ldots : \lambda w_{d+1}). \tag{2.3}$$

This gives rise to *special barycentric coordinates* $\mathbf{w}' = (w'_1, w'_2, \cdots, w'_{d+1})$ of a point $x \in \mathbb{R}^d$ with respect to the set $\mathcal{Y}$ as follows:

$$\sum_{i=1}^{d+1} w'_i = \sum_{i=1}^{d+1} \frac{w_i}{w_{tot}} = 1, \tag{2.4}$$

where $w_{tot} := \sum_{j=1}^{d+1} w_j$. For the sake of simplicity, we refer to the special (or normalized) barycentric coordinates just as "barycentric coordinates", and use $\mathbf{w}$ to denote the set of this coordinates of x. Hence, the vector $\mathbf{w}$ is the solution to the linear systems of equations

$$\mathbf{A}\mathbf{w} = \begin{bmatrix} \mathsf{y}_2 - \mathsf{y}_1 & \mathsf{y}_3 - \mathsf{y}_1 & \cdots & \mathsf{y}_{d+1} - \mathsf{y}_1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix} = x - \mathsf{y}_1 \tag{2.5}$$

where $\mathbf{A} \in \mathbb{R}^{d \times d}$ is a matrix whose columns are vectors defined by $\mathsf{y}_k - \mathsf{y}_1$ in $\mathbb{R}^d$ for $k = 2, \cdots, d+1$. Note that $w_{d+1} = 1 - \sum_{i=1}^{d} w_i$. The set $\mathbf{w}$ is unique since vectors $\mathsf{y}_k - \mathsf{y}_1$ are linearly independent but w_i are not necessarily in $(0,1)$. Barycentric coordinates define whether the point x is in $\mathfrak{S}(\mathcal{Y})$ or not, as follows:

- $x \in \mathfrak{S}(\mathcal{Y})^o$ if $w_i \in (0,1)$ for all $i = 0, 1, \cdots, d+1$: the point x is inside of the d-simplex $\mathfrak{S}(\mathcal{Y})$ where $\mathfrak{S}(\mathcal{Y})^o$ denotes the interior of $\mathfrak{S}(\mathcal{Y})$,

- $x \in \partial(\mathfrak{S}(\mathcal{Y}))$, the point x is on the boundary of $\mathfrak{S}(\mathcal{Y})$, if $w_i = 0$ and $w_j = (0,1]$ for some I such that $i \in I \subset \{0, 1, \cdots, d+1\}$ and $j \in \{0, 1, \cdots, d+1\} \setminus I$,

- $x = \mathsf{y}_i$ if $w_i = 1$ and $w_j = 0$ for any $i = 0, 1, \cdots, d+1$ and $j \neq i$: the point x is at the a corner of $\mathfrak{S}(\mathcal{Y})$,

- $x \notin \mathfrak{S}(\mathcal{Y})$ if $w_i \notin [0,1]$ for some $i \in \{0, 1, \cdots, d+1\}$: the point x is outside of $\mathfrak{S}(\mathcal{Y})$.

Barycentric coordinates of a point $x \in \mathfrak{S}(\mathcal{Y})$ can also be viewed as the convex combination of the points of $\mathcal{Y}$, the vertices on the boundary of $\mathfrak{S}(\mathcal{Y})$.

We use a general property of barycentric coordinates being invariant under affine transformations; that is, the coordinates of a point $x \in \mathbb{R}^d$ with respect to an arbitrary simplex $\mathfrak{S}$ is invariant under affine transformations of the simplex $\mathfrak{S}$. Let $\mathcal{G}$ be a group of affine transformations on $\mathbb{R}^d$ such that, for all $g \in \mathcal{G}$, $x' = g(x)$ iff $x' = Ax + b$ for some $A \in \mathbb{R}^{d \times d}$ and $b \in \mathbb{R}^d$. Hence, we have the following theorem.

Theorem 2.3.0.1. *Let $\mathcal{Y} = \{\mathsf{y}_1, \mathsf{y}_2, \cdots, \mathsf{y}_{d+1}\}$ be vertices of an arbitrary simplex $\mathfrak{S} = \mathfrak{S}(\mathcal{Y})$, and let $\mathbf{w}_{\mathfrak{S}}(x)$ denote the set of barycentric coordinates of $x \in \mathbb{R}^d$ with respect to the vertices of $\mathfrak{S}$. Also, let $\mathfrak{S}' = g(\mathfrak{S})$ be the simplex whose i'th vertex is given as $\mathsf{y}'_i = g(\mathsf{y}_i)$ for $i = 1, \cdots, d+1$. Therefore, we have that the set of barycentric coordinates $\mathbf{w}_{\mathcal{G}}(x)$ is invariant under the group $\mathcal{G}$, i.e. $\mathbf{w}_{\mathfrak{S}'}(x') = \mathbf{w}_{\mathfrak{S}}(x)$ for $x' = g(x)$.*

Proof: For $i = 1, \cdots, d+1$, let $\alpha_i = w^{(i)}_{\mathfrak{S}}(x)$ denote the i'th barycentric coordinate of x with respect to $\mathfrak{S}(\mathcal{Y})$. Given the affine transformation $g \in \mathcal{G}$ for some $A \in \mathbb{R}^{d \times d}$ and $b \in \mathbb{R}^d$ and $\mathsf{y}'_i = g(\mathsf{y}_i)$, observe that

$$\begin{aligned} x' = g(x) = Ax + b = A(\sum_{i=1}^{d+1} \alpha_i \mathsf{y}_i) + b = A(\sum_{i=1}^{d+1} \alpha_i \mathsf{y}_i) + \sum_{i=1}^{d+1} \alpha_i b \\ = \sum_{i=1}^{d+1} \alpha_i (A\mathsf{y}_i + b) = \sum_{i=1}^{d+1} \alpha_i \mathsf{y}'_i \end{aligned}$$

By the uniqueness property of barycentric coordinates, $w^{(i)}_{\mathfrak{S}}(x') = \alpha_i$. ■

2.4 Vertex and Face Regions

We first define vertex and edge regions in $\mathbb{R}^2$, and later, we generalize them to associated regions in $\mathbb{R}^d$ for $d > 2$. In $\mathbb{R}^2$, d-simplices are triangles and a Delaunay tessellation is simply called a Delaunay triangulation. We consider the vertex and face regions of both inner and outer triangles of a Delaunay tessellation of $C_H(\mathcal{X}_{1-j})$. Vertex and face regions are associated with, as the name implies, the vertices and faces of simplices, respectively.

Ceyhan and Priebe (2005) introduced the vertex and face (edges in $\mathbb{R}^2$) regions as auxiliary tools to define proximity regions. They also gave the explicit functional forms of these regions as a function of the coordinates of vertices $\{\mathsf{y}_1, \mathsf{y}_2, \mathsf{y}_3\}$. However, we characterize these regions based on barycentric coordinates as this coordinate system will be more convenient for computation in higher dimensions.

2.4.1 Vertex Regions in $\mathbb{R}^2$

Let $\mathcal{Y} = \{\mathsf{y}_1, \mathsf{y}_2, \mathsf{y}_3\} \subset \mathbb{R}^2$ be three non-collinear points, and let $T = T(\mathcal{Y})$ be the triangle formed by these points. Also, let e_i be the edge of T opposite to the vertex y_i for $i = 1, 2, 3$. We partition the triangle T into regions, called *vertex regions*. These regions are constructed based on a point, preferably a *triangle center* $M \in T^o$. Vertex regions partition T into disjoint regions (only intersecting on the boundary) such that each vertex region has only one of vertices $\{\mathsf{y}_1, \mathsf{y}_2, \mathsf{y}_3\}$ associated with it. In particular, M-vertex regions are classes of vertex regions, which are constructed by the lines from each vertex y_i to M. These lines cross the edge e_i at point M_i. By connecting M with each M_i, we attain regions associated with vertices $\{\mathsf{y}_1, \mathsf{y}_2, \mathsf{y}_3\}$. M-vertex region of y_i is denoted by $R_M(\mathsf{y}_i)$ for $i = 1, 2, 3$. For the sake of simplicity, we will refer M-vertex regions as vertex regions. Figure 2.2 illustrates the vertex regions of an acute triangle in $\mathbb{R}^2$. Hence, we have the following Propositions 2.4.1.1 for edge regions of inner triangles in $\mathbb{R}^2$.

Proposition 2.4.1.1. *Let* $\mathcal{Y} = \{\mathsf{y}_1, \mathsf{y}_2, \mathsf{y}_3\} \subset \mathbb{R}^2$ *be a set of three non-collinear points, and let the set of vertex regions* $\{R_M(\mathsf{y}_i)\}_{i=1,2,3}$ *partitions* $T(\mathcal{Y})$*. Hence for* $x, M \in T(\mathcal{Y})^o$*, we have* $x \in R_M(\mathsf{y}_i)$ *if and only if*

$$w_T^{(i)}(x) > \max_{\substack{j=1,2,3 \\ j \neq i}} \frac{m_i w_T^{(j)}(x)}{m_j}$$

for $i = 1, 2, 3$ *where* $\mathbf{w}_T(x) = \left(w_T^{(1)}(x), w_T^{(2)}(x), w_T^{(3)}(x)\right)$ *and* $\mathbf{m} = (m_1, m_2, m_3)$ *are barycentric coordinates of* x *and* M *with respect to* $T(\mathcal{Y})$*, respectively.*

Proof: It is sufficient to show the result for $i = 1$ (as others follow by symmetry).

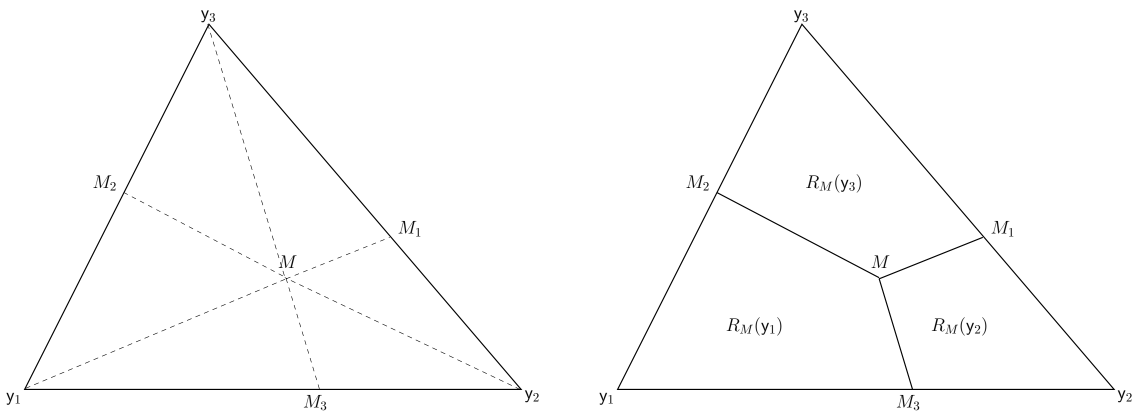


Figure 2.2: M-vertex regions of an acute triangle $T(\mathcal{Y}) = T(\mathsf{y}_1, \mathsf{y}_2, \mathsf{y}_3)$ with a center $M \in T(\mathcal{Y})^o$. (a) The dashed lines constitute the vertex regions. (b) Each M-vertex region is associated with a vertex y_i for $i = 1, 2, 3$.

Hence we show that, for $x \in R_M(\mathsf{y}_1)$, we have

$$w_T^{(1)}(x) > \max\left\{\frac{m_1 w_T^{(2)}(x)}{m_2}, \frac{m_1 w_T^{(3)}(x)}{m_3}\right\}.$$

Let $T_2(\mathcal{Y})$ and $T_3(\mathcal{Y})$ be the interiors of two triangles given by sets of points $\{\mathsf{y}_1, \mathsf{y}_2, M_2\}$ and $\{\mathsf{y}_1, \mathsf{y}_3, M_3\}$, respectively. Let $z \in T_2(\mathcal{Y})$ and let $w_{T_2}(z) = (\alpha_1, \alpha_2, \alpha_3)$ be the barycentric coordinates of z with respect to $T_2(\mathcal{Y})$. Then

$$\begin{aligned}
z &= \alpha_1 \mathsf{y}_1 + \alpha_2 \mathsf{y}_2 + \alpha_3 M_2 \\
&= \alpha_1 \mathsf{y}_1 + \alpha_2 \mathsf{y}_2 + \alpha_3 (b\mathsf{y}_1 + (1-b)\mathsf{y}_3) \\
&= (\alpha_1 + \alpha_3 b)\mathsf{y}_1 + \alpha_2 \mathsf{y}_2 + \alpha_3 (1-b)\mathsf{y}_3,
\end{aligned}$$

since M_2 lies on edge e_2, we can write it as $M_2 = b\mathsf{y}_1 + (1-b)\mathsf{y}_3$ for some $b \in (0,1)$. By the uniqueness of $\mathbf{w}_T(z)$, we have $w_T^{(1)}(z) = \alpha_1 + \alpha_3 b$ and $w_T^{(3)}(z) = \alpha_3(1-b)$. Hence,

$$\frac{w_T^{(1)}(z)}{w_T^{(3)}(z)} = \frac{\alpha_1 + \alpha_3 b}{\alpha_3(1-b)} > \frac{b}{(1-b)} = \frac{m_1}{m_3}$$

since $\alpha_i > 0$ for $i = 1, 2, 3$. Also, since M_2 and M are on the same line which crosses the edge e_2, for some $c \in (0,1)$:

$$\begin{aligned} M &= c\mathsf{y}_2 + (1-c)M_2 \\ &= c\mathsf{y}_2 + (1-c)(by_1 + (1-b)y_3) \\ &= b(1-c)\mathsf{y}_1 + c\mathsf{y}_2 + (1-b)(1-c)\mathsf{y}_3, \end{aligned}$$

Hence, $b(1-c) = m_1$ and $(1-b)(1-c) = m_3$, and observe that $m_1/m_3 = b/(1-b)$. Then, $T_2(\mathcal{Y}) = \{x \in T(\mathcal{Y})^o : w_T^{(1)}(x) > (m_1/m_3)w_T^{(3)}(x)\}$, and similarly, $T_3(\mathcal{Y}) = \{x \in T(\mathcal{Y})^o : w_T^{(1)}(x) > (m_1/m_2)w_T^{(2)}(x)\}$. Thus,

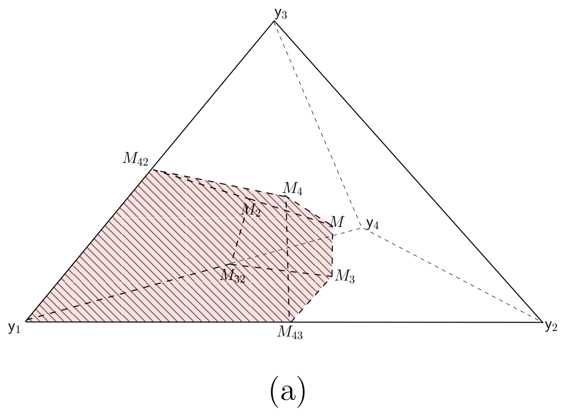
$$R_M(\mathsf{y}_1) = T_2(\mathcal{Y}) \cap T_3(\mathcal{Y}) = \left\{ x \in T(\mathcal{Y})^o : w_T^{(1)}(x) > \max\left\{ \frac{m_1 w_T^{(2)}(x)}{m_2}, \frac{m_1 w_T^{(3)}(x)}{m_3} \right\} \right\}. \quad \blacksquare$$

Note that, when $M := M_C$ the median (or the center of mass) of the triangle $T(\mathcal{Y})$, we can simplify the result of Proposition 2.4.1.1; that is, for any point $x \in T(\mathcal{Y})^o$, we have $x \in R_{M_C}(\mathsf{y}_i)$ if and only if $w_T^{(i)}(x) = \max_{j=1,2,3} w_T^{(j)}(x)$ since the set of barycentric coordinates of M_C is $\mathbf{m}_C = (1/3, 1/3, 1/3)$.

Vertex regions of outer triangles are defined in a different fashion to those in inner triangles. Let $e = \mathcal{F}$ be the edge (or facet) of $C_H(\mathcal{X}_{1-j})$ adjacent to vertices $\{\mathsf{y}_1, \mathsf{y}_2\}$. We denote the facet of $C_H(\mathcal{X}_{1-j})$ given by points y_1 and y_2 as e_∞; that is, it is opposite to a fictitious vertex y_∞ of the outer triangle $\mathscr{F}$. There is only a single vertex region of outer triangle $\mathscr{F}$, and it is the region associated with the vertex y_∞, i.e. $R_M(e_\infty)$. Therefore, we have $\mathscr{F} = R_M(e_\infty)$.

2.4.2 Vertex Regions in $\mathbb{R}^d$ with $d > 2$

The definitions of vertex regions in $\mathbb{R}^2$ can be extended to the ones in $\mathbb{R}^d$ for $d > 2$. A d-simplex is the smallest convex polytope in $\mathbb{R}^d$ constructed by a set of non-coplanar vertices $\mathcal{Y} = \{\mathsf{y}_1, \mathsf{y}_2, \cdots, \mathsf{y}_{d+1}\}$. The boundary of a d-simplex consists of k-simplices called k-faces for $0 \leq k < d$. Each k-face is a simplex defined by a subset of $\mathcal{Y}$ with k elements, hence there are $\binom{d+1}{k+1}$ k-faces in a d-simplex. Let $\mathfrak{S}(\mathcal{Y})$ be the simplex defined by the set of points $\mathcal{Y}$. Given a *simplex center* $M \in \mathfrak{S}(\mathcal{Y})^o$ (e.g. a triangle



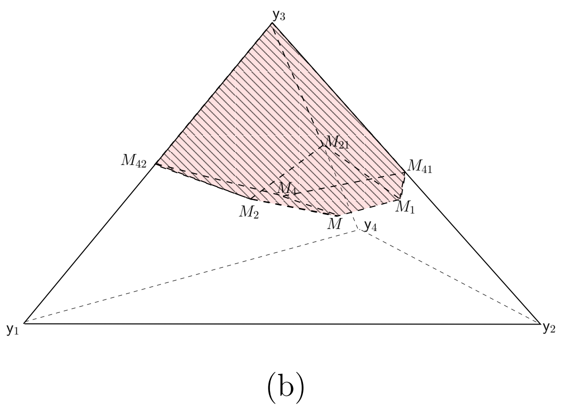


Figure 2.3: (a) M-vertex region $R_M(\mathsf{y}_1)$ of vertex y_1 and (b) $R_M(\mathsf{y}_3)$ of vertex y_3 of a 3-simplex, or a tetrahedron. M-vertex regions are shaded.

center in $\mathbb{R}^2$), there are $d+1$ M-vertex regions constructed by the set $\mathcal{Y}$. The M-vertex region of the vertex y_i is denoted by $R_M(\mathsf{y}_i)$ for $i = 1, 2, \cdots, d+1$.

For $i = 1, \ldots, d+1$, let f_i denote the $(d-1)$-face opposite to the vertex y_i. Observe that the lines through the points y_i and M cross the face f_i, a $(d-1)$-face, at the points M_i. Similarly, since the face f_i is a $(d-1)$-simplex with a center M_i for any $i = 1, \ldots, d+1$, we can find the centers of $(d-2)$-faces of this $(d-1)$-simplex. Note that both M_i and M are of same type of centers of their respective simplices f_i and $\mathfrak{S}(\mathcal{Y})$. The vertex region $R_M(\mathsf{y}_i)$ is the convex hull of the points y_i, $\{M_j\}_{j=1;j\neq i}^{d+1}$, and centers of all k-faces (which are also k-simplices) adjacent to y_i for $k = 1, \ldots, d-2$. Illustration of the vertex regions $R_M(\mathsf{y}_1)$ and $R_M(\mathsf{y}_3)$ of a 3-simplex (tetrahedron) are given in Figure 2.3. Each 2-face of this 3-simplex is a 2-simplex (a triangle). For example, in Figure 2.3(a), the points M_2, M_3 and M_4 are centers of f_2, f_3 and f_4, respectively. Moreover, these 2-simplices also have faces (1-faces or edges of the 3-simplex), and the centers of these faces are $\{M_{ij}\}_{i,j=1;i\neq j}^{4}$. Hence, the vertex region $R_M(\mathsf{y}_1)$ is a convex polytope of points $\{\mathsf{y}_1, M, M_2, M_3, M_4, M_{32}, M_{42}, M_{43}\}$ and $R_M(\mathsf{y}_3)$ is a convex polytope of points $\{\mathsf{y}_3, M, M_2, M_4, M_1, M_{42}, M_{41}, M_{21}\}$. The following theorem is an extension of the Proposition 2.4.1.1 to higher dimensions.

Theorem 2.4.2.1. *Let $\mathcal{Y} = \{\mathsf{y}_1, \mathsf{y}_2, \cdots, \mathsf{y}_{d+1}\} \subset \mathbb{R}^d$ be a set of non-coplanar points* *for $d > 0$, and let the set of vertex regions $\{R_M(\mathsf{y}_i)\}_{i=1}^{d+1}$ partitions $\mathfrak{S}(\mathcal{Y})$. Hence, for $x, M \in \mathfrak{S}(\mathcal{Y})^o$, we have $x \in R_M(\mathsf{y}_i)$ if and only if*

$$w_{\mathfrak{S}}^{(i)}(x) > \max_{\substack{j=1,\cdots,d+1\\ j\neq i}} \frac{m_i w_{\mathfrak{S}}^{(j)}(x)}{m_j} \tag{2.6}$$

where $\mathbf{w}_{\mathfrak{S}}(x) = \left(w_{\mathfrak{S}}^{(1)}(x), \cdots, w_{\mathfrak{S}}^{(d+1)}(x)\right)$ and $\mathbf{m} = (m_1, \ldots, m_{d+1})$ are the barycentric coordinates of x and M with respect to $\mathfrak{S}(\mathcal{Y})$, respectively.

Proof: We prove this theorem by induction on dimension d. The proof of the case $d = 1$ is trivial. For $\mathfrak{S}(\mathcal{Y}) = (\mathsf{y}_1, \mathsf{y}_2) \subset \mathbb{R}$ and $\mathsf{y}_1 < \mathsf{y}_2$, the vertex regions $R_M(\mathsf{y}_1)$ and $R_M(\mathsf{y}_2)$ are the intervals (y_1, M) and (M, y_2), respectively ($\{x = M\}$ and $\{x = \mathsf{y}_i\}$ have zero $\mathbb{R}$-Lebesgue measure). For $\alpha_1 \in (0, 1)$ and $\alpha_2 = 1 - \alpha_1$, let $\alpha_1\mathsf{y}_1 + \alpha_2\mathsf{y}_2$ be the convex (or barycentric) combination of $x \in \mathfrak{S}(\mathcal{Y})$. Hence, $x \in (\mathsf{y}_1, M) = R_M(\mathsf{y}_1)$ if and only if $\alpha_1/\alpha_2 > m_1/m_2$. The case $d = 2$ is proved in Proposition 2.4.1.1. Thus, there only remains the case $d > 2$. We suppose the statement is true for all faces of the d-simplices which are $d - 1$ dimensional, and by that, we will show that the statement is also true for the d-simplex which is d dimensional.

It is sufficient to show the result for y_1 (as the others follow by symmetry). Let $x \in R_M(\mathsf{y}_1)$ and note that the elements of the set of $(d-1)$-faces, $\{f_j\}_{j=2}^{d+1}$, are adjacent to y_1. Each of these faces are of $d - 1$ dimensions. Hence, they are $(d - 1)$-simplices and they also have their own vertex regions. Thus, let $R_{M_i}(\mathsf{y}_j, f_i)$ be the vertex region of y_j with respect to $(d - 1)$-simplex f_i for $j \neq i$. Note that M_i is the center of f_i. Now, let $w_{f_i}(z, \mathsf{y}_j) = w_{ij}$ be the barycentric coordinate of point z corresponding to y_j with respect to the f_i. Observe that w_{ii} is not defined since y_i is not a vertex of the face f_i.

Moreover, let $\mathbf{m}' = (m'_1, \ldots, m'_{i-1}, m'_{i+1}, \ldots, m'_{d+1})$ be the barycentric coordinates of M_i with respect to f_i, and note that M_i is a linear combination of M and y_i. Also, observe that m'_i is not defined since the vertex y_i is not a vertex of f_i. Hence, for $\beta \geq 1$,

$$M_i = \beta M + (1 - \beta)\mathsf{y}_i = \beta \left(\sum_{t=1;t\neq i}^{d+1} m_t \mathsf{y}_t \right) + (1 - \beta)\mathsf{y}_i. \tag{2.7}$$

Therefore, by the uniqueness of barycentric coordinates, $m'_t = \beta m_t$ for $t = 1, \ldots, d+1$ and $t \neq i$. Note that $(1-\beta) = 0$ since $M_i \in f_i$ and also $f_i \subset \partial(\mathfrak{S}(\mathcal{Y}))$. Hence, $\beta = 1$ which implies $m'_t = m_t$ for all $t \neq i$. Then, $m'_1/m'_j = m_1/m_j$ for $j = 2, 3, \ldots, d+1$ and $j \neq i$. We use this result on our induction hypothesis.

Now, for $i = 2, \ldots, d+1$, let the face f_i and line defined by x and y_i cross at the point z_i. Observe that $z_i \in f_i$, and since f_i is a $(d-1)$-simplex and $x \in R_M(\mathsf{y}_1)$, see that $z_i \in R_{M_i}(\mathsf{y}_1, f_i)$. By induction hypothesis and (2.7), we observe that $z_i \in R_{M_i}(\mathsf{y}_1, f_i)$ if and only if $w_{i1} > (m'_1/m'_j)w_{ij}$ if and only if $w_{i1} > (m_1/m_j)w_{ij}$ for $j = 2, 3, \ldots, d+1$ and $j \neq i$. Since the point x is the convex (and linear) combination of z_i and y_i, for $\alpha \in (0,1)$, we have

$$x = (1-\alpha)\mathsf{y}_i + \alpha z_i = (1-\alpha)\mathsf{y}_i + \alpha\left(\sum_{k=1;k\neq i}^{d+1} w_{ik}\right).$$

By the uniqueness property of barycentric coordinates, it follows that $w_{\mathfrak{S}}^{(1)}(x) = \alpha w_{i1}$ and $w_{\mathfrak{S}}^{(j)}(x) = \alpha w_{ij}$. Hence,

$$\frac{w_{\mathfrak{S}}^{(1)}(x)}{w_{\mathfrak{S}}^{(j)}(x)} = \frac{w_{i1}}{w_{ij}} > \frac{m_1}{m_j}. \tag{2.8}$$

Since (2.8) is true for all $i = 2, \ldots, d+1$, we see that $x \in R_M(y_1)$ if and only if $w_{\mathfrak{S}}^{(1)}(x) > (m_1/m_i)w_{\mathfrak{S}}^{(i)}(x)$. Hence, the result follows. ■

For $M = M_C$ and for any point $x \in \mathfrak{S}(\mathcal{Y})^o$, we have $x \in R_{M_C}(\mathsf{y}_i)$ if and only if $w_T^{(i)}(x) = \max_j w_T^{(j)}(x)$ since the set of barycentric coordinates of M_C is $\mathbf{m}_C = (1/(d+1), 1/(d+1), \ldots, 1/(d+1))$. The M_C-vertex regions are particularly appealing for our proportional-edge proximity maps.

Vertex regions of outer triangles are defined in a different fashion to those in inner triangles. Let $e = \mathcal{F}$ be the edge (or facet) of $C_H(\mathcal{X}_{1-j})$ adjacent to vertices $\{\mathsf{y}_1, \mathsf{y}_2\}$. We denote the facet of $C_H(\mathcal{X}_{1-j})$ given by points y_1 and y_2 as e_∞; that is, it is opposite to a fictitious vertex y_∞ of the outer triangle $\mathscr{F}$. There is only a single vertex region of outer triangle $\mathscr{F}$, and it is the region associated with the vertex y_∞, i.e. $R_M(e_\infty)$. Therefore, we have $\mathscr{F} = R_M(e_\infty)$.

Let $\mathcal{F}$ be a facet of $C_H(\mathcal{X}_{1-j})$, and let $\mathcal{Y} = \{\mathsf{y}_1, \mathsf{y}_2 \ldots, \mathsf{y}_d\} \subset \mathbb{R}^d$ be a set of adjacent points on the boundary of $C_H(\mathcal{X}_{1-j})$ associated with $\mathcal{F}$. Let the outer simplex $\mathscr{F} \subset \mathbb{R}^d$ be defined by the set $\mathcal{Y}$ and the rays $\{\overrightarrow{C_M \mathsf{y}_1}, \ldots, \overrightarrow{C_M \mathsf{y}_d}\}$. We denote the facet of $C_H(\mathcal{X}_{1-j})$ given by the set $\mathcal{Y}$ as f_∞; that is, it is opposite to a fictitious vertex y_∞ of the outer simplex $\mathscr{F}$. There is only a single vertex region of outer simplex $\mathscr{F}$, and it is the region associated with the vertex y_∞, i.e. $R_M(f_\infty)$. Therefore, we have $\mathscr{F} = R_M(f_\infty)$.

2.4.3 Edge Regions (Face Regions in $\mathbb{R}^2$)

Let $\mathcal{Y} = \{\mathsf{y}_1, \mathsf{y}_2, \mathsf{y}_3\} \subset \mathbb{R}^2$ be the set of non-collinear points that constitutes some inner triangle $T = T(\mathcal{Y})$ of $C_H(\mathcal{X}_{1-j})$. Let e_i be the edge opposite to the vertex y_i for $i = 1, 2, 3$. We partition the triangle T into regions, namely *edge regions*. These regions are constructed based on a point, preferably a *triangle center* $M \in T^o$. Here, T_I^o is the interior of the triangle T. Edge regions are associated with the edges opposite to each vertex of $\mathcal{Y}$. We attain the edge regions by joining each y_i with M, also referred as M-edge regions. M-edge region of e_i is denoted by $R_M(e_i)$ for $i = 1, 2, 3$. Observe that in M-edge regions, the interiors of these regions are disjoint. For the sake of simplicity, we will refer M-edge regions as edge regions. Hence, we have the following Propositions 2.4.3.1 for edge regions of inner triangles in $\mathbb{R}^2$.

Proposition 2.4.3.1. *Let $\mathcal{Y} = \{\mathsf{y}_1, \mathsf{y}_2, \mathsf{y}_3\} \subset \mathbb{R}^2$ be three non-collinear points, and let the set of edge regions $\{R_M(e_i)\}_{i=1,2,3}$ partition $T(\mathcal{Y})$. Hence for $x, M \in T(\mathcal{Y})^o$, we have $x \in R_M(e_i)$ if and only if*

$$w_T^{(i)}(x) < \min_{\substack{j=1,2,3 \\ j \neq i}} \frac{m_i w_T^{(j)}(x)}{m_j}$$

for $i = 1, 2, 3$ where $\mathbf{w}_T(x) = \left(w_T^{(1)}(x), w_T^{(2)}(x), w_T^{(3)}(x)\right)$ and $\mathbf{m} = (m_1, m_2, m_3)$ are barycentric coordinates of x and M with respect to $T(\mathcal{Y})$, respectively.

Proof: It is sufficient to show the result for $i = 1$ (as the others follow by symmetry). Hence we show that, for $x \in R_M(e_1)$, we have

$$w_T^{(1)}(x) < \min\left\{\frac{m_1 w_T^{(2)}(x)}{m_2}, \frac{m_1 w_T^{(3)}(x)}{m_3}\right\}.$$

Let $T_2(\mathcal{Y})$ and $T_3(\mathcal{Y})$ be the interiors of two triangles constructed by sets of points $\{\mathsf{y}_2, \mathsf{y}_3, M_2\}$ and $\{\mathsf{y}_3, \mathsf{y}_2, M_3\}$, respectively. Let $z \in T_2(\mathcal{Y})$ and let $w_{T_2}(z) = (\alpha_1, \alpha_2, \alpha_3)$ be the barycentric coordinates of z with respect to $T_2(\mathcal{Y})$. Then

$$
\begin{aligned}
z &= \alpha_1 \mathsf{y}_2 + \alpha_2 \mathsf{y}_3 + \alpha_3 M_2 \\
&= \alpha_1 \mathsf{y}_2 + \alpha_2 \mathsf{y}_3 + \alpha_3 (b\mathsf{y}_1 + (1-b)\mathsf{y}_3) \\
&= \alpha_3 b \mathsf{y}_1 + \alpha_1 \mathsf{y}_2 + (\alpha_2 + (1-b)\alpha_3)\mathsf{y}_3
\end{aligned}
$$

since M_2 lies on edge e_2, we can write it as $M_2 = b\mathsf{y}_1 + (1-b)\mathsf{y}_3$ for some $b \in (0,1)$. By the uniqueness of $\mathbf{w}_T(z)$, we have $w_T^{(1)}(z) = \alpha_3 b$ and $w_T^{(3)}(z) = \alpha_2 + \alpha_3(1-b)$. Hence,

$$
\frac{w_T^{(1)}(z)}{w_T^{(3)}(z)} = \frac{\alpha_3 b}{\alpha_2 + \alpha_3(1-b)} < \frac{b}{(1-b)} = \frac{m_1}{m_3}
$$

since $\alpha_i > 0$ for $i = 1, 2, 3$. Also, since M_2 and M are on the same line which intersects the edge e_2, for some $c \in (0,1)$:

$$
\begin{aligned}
M &= c\mathsf{y}_2 + (1-c)M_2 \\
&= c\mathsf{y}_2 + (1-c)(by_1 + (1-b)y_3) \\
&= b(1-c)\mathsf{y}_1 + c\mathsf{y}_2 + (1-b)(1-c)\mathsf{y}_3,
\end{aligned}
$$

Hence, $b(1-c) = m_1$ and $(1-b)(1-c) = m_3$, and observe that $m_1/m_3 = b/(1-b)$. Then, $T_2(\mathcal{Y}) = \{x \in T(\mathcal{Y})^o : w_T^{(1)}(x) < (m_1/m_3) w_T^{(3)}(x)\}$, and similarly, $T_3(\mathcal{Y}) = \{x \in T(\mathcal{Y})^o : w_T^{(1)}(x) < (m_1/m_2) w_T^{(2)}(x)\}$. Thus,

$$
R_M(e_1) = T_2(\mathcal{Y}) \cap T_3(\mathcal{Y}) = \left\{ x \in T(\mathcal{Y})^o : w_T^{(1)}(x) < \min\left\{ \frac{m_1 w_T^{(2)}(x)}{m_2}, \frac{m_1 w_T^{(3)}(x)}{m_3} \right\} \right\}. \qquad \blacksquare
$$

Note that, when $M := M_C$ the median (or the center of mass) of the triangle $T(\mathcal{Y})$, we simplify the result of Proposition 2.4.3.1; that is, for any point $x \in T(\mathcal{Y})^o$, we have $x \in R_{M_C}(e_i)$ if and only if $w_T^{(i)}(x) = \min_{j=1,2,3} w_T^{(j)}(x)$ since the barycentric coordinates of M_C is $\mathbf{m}_C = (1/3, 1/3, 1/3)$.

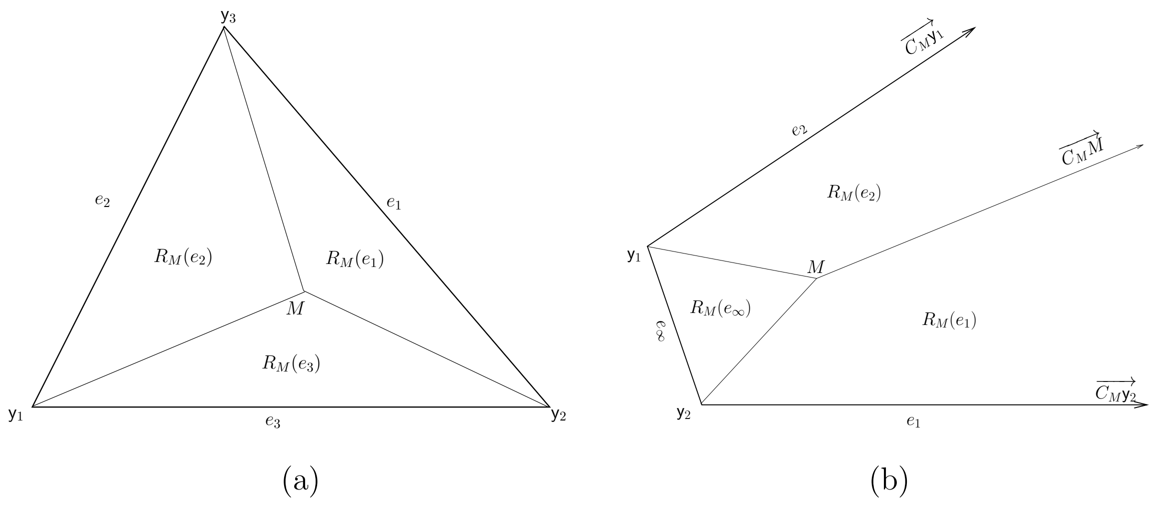


Figure 2.4: (a) Edge regions of an acute inner triangle $T(\mathcal{Y}) = T(\mathsf{y}_1, \mathsf{y}_2, \mathsf{y}_3)$ with a center $M \in T^o$. (b) Edge regions of an outer triangle $T(\mathcal{Y}) = T(\mathsf{y}_1, \mathsf{y}_2)$ with a center $M \in T^o$.

Edge regions of outer triangles are defined in a similar fashion to those in inner triangles. However, some of these regions are bounded and some are unbounded unlike in inner triangles. Let $\mathscr{F} \subset \mathbb{R}^2$ be an outer triangle defined by the adjacent boundary points $\{\mathsf{y}_1, \mathsf{y}_2\} \subset \mathbb{R}^2$ of $C_H(\mathcal{X}_{1-j})$ and by rays $\overrightarrow{C_M\mathsf{y}_1}$ and $\overrightarrow{C_M\mathsf{y}_2}$ for C_M being the median of the boundary points of $C_H(\mathcal{X}_{1-j})$. Also, let $e = \mathcal{F}$ be the edge (or facet) of $C_H(\mathcal{X}_{1-j})$ adjacent to vertices $\{\mathsf{y}_1, \mathsf{y}_2\}$. We partition $\mathscr{F}$ into regions associated with its edges. These regions are constructed based on a point, preferably a point realized as a *center* $M \in \mathscr{F}$. We attain the edge regions by joining each y_i with M. We define an additional line in the direction of $\overrightarrow{C_M M}$ to attain all edge regions. We denote the facet of $C_H(\mathcal{X}_{1-j})$ given by points y_1 and y_2 as e_∞; that is, it is opposite to a fictitious vertex y_∞ of the outer triangle $\mathscr{F}$. The edge region of e_i is denoted by $R_M(e_i)$ for $i = 1, 2, \ldots, d$, and the edge region of e_∞ is by $R_M(e_\infty)$. Observe that, since the outer triangle is an unbounded region, so is the edge regions associated with edges e_i, however the edge region of e_∞ is a regular bounded triangle. The interiors of these regions are disjoint. Figure 2.4(b) illustrates the edge regions in an acute inner triangle and in an outer triangle.

2.4.4 Face Regions in $\mathbb{R}^d$ for $d > 2$

The definitions of edge regions in $\mathbb{R}^2$ can be extended to the face regions in $\mathbb{R}^d$ for $d > 2$. A d-simplex is the smallest convex polytope in $\mathbb{R}^d$ constructed by a set of non-coplanar vertices $\mathcal{Y} = \{\mathsf{y}_1, \mathsf{y}_2, \cdots, \mathsf{y}_{d+1}\}$. The boundary of a d-simplex consists of k-simplices called k-faces for $0 \leq k < d$. Each k-face is a simplex defined by a subset of $\mathcal{Y}$ with k elements, hence there are $\binom{d+1}{k+1}$ k-faces in a d-simplex. Let $\mathfrak{S}(\mathcal{Y})$ be the d-simplex defined by the set of points $\mathcal{Y}$.

In $\mathbb{R}^d$ for $d > 2$, M-edge regions become M-face regions. Given a *simplex center* $M \in \mathfrak{S}(\mathcal{Y})^o$, there are $d+1$ M-face regions constructed by the set $\mathcal{Y}$. For $i = 1, \ldots, d+1$, let f_i denote the $(d-1)$-face opposite to the vertex y_i. The M-face region of the face f_i is denoted by $R_M(f_i)$ for $i = 1, 2, \cdots, d+1$. Let $\mathfrak{S} = \mathfrak{S}(\mathcal{Y})$ denote an inner simplex with vertex set $\mathcal{Y}$. Given a *simplex center* $M \in \mathfrak{S}^o$ (e.g. a triangle center in $\mathbb{R}^2$), there are $d+1$ M-face regions associated with each point of $\mathcal{Y}$. The M-face region of f_i is denoted by $R_M(f_i)$ for $i = 1, 2, \ldots, d+1$. The face region $R_M(f_i)$ is constructed by the points M and $\{\mathsf{y}_j\}_{j=1;j\neq i}^{d+1}$. Thus, a face region of a d-simplex is also a d-simplex. An illustration of the face region of f_3 for $\mathbb{R}^3$ is given in Figure 2.5(a). The face region $R_M(f_3)$ is the 3-simplex with points $\{\mathsf{y}_1, \mathsf{y}_3, \mathsf{y}_4, M\}$. Note that each 2-face of the 3-simplex (tetrahedron) is a 2-simplex (a triangle).

Theorem 2.4.4.1. *Let $\mathcal{Y} = \{\mathsf{y}_1, \mathsf{y}_2, \ldots, \mathsf{y}_{d+1}\} \subset \mathbb{R}^d$ be a set of non-coplanar points for $d > 0$, and let the set of face regions $\{R_M(f_i)\}_{i=1}^{d+1}$ partitions $\mathfrak{S}(\mathcal{Y})$. Hence, for $x, M \in \mathfrak{S}(\mathcal{Y})^o$, we have $x \in R_M(f_i)$ if and only if*

$$w_{\mathfrak{S}}^{(i)}(x) < \min_{\substack{j=1,\ldots,d+1\\ j\neq i}} \frac{m_i w_{\mathfrak{S}}^{(j)}(x)}{m_j}. \tag{2.9}$$

where $\mathbf{w}_{\mathfrak{S}}(x) = \left(w_{\mathfrak{S}}^{(1)}(x), \cdots, w_{\mathfrak{S}}^{(d+1)}(x)\right)$ and $\mathbf{m} = (m_1, \ldots, m_{d+1})$ are the barycentric coordinates of x and M with respect to $\mathfrak{S}(\mathcal{Y})$, respectively.

Proof: We prove this theorem by induction on dimension d. The proof of the case $d = 1$ is trivial. For $\mathfrak{S}(\mathcal{Y}) = (\mathsf{y}_1, \mathsf{y}_2) \subset \mathbb{R}$ and $\mathsf{y}_1 < \mathsf{y}_2$, the edge regions $R_M(e_1)$ and $R_M(e_2)$ are the intervals (M, y_2) and (y_1, M), respectively ($\{x = M\}$ and $\{x = \mathsf{y}_i\}$

have zero $\mathbb{R}$-Lebesgue measure). Note that, edge regions in $\mathbb{R}$ are also vertex regions. For $\alpha_1 \in (0,1)$ and $\alpha_2 = 1-\alpha_1$, let $\alpha_1\mathsf{y}_1 + \alpha_2\mathsf{y}_2$ be the convex (or barycentric) combination of $x \in \mathfrak{S}(\mathcal{Y})$. Hence, $x \in (M, \mathsf{y}_2) = R_M(e_1)$ if and only if $\alpha_1/\alpha_2 < m_1/m_2$. The case $d=2$ is proved in Proposition 2.4.3.1. Thus, there only remains the case $d>2$. We suppose the statement is true for all faces of the d-simplices which are $d-1$ dimensional, and by that, we will show that the statement is also true for the d-simplex which is d dimensional.

It is sufficient to show the result for f_1 (as the others follow by symmetry). Let $x \in R_M(f_1)$ and note that the elements of the set of $(d-1)$-faces, $\{f_j\}_{j=2}^{d+1}$, are adjacent to y_1. Each of these faces are of $d-1$ dimensions. Hence, they are $(d-1)$-simplices and they also have their own face regions. Thus, let R_{ij} be the face region of f_j with respect to $(d-1)$-simplex f_i for $j \neq i$. Note that M_i is the center of f_i. Now, let w_{ij} be the barycentric coordinate of point z corresponding to y_j with respect to the f_i. Observe that w_{ii} is not defined since y_i is not a vertex of the face f_i.

We follow the proof of Theorem 2.4.2.1. Let $\mathbf{m}' = (m'_1, \ldots, m'_{i-1}, m'_{i+1}, \ldots, m'_{d+1})$ be the set of barycentric coordinates of the center M_i with respect to f_i, and note that M_i is a linear combination of M and y_i. Hence, as in Theorem 2.4.2.1, $m'_1/m'_j = m_1/m_j$ for $j = 2,3,\ldots,d+1$ and $j \neq i$. By induction hypothesis, we observe that $z_i \in R_{i1}$ if and only if $w_{i1} < (m'_1/m'_j)w_{ij}$ if and only if $w_{i1} < (m_1/m_j)w_{ij}$ for $j = 2,3,\ldots,d+1$ and $j \neq i$. Since the point x is the convex (and linear) combination of z_i and y_i, for $\alpha \in (0,1)$, we have

$$x = (1-\alpha)\mathsf{y}_i + \alpha z_i = (1-\alpha)\mathsf{y}_i + \alpha\left(\sum_{k=1;k\neq i}^{d+1} w_{ik}\right).$$

By the uniqueness property of barycentric coordinates, it follows that $w_{\mathfrak{S}}^{(1)}(x) = \alpha w_{i1}$ and $w_{\mathfrak{S}}^{(j)}(x) = \alpha w_{ij}$. Hence,

$$\frac{w_{\mathfrak{S}}^{(1)}(x)}{w_{\mathfrak{S}}^{(j)}(x)} = \frac{w_{i1}}{w_{ij}} < \frac{m_1}{m_j}. \tag{2.10}$$

Since (2.10) is true for all $i = 2,\ldots,d+1$, we see that $x \in R_M(f_1)$ if and only if $w_{\mathfrak{S}}^{(1)}(x) < (m_1/m_i)w_{\mathfrak{S}}^{(i)}(x)$. Hence, the result follows. ■

For $M = M_C$, we simplify the result of Theorem 2.4.4.1; that is, for any point $x \in \mathfrak{S}(\mathcal{Y})^o$, we have $x \in R_{M_C}(e_i)$ if and only if $w_T^{(i)}(x) = \min_j w_T^{(j)}(x)$ since the barycentric coordinates of M_C is $\mathbf{m}_C = (1/(d+1), 1/(d+1), \ldots, 1/(d+1))$. The M_C-vertex regions are particularly appealing for our central-similarity proximity maps.

Let $\mathcal{F}$ be a facet of $C_H(\mathcal{X}_{1-j})$, and let $\mathcal{Y} = \{\mathsf{y}_1, \mathsf{y}_2 \ldots, \mathsf{y}_d\} \subset \mathbb{R}^d$ be a set of adjacent points on the boundary of $C_H(\mathcal{X}_{1-j})$ associated with $\mathcal{F}$. Let the outer simplex $\mathscr{F} \subset \mathbb{R}^d$ be defined by the set $\mathcal{Y}$ and the rays $\{\overrightarrow{C_M\mathsf{y}_1}, \ldots, \overrightarrow{C_M\mathsf{y}_d}\}$ for C_M being the median of $C_H(\mathcal{X}_{1-j})$. The face regions are constructed based on a point, or center, $M \in \mathscr{F}^o$. We define an additional line in the direction of $M - C_M$ to attain all face regions. We denote the facet of $C_H(\mathcal{X}_{1-j})$ given by the set $\mathcal{Y}$ as f_∞; that is, it is opposite to a fictitious vertex y_∞ of the outer simplex $\mathscr{F}$. The face region of f_i is denoted by $R_M(f_i)$ for $i = 1, 2, \ldots, d$, and the face region of f_∞ is by $R_M(f_\infty)$. The interiors of these regions are disjoint. Similar to the case when $d = 2$, the face region $R_M(f_\infty)$ is bounded whereas others are not, i.e. $R_M(f_i)$ for $i = 1, 2, \ldots, d$. Figure 2.4(b) illustrates the face regions of f_2 in an outer simplex. See that vertices $\{\mathsf{y}_1, \mathsf{y}_3\}$ and vectors $\{\overrightarrow{C_M\mathsf{y}_1}, \overrightarrow{C_M\mathsf{y}_3}\}$ establishes the unbounded face f_2. Hence the face region $R_M(f_2)$ is the unbounded region given by points $\{\mathsf{y}_1, \mathsf{y}_3, M\}$ and rays $\{\overrightarrow{C_M\mathsf{y}_1}, \overrightarrow{C_M\mathsf{y}_3}, \overrightarrow{MC_M}\}$.

2.5 Spatial Data Analysis and Ripley's K function

We use spatial data analysis tools to test if clusters exist in the domain. One such test, based on the Ripley's K function, exploits distances between points to determine the second-order moments of point processes in a window $\mathcal{W} \subset \Omega$. Human eye is proficient in detecting the second-order properties of spatial objects in 2 dimensional (or in 3 dimensional) domains (Julesz, 1975). Moreover, Ripley (1976) states that the intensity λ and the function K sufficiently summarize first and second-order moments of stationary point processes, respectively, which can be used to devise tests for analyzing spatial data sets. We offer algorithms that employs both CCDs and Ripley's K function to estimate the supports of individual hidden classes in a data

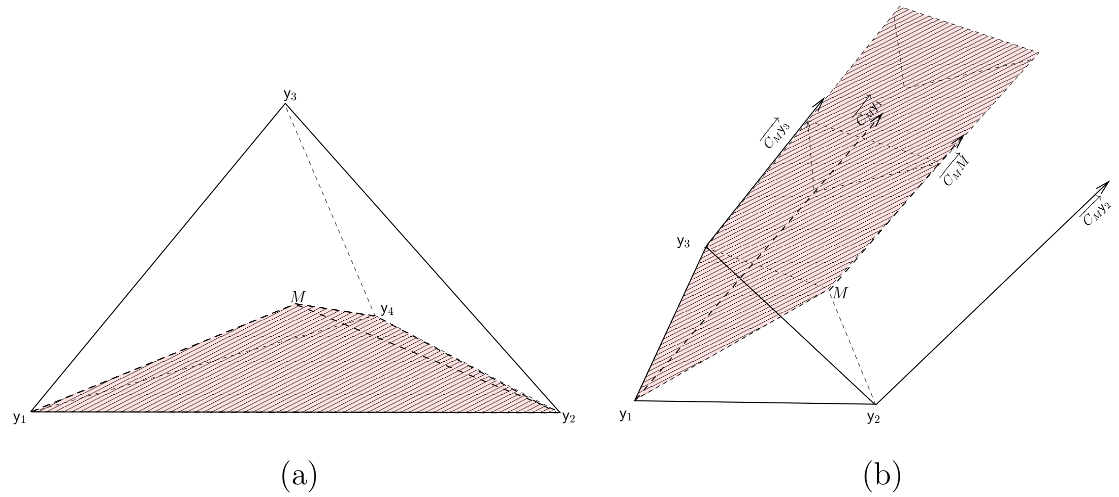


Figure 2.5: (a) Face region $R_M(f_3)$ of face f_3 of a 3-simplex, or a tetrahedron. (b) Face region $R_M(f_2)$ of a face f_2 of a outer simplex. Face regions are shaded.

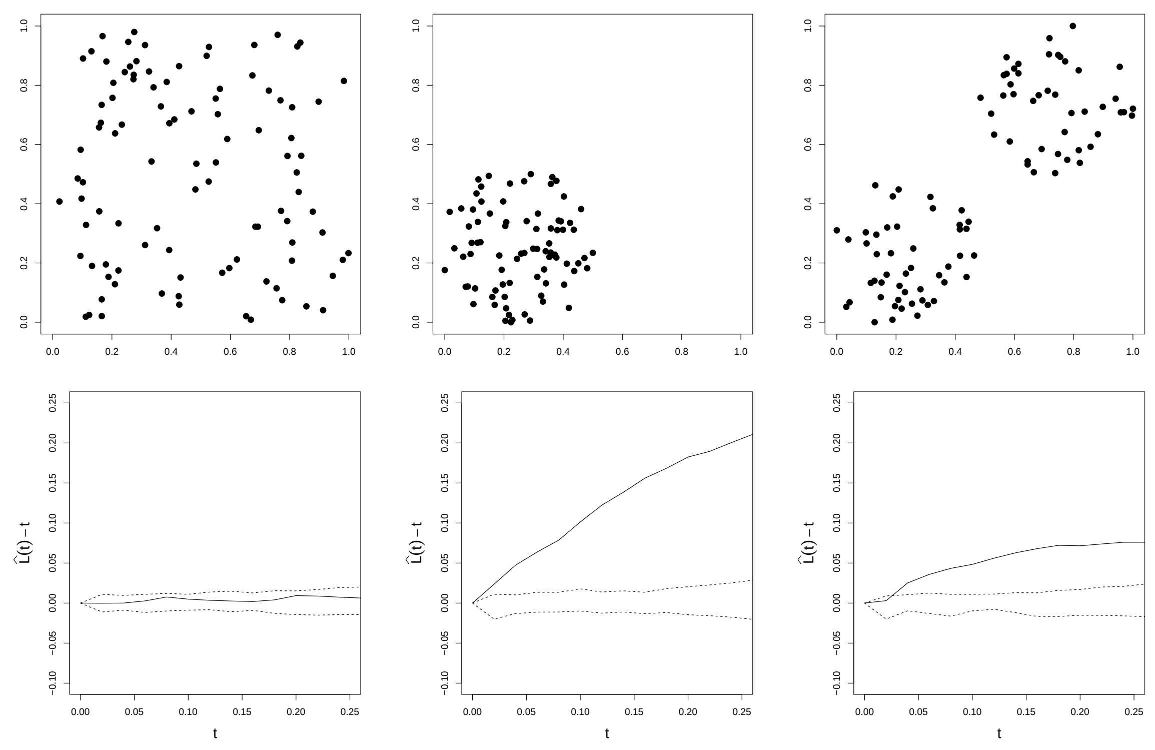
set. Here, we review the K function and methods for testing spatial clusterings for the sake of completeness.

Mapped point patterns recieved extensive attention in the spatial data analysis literature. Methods based on quadrats and nearest neighbor distances are some of the most popular methods (Ripley, 2005). In addition, Ripley (1977) showed that the second-order moments of homogeneous spatial processes could be reduced to a function $K(t)$ on distance $t \in \mathbb{R}_+$; that is, the distances between points are exploited to test whether a set of random events fit a certain spatial distribution. Now, let λ be the expected number of events in a region $A \subset \Omega$ of unit area (or volume). Hence, $\lambda^2 K(t)$ is the expected number of ordered pairs at most t distance apart.

Let $\mathcal{Z} = \{Z_1, Z_2, \cdots, Z_n\}$ be a set of $\mathbb{R}^d$-valued random events in a *window* $\mathcal{W} \subset \mathbb{R}^d$. We use the estimate $\hat{K}(t)$ of the expected number of pairs that are at most t distance apart from each other in $\mathcal{W}$ (Ripley, 1976):

$$\hat{K}(t) := \frac{\text{Vol}(A)}{n(n-1)} \sum_{z \in \mathcal{Z}} \sum_{\substack{z' \in \mathcal{Z} \\ z' \neq z}} I(d(z, z') < t) \vartheta(z, z'). \tag{2.11}$$

Here, $\vartheta(z, z')$ is the weighting function to correct the bias resulted by the edge effects. Specifically, the neighbourhood of z with radius t is a ball $B(z, t)$ which may



(a) (b) (c)

Figure 2.6: Three examples of point patterns where (a) one is a realization of homogeneous Poisson process, (b) one is a clustered process with one cluster, and the other is (c) a process with two clusters. The realizations are given on top, and the corresponding %99 envelopes and the $\hat{L}(t) - t$ are given at the bottom.

sufficiently be close to the edge of the window $\mathcal{W}$ such that $B(z,t) \setminus \mathcal{W} \neq \emptyset$. Since $\mathcal{Z} \subset \mathcal{W}$, some subset of $B(z,t)$ would be empty, and thus $\hat{K}(t)$ in Equation 2.11 will be a biased estimator of $K(t)$ if $\vartheta(z,z') = 1$ for all $z, z' \in \mathcal{W}$. Hence, we choose a such a correction coefficient $\vartheta(z,z')$ for any $z \in \mathcal{Z}$ that is proportional to the volume of $B(z,t) \cap \mathcal{W}$. We use the translation correction method for our clustering algorithms (Baddeley et al., 2015). Although $\hat{K}(t)$ can be computed for any value of $t > 0$, an appropriate $t_{max} \geq t$ value is in order since high values of t may increase the variance of $\hat{K}(t)$. Hence, we restrict the t_{max} depending on the window geometry; for example, a t_{max} value equal to a quarter of the smaller side length of a rectangular window is appropriate (Baddeley et al., 2015).

We test if the realizations of these random events are drawn from events of homogenuous Poisson processes (or from events of any spatial distribution; but we assume complete spatial randomness). However, the distribution of $\hat{K}(t)$ is unknown. Under the assumption of complete spatial randomness (CSR), it is known that $K(t) = \pi t^2$ for $\mathbb{R}^2$, i.e. the area/volume of a ball with radius t. To test the null hypothesis, we use the simulated data sets to build confidence bands with minimum and maximum $\hat{K}(t)$ values. Each simulated data set is composed of a random sample of uniformly distributed points inside $\mathcal{W}$. We record minimum and maximum $\hat{K}(t)$ for each value of t. These values are equivalent to α'th and $1-\alpha$'th quantiles, respectively, which establish %95 confidence bands with $N = 19$ Monte Carlo replications and %99 confidence bands with $N = 99$ replications. Hence, the more the number of replications N, the more powerful the test is. We test if the $\hat{K}(t)$ of the data set is equal or greater(less) than the $N(1-\alpha/2)$'th ($N(\alpha/2)$'th) value of all $\hat{K}(t)$ values (from both simulated data sets and the real data set). If this is the case for any value of t, the null hypothesis is rejected. Another expression, $L(t) - t$, on t provides a better visualization of the envelope which is defined as follows: for $\mathcal{Z} \subset \mathbb{R}^2$ (Ripley, 1979),

$$L(t) := \sqrt{\frac{K(t)}{\pi}} \tag{2.12}$$

The estimate $\hat{L}(t)$ is defined similarly. It is easy to see that, for $K(t) = \pi t^2$, we have $L(t) - t = 0$. Hence, the envelope of $L(t) - t$ provides confidence bands around the 0 line which is visually more convenient. In Figure 2.6, we illustrate three example sets of point patterns in a unit square window $\mathcal{W}$ that one is drawn from a homogeneous point pattern, and others are clustered processes (where some regions in the window either have different local density or zero Lebesgue measure). We also provide the $\hat{L}(t) - t$ curves.

In Figure 2.6, upper and lower dashed curves of the envelopes represent the maximum and minimum possible values of $\hat{L}(t) - t$ corresponding to each value of t. If the $\hat{L}(t) - t$ curve of the data set is not (entirely) inside the envelope, it is concluded that there is no sufficient evidence of the data set being drawn from a homogeneous

Poisson process (the data set does not show CSR). However, as illustrated with a window of one or two clusters in Figure 2.6, the curve of the data set is outside the envelopes which indicates the rejection of the null hypothesis. We use $\hat{L}(t) - t$ to find the radii or the proximity regions that encapsulate subsets of the data sets which follow CSR inside the covering ball. We establish a collection of subsets given by these regions (viewed as a window $\mathcal{W}$) such that we use the union of such collection of regions to estimate the supports of hidden classes.

Chapter 3

PROXIMITY MAPS AND PROXIMITY CATCH DIGRAPHS

3.1 *Introduction*

The *proximity map* $\mathcal{N}(\cdot) : \Omega \to 2^{\Omega}$ associates with each point $x \in \mathcal{X}$, a *proximity region* $\mathcal{N}(x) \subset \Omega$. Consider the data-random (or vertex-random) proximity catch digraph (PCD) $D = (\mathcal{V}, \mathcal{A})$ with vertex set $\mathcal{V} = \mathcal{X}$ and arc set $\mathcal{A}$ defined by $(u, v) \in \mathcal{A} \iff \{u, v\} \subset \mathcal{X}$ and $v \in \mathcal{N}(u)$. The digraph D depends on the (joint) distribution of $\mathcal{X}$, and on the map $\mathcal{N}(\cdot)$. The adjective *proximity* — for the digraph D and for the map $\mathcal{N}(\cdot)$ — comes from thinking of the region $\mathcal{N}(x)$ as representing those points in Ω "close" to x (Jaromczyk and Toussaint, 1992; Toussaint, 1980). The binary relation $u \sim v$, which is defined as $v \in \mathcal{N}(u)$, is asymmetric, thus the adjacency of u and v is represented with directed edges or arcs which yield a digraph instead of a graph. Let us also consider the case where the data set $\mathcal{X}$ is composed of two non-empty sets, $\mathcal{X}_0$ and $\mathcal{X}_1$, with sample sizes $n_0 := |\mathcal{X}_0|$ and $n_1 := |\mathcal{X}_1|$, respectively. For $j = 0, 1$, we investigate the PCD D_j, associated with $\mathcal{X}_j$ against $\mathcal{X}_{1-j}$. Here, we specify $\mathcal{X}_j$ as the target class and $\mathcal{X}_{1-j}$ as the non-target class. Hence, in the definitions of our PCDs, the only difference is switching the roles of $\mathcal{X}_0$ and $\mathcal{X}_1$. For $j = 0$, $\mathcal{X}_0$ becomes the target class, and for $j = 1$, $\mathcal{X}_1$ becomes the target class.

CCCDs are, in fact, equivalent to PCDs whose proximity maps constitutes spheres in $\mathbb{R}^d$, namely spherical proximity maps. In addition to CCCDs, Ceyhan (2005) introduced three families of PCDs to analytically compute the distribution of the domination number of such digraphs in a two class setting. Domination number and, another graph invariant, the arc density (the ratio of number of arcs in a digraph to the total number of arcs possible) of these PCDs have been used for testing spatial patterns of segregation and association (Ceyhan and Priebe, 2005; Ceyhan et al., 2007, 2006).

Two of the proximity maps associated with PCDs introduced by Ceyhan and Priebe (2005) are *simplicial* proximity maps (establishes regions that constitute simplices in $\mathbb{R}^d$) defined for the points of the target class $\mathcal{X}_j$ in the convex hull of the non-target class, $C_H(\mathcal{X}_{1-j})$. These simplicial proximity maps are, namely, *proportional-edge* and *central-similarity* proximity maps. However, by introducing the *outer simplices* associated with the facets of $C_H(\mathcal{X}_{1-j})$, we extend the definition of the simplicial proximity maps to $\mathbb{R}^d \setminus C_H(\mathcal{X}_{1-j})$. Simplicial proximity regions are d-simplices in $C_H(\mathcal{X}_{1-j})$ (triangles in $\mathbb{R}^2$ and tetrahedrons in $\mathbb{R}^3$) and d-polytopes in $\mathbb{R}^d \setminus C_H(\mathcal{X}_{1-j})$. After partitioning $\mathbb{R}^d$ into disjoint regions, we further partition each simplex $\mathfrak{S}$ and polytope $\mathscr{F}$ into vertex and face regions, and construct the simplicial proximity regions $\mathcal{N}(x)$ for $x \in \mathfrak{S}$. Here, we define the regions $\mathcal{N}(x)$ as open sets in $\mathbb{R}^d$.

3.2 Spherical Proximity Maps

Class Cover Catch Digraphs (CCCDs) are graph theoretic representations for the CCP (Priebe et al., 2001, 2003a). The goal in CCP is to find a region that encapsulates all the members of the class of interest where this particular region can be viewed as a *cover*. Let $\mathcal{X}_0 = \{x_1, x_2, ..., x_{n_0}\} \subset \mathbb{R}^d$ and $\mathcal{X}_1 = \{y_1, y_2, ..., y_{n_1}\} \subset \mathbb{R}^d$ be sets of observations from two classes of a data set. Without loss of generality, assume that the target class (i.e. the class of interest) is $\mathcal{X}_0$. In a CCCD, for $u, v \in \mathcal{X}_0$, let $B(u, r)$ be the hyperball centered at u with radius $r = r(u)$. A CCCD D_0 is a digraph $D_0 = (\mathcal{V}_0, \mathcal{A}_0)$ with vertex set $\mathcal{V}_0 = \mathcal{X}_0$ and the arc set $\mathcal{A}_0$ where $(u, v) \in \mathcal{A}_0$ iff $v \in B(u, r)$. CCCDs are a family of PCDs using *spherical* proximity maps, letting $\mathcal{N}(x) := B(x, r(x))$. One particular family of CCCDs are called pure-CCCDs (P-CCCDs) wherein the *covering* ball (or covering hyperball) $B(u, r)$ is viewed as a region, with u as its center, expanding until it hits the closest non-target class point. Therefore, $r = r(u) := \min_{v \in \mathcal{X}_1} d(u, v)$ (Marchette, 2010). Here, $d(.,.)$ can be any dissimilarity measure but we use the Euclidean distance henceforth. For all $u \in \mathcal{X}_0$, the radius r is defined in such a way that no non-target point of $\mathcal{X}_1$ is in the covering ball $B(u, r)$, i.e. $\mathcal{X}_1 \cap B(u, r) = \emptyset$. The digraph D_1 is established by interchanging the roles of target class being $\mathcal{X}_1$ and non-target class being $\mathcal{X}_0$.

3.3 Proportional-Edge Proximity Maps

We use a type of proximity map with expansion parameter r, namely *proportional-edge* (PE) proximity map, denoted by $\mathcal{N}_{PE}(\cdot, r)$. The PE proximity map and the associated digraphs, PE-PCDs, are defined in Ceyhan and Priebe (2005). Currently, PE-PCDs are only defined for the points in $\mathcal{X}_j \cap C_H(\mathcal{X}_{1-j})$. Hence, for the remaining points of the target class $\mathcal{X}_j$, i.e. $\mathcal{X}_j \setminus C_H(\mathcal{X}_{1-j})$, we extend the definition of PE proximity maps to the outer simplices. Hence, we will be able to show later that the resulting PCDs have computationally tractable minimum dominating sets which are equivalent to the exact minimum prototype sets of PE-PCD classifiers for the entire data set.

3.3.1 PE proximity Maps of d-Simplices

For $r \in [1, \infty)$, we define $\mathcal{N}_{PE}(\cdot, r)$ to be the PE proximity map associated with a triangle $T = T(\mathcal{Y})$ formed by the set of non-collinear points $\mathcal{Y} = \{\mathsf{y}_1, \mathsf{y}_2, \mathsf{y}_3\} \subset \mathbb{R}^2$. Let $R_{M_C}(\mathsf{y}_1)$, $R_{M_C}(\mathsf{y}_2)$ and $R_{M_C}(\mathsf{y}_3)$ be the vertex regions associated with vertices y_1,y_2 and y_3. Note that the barycentric coordinates of M_C are $(1/3 : 1/3 : 1/3)$. For $x \in T^o$, let $v(x) \in \mathcal{Y}$ be the vertex whose region contains x; hence $x \in R_{M_C}(v(x))$. If x falls on the boundary of two vertex regions, or on M_C, we assign $v(x)$ arbitrarily. Let $e(x)$ be the edge of T opposite to $v(x)$. Let $\ell(v(x), x)$ be the line parallel to $e(x)$ through x. Let $d(v(x), \ell(v(x), x))$ be the Euclidean (perpendicular) distance from $v(x)$ to $\ell(v(x), x)$. For $r \in [1, \infty)$, let $\ell_r(v(x), x)$ be the line parallel to $e(x)$ such that $d(v(x), \ell_r(v(x), x)) = r d(v(x), \ell(v(x), x))$. Let $T_r(x)$ be the triangle similar to and with the same orientation as T where $T_r(x)$ has $v(x)$ as a vertex and $\ell_r(v(x), x)$ as edge opposite of $v(x)$. Then the *proportional-edge* proximity region $\mathcal{N}_{PE}(x, r)$ is

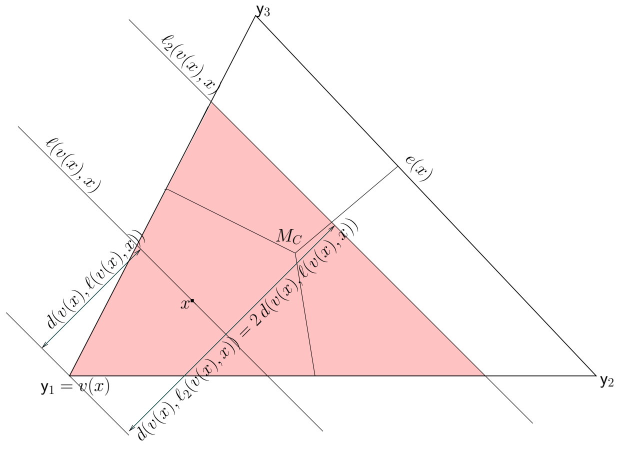
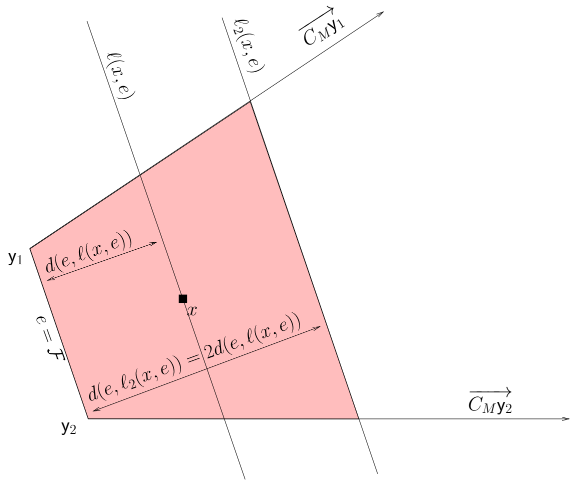


Figure 3.1: The proportional-edge proximity region (shaded), $\mathcal{N}_{PE}(x, r = 2)$, in a triangle $T \subseteq \mathbb{R}^2$.

defined to be $T_r(x) \cap T$. Figure 3.1 illustrates a PE proximity region $\mathcal{N}_{PE}(x,r)$ of a point x in an acute triangle.

The extension of $\mathcal{N}_{PE}(\cdot, r)$ to $\mathbb{R}^d$ for $d > 2$ is straightforward. Now, let $\mathcal{Y} = \{\mathsf{y}_1, \mathsf{y}_2, \cdots, \mathsf{y}_{d+1}\}$ be a set of $d+1$ non-coplanar points, and represent the simplex formed by the these points as $\mathfrak{S} = \mathfrak{S}(\mathcal{Y})$. We define the PE proximity map as follows. Given a point $x \in \mathfrak{S}^o$, let $v(x)$ be the vertex in whose region x falls (if x falls on the boundary of two vertex regions or on M_C, we assign $v(x)$ arbitrarily.) Let $\varphi(x)$ be the face opposite to vertex $v(x)$, and $\eta(v(x), x)$ be the hyperplane parallel to $\varphi(x)$ which contains x. Let $d(v(x), \eta(v(x), x))$ be the (perpendicular) Euclidean distance from $v(x)$ to $\eta(v(x), x)$. For $r \in [1, \infty)$, let $\eta_r(v(x), x)$ be the hyperplane parallel to $\varphi(x)$ such that $d(v(x), \eta_r(v(x), x)) = r\, d(v(x), \eta(v(x), x))$. Let $\mathfrak{S}_r(x)$ be the polytope similar to and with the same orientation as $\mathfrak{S}$ having $v(x)$ as a vertex and $\eta_r(v(x), x)$ as the opposite face. Then the PE proximity region is given by $\mathcal{N}_{PE}(x, r) := \mathfrak{S}_r(x) \cap \mathfrak{S}$.

We consider the Delaunay tessellation (assumed to exist) of $\mathcal{X}_{1-j}$ where $\mathcal{S}^{(1)}_{1-j} = \{\mathfrak{S}_1, \ldots, \mathfrak{S}_K\}$ denotes the set of all Delaunay cells (which are d-simplices). We construct the proximity region $\mathcal{N}_{PE}(x, r)$ of a point $x \in \mathcal{X}_j$ depending on which d-simplex $\mathfrak{S}_k$ this point reside in. Observe that, this construction pertains to points in $\mathcal{X}_j \cap C_H(\mathcal{X}_{1-j})$ only.



$\ell(x,e)$
$\ell_2(x,e)$
$\overrightarrow{C_M\mathsf{y}_1}$
y_1
$d(e,\ell(x,e))$
x
$e=\mathcal{F}$
$d(e,\ell_2(x,e))=2d(e,\ell(x,e))$
$\overrightarrow{C_M\mathsf{y}_2}$
y_2

Figure 3.2: The proportional-edge proximity region, $\mathcal{N}_{PE}(x, r = 2)$ (shaded), in an outer triangle $\mathscr{F} \subseteq \mathbb{R}^2$.

3.3.2 PE proximity Maps of Outer Simplices

For points of the target class $\mathcal{X}_j$ outside of the convex hull of the non-target class $\mathcal{X}_{1-j}$, i.e. $\mathcal{X}_j \setminus C_H(\mathcal{X}_{1-j})$, we define the PE proximity maps similar to the ones defined for d-simplices. Let $\mathscr{F} \subset \mathbb{R}^2$ be an outer triangle defined by the adjacent boundary points $\{\mathsf{y}_1, \mathsf{y}_2\} \subset \mathbb{R}^2$ of $C_H(\mathcal{X}_{1-j})$ and by rays $\overrightarrow{C_M\mathsf{y}_1}$ and $\overrightarrow{C_M\mathsf{y}_2}$ for C_M being the median of the boundary points of $C_H(\mathcal{X}_{1-j})$. Also, let $e = \mathcal{F}$ be the edge (or facet) of $C_H(\mathcal{X}_{1-j})$ adjacent to vertices $\{\mathsf{y}_1, \mathsf{y}_2\}$. Note that there is no center in an outer triangle, but there is only a single vertex region. For $r \in [1, \infty)$, we define $\mathcal{N}_{PE}(\cdot, r)$ to be the PE proximity map of the outer triangle. For $x \in \mathscr{F}^o$, let $\ell(x, e)$ be the line parallel to e through x, and let $d(e, \ell(x, e))$ be the Euclidean distance from e to $\ell(x, e)$. For $r \in [1, \infty)$, let $\ell_r(x, e)$ be the line parallel to e such that $d(e, \ell_r(x, e)) = rd(e, \ell(x, e))$. Let $\mathscr{F}_r(x)$ be a polygon similar to the outer triangle $\mathscr{F}$ such that $\mathscr{F}_r(x)$ has e and $e_r(x) = \ell_r(x, e) \cap \mathscr{F}$ as its two edges, however $\mathscr{F}_r(x)$ is a bounded region whereas $\mathscr{F}$ is not. Then, the proximity region $\mathcal{N}_{PE}(x, r)$ is defined to be $\mathscr{F}_r(x)$. Figure 3.2 illustrates a PE proximity region $\mathcal{N}_{PE}(x, r)$ of a point x in an outer triangle.

The extension of $\mathcal{N}_{PE}(\cdot, r)$ of outer triangles to $\mathbb{R}^d$ for $d > 2$ is straightforward.

Let $\mathscr{F} \subset \mathbb{R}^d$ be an outer simplex defined by a set of adjacent points $\{\mathsf{y}_1, \dots, \mathsf{y}_d\} \subset \mathbb{R}^d$ on the boundary of $C_H(\mathcal{X}_{1-j})$, and let $\mathscr{F}$ be defined by rays $\{\overrightarrow{C_M \mathsf{y}_1}, \dots, \overrightarrow{C_M \mathsf{y}_d}\}$. Also, let $\mathcal{F}$ be the facet of $C_H(\mathcal{X}_{1-j})$ adjacent to vertices $\{\mathsf{y}_1, \dots, \mathsf{y}_d\}$. We define the PE proximity map as follows. Given a point $x \in \mathscr{F}^o$, let $\eta(x, \mathcal{F})$ be the hyperplane parallel to $\mathcal{F}$ through x and let $d(\mathcal{F}, \eta(x, \mathcal{F}))$ be the Euclidean distance from $\mathcal{F}$ to $\eta(x, \mathcal{F})$. For $r \in [1, \infty)$, let $\eta_r(x, \mathcal{F})$ be the hyperplane parallel to $\mathcal{F}$ such that $d(\mathcal{F}, \eta_r(x, \mathcal{F})) = rd(\mathcal{F}, \eta(x, \mathcal{F}))$. Let $\mathscr{F}_r(x)$ be the polytope similar to the outer simplex $\mathscr{F}$ such that $\mathscr{F}_r(x)$ has $\mathcal{F}$ and $\mathcal{F}_r(x) = \eta_r(x) \cap \mathscr{F}$ as its two faces. Then, the proximity region $\mathcal{N}_{PE}(x, r)$ is defined to be $\mathscr{F}_r(x)$.

The convex hull $C_H(\mathcal{X}_{1-j})$ has at least $d+1$ facets (exactly $d+1$ when $n_{1-j} = d+1$), and since each outer simplex is associated with a facet, the number of outer simplices is at least $d+1$. Let $\mathcal{S}^{(2)}_{1-j} = \{\mathscr{F}_1, \dots, \mathscr{F}_L\}$ denotes the set of all outer simplices. This construction handles the points in $\mathcal{X}_j \setminus C_H(\mathcal{X}_{1-j})$ only. Together with the points inside $C_H(\mathcal{X}_{1-j})$, the PE-PCD D_j, whose vertex set is $\mathcal{V}_j = \mathcal{X}_j$, has at least

$$\sum_{k=1}^{K} I(\mathcal{X}_j \cap \mathfrak{S}_k \neq \emptyset) + \sum_{l=1}^{L} I(\mathcal{X}_j \cap \mathscr{F}_l \neq \emptyset)$$

many disconnected components.

3.4 Central-Similarity Proximity Maps

We use a type of proximity map with expansion parameter τ, namely *central-similarity* (CS) proximity map, denoted by $\mathcal{N}_{CS}(\cdot, \tau)$. The CS proximity map and the associated digraphs, CS-PCDs, are defined in Ceyhan and Priebe (2005). Currently, CS-PCDs are only defined for the points in $\mathcal{X}_j \cap C_H(\mathcal{X}_{1-j})$. Hence, for the remaining points of the target class $\mathcal{X}_j$, i.e. $\mathcal{X}_j \setminus C_H(\mathcal{X}_{1-j})$, we extend the definition of CS proximity maps to the outer simplices. However, if a point x is outside of $C_H(\mathcal{X}_{1-j})$, the point is in an outer simplex. In that case, the region $\mathcal{N}(x)$ is "similar" to this outer simplex which is an unbounded convex polytope. Although the construction of proximity regions differ in inside and outside of the convex hull, the proximity regions are associated with the face (edges in $\mathbb{R}^2$) regions.

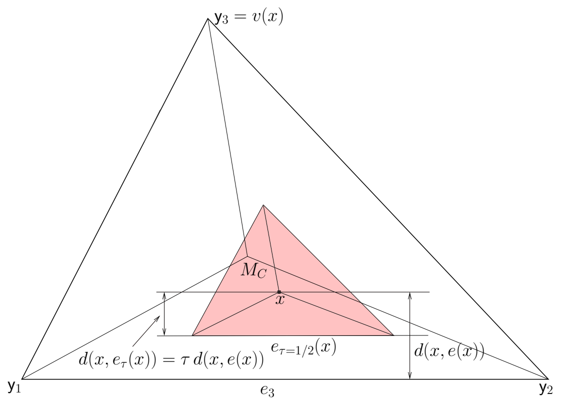


Figure 3.3: The central-similarity proximity region (shaded) of an acute triangle , $N_{CS}(x, \tau = 1/2)$ in $\mathbb{R}^2$.

3.4.1 CS proximity Maps for d-simplices

For $\tau \in (0, \infty)$, we define $\mathcal{N}_{CS}(\cdot, \tau)$ to be the CS proximity map associated with a triangle $T = T(\mathcal{Y})$ formed by the set of non-collinear points $\mathcal{Y} = \{y_1, y_2, y_3\} \subset \mathbb{R}^2$. Let $R_{M_C}(e_1)$, $R_{M_C}(e_2)$ and $R_{M_C}(e_3)$ be the edge regions associated with edge e_1,e_2 and e_3. Note that the barycentric coordinates of M_C are $(1/3 : 1/3 : 1/3)$. For $x \in T^o$, let $e(x)$ be the edge whose region contains x; hence $x \in R(e(x))$. If x falls on the boundary of two edge regions, or at the center of mass, we assign $e(x)$ arbitrarily. Let $d(x, e(x))$ be the Euclidean distance from x to the edge $e(x)$. For $\tau \in (0, \infty)$, the triangle $T_\tau(x)$ has an edge $e_\tau(x)$ parallel to $e(x)$ such that $d(x, e_\tau(x)) = \tau d(x, e(x))$. Here, $d(e_\tau(x), e(x)) \leq d(x, e(x))$ for $\tau \in (0, 1]$, and $d(e_\tau(x), e(x)) > d(x, e_\tau(x))$ for $\tau > 1$. The triangle $T_\tau(x)$ has the same orientation as T, and the points x is the center of mass of $T_\tau(x)$. Hence, the proximity region $N_{CS}(x, \tau)$ is defined to be $T_\tau(x) \cap T$. Figure 3.3 illustrates the proximity region $N_{CS}(x, \tau)$. Observe that $T_\tau(x) \subseteq T$ if $\tau \in (0, 1]$.

The extension of $N_{CS}(\cdot, \tau)$ to $\mathbb{R}^d$ for $d > 2$ is straightforward. Now, let $\mathcal{Y} = \{y_1, y_2, \cdots, y_{d+1}\}$ be a set of $d + 1$ non-coplanar points, and represent the simplex formed by the these points as $\mathfrak{S} = \mathfrak{S}(\mathcal{Y})$. We define the central-similarity proximity map as follows. Let $f(x)$ be the face whose region x falls If x falls on the boundary of two face regions or at the center of mass, we assign $f(x)$ arbitrarily. Let $d(x, f(x))$ be the Euclidean distance from x to $f(x)$. For $\tau \in (0, \infty]$, the simplex $\mathfrak{S}_\tau(x)$ has a face $f_\tau(x)$ parallel to $f(x)$ such that $d(x, f_\tau(x)) = \tau d(x, f(x))$. The simplex $\mathfrak{S}_\tau(x)$ has the same orientation as $\mathfrak{S}$, and the points x is the center of mass of $\mathfrak{S}_\tau(x)$. Then the central-similarity proximity region is given by $N_{CS}(x, \tau) := \mathfrak{S}_\tau(x) \cap \mathfrak{S}$.

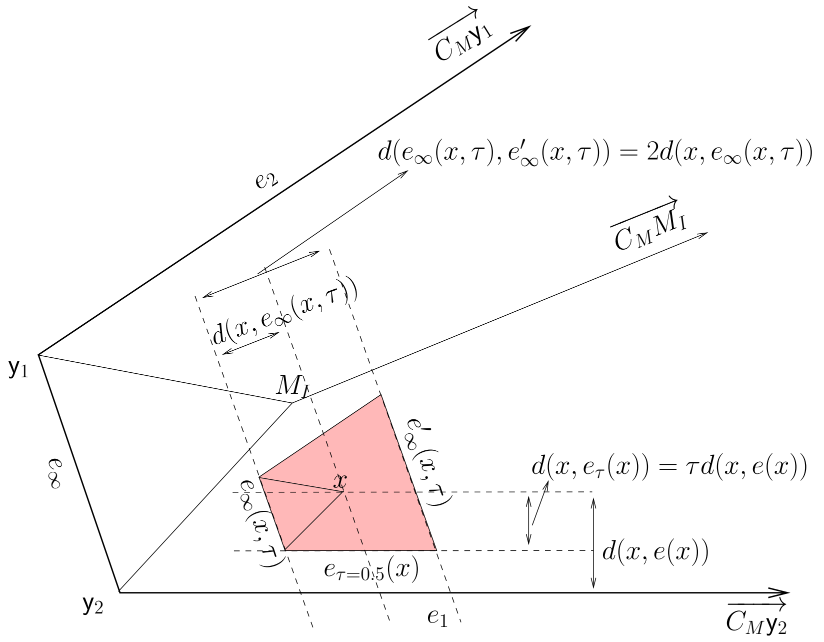
So far, we assumed $n_{1-j} = d + 1$ in the definitions for simplicity. If $n_{1-j} > d + 1$ then we consider the Delaunay tessellation (assumed to exist) of $\mathcal{X}_{1-j}$ where $\mathcal{S}_{1-j} = \{\mathfrak{S}_1, \ldots, \mathfrak{S}_K\}$ denotes the set of all Delaunay cells, and $m_{1-j} = |\{\mathfrak{S} \in \mathcal{S}_{1-j} : \mathfrak{S} \cap \mathcal{X}_{1-j} \neq \emptyset\}|$ denotes the number of triangles having vertices in $\mathcal{X}_{1-j}$. Observe that, this construction of D_j deals with points in $\mathcal{X}_j \cap C_H(\mathcal{X}_{1-j})$ only. For these points, the digraph has at least $\sum_{k=1}^{K} I(\mathcal{X}_j \cap \mathfrak{S}_k \neq \emptyset)$ components.

3.4.2 CS proximity Maps for Outer Simplices

For points of the target class $\mathcal{X}_j$ outside of the convex hull of the non-target class $\mathcal{X}_{1-j}$, i.e. $\mathcal{X}_j \setminus C_H(\mathcal{X}_{1-j})$, we define the central-similarity proximity maps similar to the ones in d-simplices. Let $\mathscr{F} \subset \mathbb{R}^2$ be an outer triangle defined by the adjacent boundary points $\{\mathsf{y}_1, \mathsf{y}_2\} \subset \mathbb{R}^2$ of $C_H(\mathcal{X}_{1-j})$. Let the center of $\mathscr{F}$ be $M = M_I$, the point equally distant to all edges of outer triangle $\mathscr{F}$, namely the *incenter*.

For $\tau \in (0, \infty)$, we define $N_{CS}(\cdot, \tau)$ to be the central-similarity proximity map. Let $R_{M_I}(e_1)$, $R_{M_I}(e_2)$ and $R_{M_I}(e_\infty 3)$ be the edge regions associated with edge e_1,e_2 and e_∞ using line segments from the incenter M_I to the vertices y_j, and the ray $\overrightarrow{C_M M_I}$. For $x \in T^o$, let $e(x)$ be the edge whose region contains x; hence $x \in R_{M_I}(e(x))$. If x falls on the boundary of two edge regions, or at M_I, we assign $e(x)$ arbitrarily. Let $d(x, e(x))$ be the Euclidean distance from x to the edge $e(x)$. For $\tau \in (0, \infty)$, the convex polytope $\mathscr{F}_\tau(x)$ has an edge $e_\tau(x)$ parallel to $e(x)$ such that $d(x, e_\tau(x)) = \tau d(x, e(x))$. Here, $d(e_\tau(x), e(x)) \leq d(x, e(x))$ for $\tau \in (0, 1]$, and $d(e_\tau(x), e(x)) > d(x, e_\tau(x))$ for $\tau > 1$. The convex polytope $\mathscr{F}_\tau(x)$ has the same orientation as $\mathscr{F}$; that is,

(i) $\mathscr{F}_\tau(x)$ has an edge $e_\infty(x, \tau)$ parallel to e_∞, and has edges $e_j(x, \tau)$ parallel to e_j



$\overrightarrow{C_M \mathsf{y}_1}$
$d(e_\infty(x,\tau), e'_\infty(x,\tau)) = 2d(x, e_\infty(x,\tau))$
e_2
$\overrightarrow{C_M M_I}$
$d(x, e_\infty(x,\tau))$
y_1
M_I
e_∞
$e'_\infty(x,\tau)$
$d(x, e_\tau(x)) = \tau d(x, e(x))$
x
$e_\infty(x,\tau)$
$d(x, e(x))$
$e_{\tau=0.5}(x)$
y_2
e_1
$\overrightarrow{C_M \mathsf{y}_2}$

Figure 3.4: The central-similarity proximity region (shaded) of an outer simplex, $N_{CS}(x, \tau = 0.5)$ in $\mathbb{R}^2$.

for $j = 1, 2$. The polytope $\mathscr{F}_\tau(x)$ has an additional edge $e'(x,\tau)$ parallel to e_∞ such that $d(e'(x,\tau), e_\infty(x,\tau)) = 2 * d(x, e_\infty(x,\tau))$;

(ii) The point x is as the same type of center of $\mathscr{F}_\tau(x)$ as M_I of $\mathscr{F}$.

Note that $\mathscr{F}_\tau(x)$ is bounded unlike $\mathscr{F}$. Hence, the proximity region $N_{CS}(x,\tau)$ is defined to be $\mathscr{F}_\tau(x) \cap \mathscr{F}$. Figure 3.4 illustrates the proximity region $N_{CS}(x,\tau)$. Observe that $\mathscr{F}_\tau(x) \subseteq \mathscr{F}$ if $\tau \in (0, 1]$.

The extension of $\mathcal{N}_{CS}(\cdot, \tau)$ of outer triangles to $\mathbb{R}^d$ for $d > 2$ is straightforward. Let $\mathscr{F} = \mathscr{F}(\mathcal{Y})$ be an outer simplex defined by the adjacent boundary points $\{\mathsf{y}_1, \ldots, \mathsf{y}_d\} \subset \mathbb{R}^d$ of $C_H(\mathcal{X}_{1-j})$ and rays $\{\overrightarrow{C_M \mathsf{y}_1}, \ldots, \overrightarrow{C_M \mathsf{y}_d}\}$. Let $f(x)$ be the face whose region x falls. If x falls on the boundary of two face regions or at the center of mass, we assign $f(x)$ arbitrarily. Let $d(x, f(x))$ be the Euclidean distance from x to $f(x)$. For $\tau \in (0, \infty]$, the simplex $\mathscr{F}_\tau(x)$ has a face $f_\tau(x)$ parallel to $f(x)$ such that $d(x, f_\tau(x)) = \tau d(x, f(x))$. The convex polytope $\mathscr{F}_\tau(x)$ has the same orientation as $\mathscr{F}$ such that $\mathscr{F}_\tau(x)$ has a face $f_\infty(x,\tau)$ parallel to f_∞, and has face $f_j(x,\tau)$ parallel to f_j for $j = 1, 2, \ldots, d$. The polytope $\mathscr{F}_\tau(x)$ has an additional face $f'(x,\tau)$ parallel to f_∞ such that $d(f'(x,\tau), f_\infty(x,\tau)) = 2 * d(x, f_\infty(x,\tau))$. Moreover, the point x is as th same of center of $\mathscr{F}_\tau(x)$ as M_I for $\mathscr{F}$. Note that the polytope $\mathscr{F}_\tau(x)$ is bounded unlike $\mathscr{F}$. Hence the central-similarity proximity region of $x \in \mathscr{F}$ is given by $N_{CS}(x,\tau) := \mathscr{F}_\tau(x) \cap \mathscr{F}$.

The convex hull $C_H(\mathcal{X}_{1-j})$ has at least $d+1$ facets (exactly $d+1$ when $n_{1-j} = d+1$), and since each outer simplex is associated with a facet, the number of outer simplices is at least $d + 1$. Let $\mathcal{S}^{(2)}_{1-j} = \{\mathscr{F}_1, \ldots, \mathscr{F}_L\}$ denotes the set of all outer simplices. This construction handles the points in $\mathcal{X}_j \setminus C_H(\mathcal{X}_{1-j})$ only. Together with the points inside $C_H(\mathcal{X}_{1-j})$, the CS-PCD D_j, whose vertex set is $\mathcal{V}_j = \mathcal{X}_j$, has at least

$$\sum_{k=1}^{K} I(\mathcal{X}_j \cap \mathfrak{S}_k \neq \emptyset) + \sum_{l=1}^{L} I(\mathcal{X}_j \cap \mathscr{F}_l \neq \emptyset)$$

many components. Collection of components both from inner and outer simplices constitutes a digraph D_j with the vertex set $\mathcal{X}_j$.

3.5 Minimum Dominating Sets

We model the target class, or the data set, with a digraph D such that prototype sets are equivalent to dominating sets of D. Ceyhan (2010) determined the appealing properties of *minimum dominating set* of CCCDs in $\mathbb{R}$ as a guideline in defining new parametric digraphs relative to the Delaunay tessellation of the non-target class. In $\mathbb{R}$, CCCDs have computationally tractable minimum dominating sets, and the exact distribution of *domination number* is known for target class points which are uniformly distributed within each cell. However, there is no polynomial time algorithm providing the exact minimum dominating sets of CCCDs in $\mathbb{R}^d$ for $d > 1$. In this section, we review the greedy algorithms for find the minimum dominating sets of CCCDs. Moreover, we provide a characterization of minimum dominating sets of PE-PCDs with barycentric coordinate systems and use them to introduce algorithms for finding the prototype sets in polynomial time.

We model the support of the data set, i.e. $s(F)$, by a mixture of proximity regions. Our estimate for the support of the class $\mathcal{X}$ is $Q := \cup_{x \in \mathcal{X}} \mathcal{N}(x)$ such that $\mathcal{X} \subset Q$. Nevertheless, the support of the target class $\mathcal{X}$ can be estimated by a cover with lower complexity (fewer proximity regions). For that purpose, we wish to reduce the model complexity by selecting an appropriate subset of proximity regions that still gives approximately the same estimate as Q; that is, let this cover be defined as $C := \cup_{x \in S} \mathcal{N}(x)$, where S is a prototype set of points $\mathcal{X}$ such that $\mathcal{X} \subset C$. Given the data set has two classes (or more), we model the class cover in a similar fashion. For the support of class conditional distributions $s(F_0)$ and $s(F_1)$, we estimate these classes with lower complexity covers $C_j := \cup_{x \in S_j} \mathcal{N}(x)$ such that $\mathcal{X}_j \subset C_j$ for $j = 0, 1$. Here, S_j is the prototype set of the target class $\mathcal{X}_j$ associated with the PCD D_j. A reasonable choice of prototype sets for our class covers are the minimum dominating sets of PE-PCDs, whose elements are often more "central" than the arbitrary sets of the same size. Dominating sets of minimum size are appealing since the size of the prototype sets determine the complexity of the model; that is, the smaller the set in cardinality (i.e. the model is lower in complexity), the higher the expected classification performance (Gao et al., 2013; Mehta et al., 1995; Rissanen, 1989).

PCD can easily be generalized to the multi-class case with J number of classes. To establish the set of covers $\mathcal{C} = \{C_1, C_2, \ldots, C_J\}$, the set of PCDs $\mathscr{D} = \{D_1, \ldots, D_J\}$, and the set of MDSs $\mathscr{S} = \{S_1, S_2 \ldots, S_J\}$ associated with a set of classes $\mathfrak{X} = \{\mathcal{X}_1, X_2, \ldots, \mathcal{X}_J\}$, we gather the classes into two classes as $\mathcal{X}_T = \mathcal{X}_j$ and $\mathcal{X}_{NT} = \cup_{t \neq j} \mathcal{X}_t$ for $t, j = 1, \ldots, J$. We refer to classes $\mathcal{X}_T$ and $\mathcal{X}_{NT}$ as target and non-target class, respectively. More specifically, target class is the class we want to find the cover of, and the non-target class is the union of the remaining classes. We transform the multi-class case into the two-class setting and find the cover of j'th class, C_j.

In general, a digraph $D = (\mathcal{V}, \mathcal{A})$ of order $n = |\mathcal{V}|$, a vertex v *dominates* itself and all vertices of the form $\{u : (v, u) \in \mathcal{A}\}$. A *dominating set*, S_D, for the digraph D is a subset of $\mathcal{V}$ such that each vertex $v \in \mathcal{V}$ is dominated by a vertex in S_D. A *minimum dominating set* (MDS), S_{MD}, is a dominating set of minimum cardinality, and the *domination number*, $\gamma(D)$, is defined as $\gamma(D) := |S_{MD}|$. If an MDS is of size one, we call it a *dominating point*. Finding an MDS is, in general, an NP-hard optimization problem (Arora and Lund, 1996; Karr, 1992). However, an approximate MDS can be obtained in $O(n^2)$ using a well-known greedy algorithm given in Algorithm 1 (Chvatal, 1979; Parekh, 1991). PCDs using $\mathcal{N}(\cdot) = \mathcal{N}_S(\cdot, \theta)$ (or CCCDs with parameter θ) and $\mathcal{N}(\cdot) = \mathcal{N}_{CS}(\cdot, \tau)$ are examples of such digraphs. The existence of polynomial time algorithm for find the MDS of CS-PCDs is still an open problem (Ceyhan, 2005).

Algorithm 1 The greedy algorithm for finding an approximate MDS of a digraph D. Here, $D[H]$ is the digraph induced by the set of vertices $H \subseteq \mathcal{V}$ (see West, 2000).

```
Input: A digraph $D = (\mathcal{V}, \mathcal{A})$
Output: An approximate MDS, $S$
  $H = \mathcal{V}$ and $S = \emptyset$
  while $H \neq \emptyset$ do
    $v^* \leftarrow \text{argmax}_{v \in \mathcal{V}(D)} |\{u \in \mathcal{V}(D) : (v, u) \in \mathcal{A}(D)\}|$
    $S \leftarrow S \cup \{v^*\}$
    $\bar{N}(v^*) \leftarrow \{u \in \mathcal{V}(D) : (v^*, u) \in \mathcal{A}(D)\} \cup \{v^*\}$
    $H \leftarrow \mathcal{V}(D) \setminus \bar{N}(v^*)$
    $D \leftarrow D[H]$
  end while
```

Algorithm 1 can be altered in many ways that satisfies some other properties. We will see that it is more desirable for cluster catch digraph, unsupervised adaptations of CCCDs, algorithms to choose covering balls closer to the cluster centers. These balls are bigger in size and have higher cardinality than those located around edges of hidden class, or cluster, supports. Hence, we can change Algorithm 1 in such way that the members of MDS have relatively high out-degrees, i.e. $d_{out}(v) := |\{u : (v, u) \in A(D)\}|$, compared to the MDSs found by Algorithm 1. In each iteration, Algorithm 2 chooses the vertex with both having maximum out degree and having arcs to those vertices not yet covered. This approach may produce MDSs whose covering balls are much closer to the center of clusters.

Algorithm 2 The greedy algorithm for finding an approximate MDS of a digraph D exploiting out degrees. Here, $D[H]$ is the digraph induced by the set of vertices $H \subseteq \mathcal{V}$ (see West, 2000).

```
Input: A digraph $D = (\mathcal{V}, \mathcal{A})$
Output: An approximate dominating set, $S$
  $S = \emptyset$
  while $\mathcal{V}(D) \neq \emptyset$ do
    $\mathcal{V}' \leftarrow \{v \in \mathcal{V} : (v,u) \in \mathcal{A},\, u \in \mathcal{V}(D)\}$
    $v^* \leftarrow \text{argmax}_{v \in \mathcal{V}'} |\{u \in \mathcal{V} : (v,u) \in \mathcal{A}\}|$
    $S \leftarrow S \cup \{v^*\}$
    $\mathcal{V} \leftarrow \mathcal{V} \setminus v^*$
    $\bar{N}(v^*) \leftarrow \{u \in \mathcal{V}(D) : (v^*,u) \in \mathcal{A}\} \cup \{v^*\}$
    $H \leftarrow \mathcal{V}(D) \setminus \bar{N}(v^*)$
    $D \leftarrow D[H]$
  end while
```

Algorithm 1 selects the vertex with the highest out-degree; however, we may use other attributes of vertices to select members of the MDSs; that is, let $sc : \mathcal{V} \rightarrow \mathbb{R}$ be a function on the vertices of a digraph, and let each vertex v is associated with a score $sc(v)$. Algorithm 3 adds a vertex $v \in \mathcal{V}$ to the set of dominating points S if it maximizes the score $sc(v)$.

The simplest of scoring functions for digraphs is the out-degree $d_{out}(v)$ of a vertex $v \in \mathcal{V}(D)$. In Algorithm 1, the out-degrees of vertices change each time a dominating point added to S. However, when $sc(v) := d_{out}(v)$ is fixed, the Algorithm 3 takes the original out degrees into account. This algorithm will successfully locate points of high domain influence which potentially correspond to the centers of some hidden classes or clusters when PCDs used for clustering tasks. Note that all these algorithms can be easily modified to be used on graphs.

Algorithm 3 The greedy algorithm exploiting the scoring function $sc : V \to R$ for finding an approximate MDS of a digraph D. Here, $D[H]$ is the digraph induced by the set of vertices $H \subseteq \mathcal{V}$ (see West, 2000).

Input: A digraph $D = (\mathcal{V}, \mathcal{A})$ and a score set $sc(V) := sc(v) : v \in V$

Output: An approximate dominating set, S

```
  $H = \mathcal{V}$ and $S = \emptyset$
  while $H \neq \emptyset$ do
    $v^* \leftarrow \text{argmax}_{v \in \mathcal{V}(D)} sc(v)$
    $S \leftarrow S \cup \{v^*\}$
    $H \leftarrow \mathcal{V}(D) \setminus \bar{N}(v^*)$
    $D \leftarrow D[H]$
  end while
```

3.5.1 MDSs for PCDs of Simplicial Proximity Maps

Exact MDS of PCDs with proximity maps $\mathcal{N}(\cdot) = \mathcal{N}_{PE}(\cdot, r)$ are computationally tractable unlike PCDs with maps $\mathcal{N}_S(\cdot, \theta)$. Many attributes of these PE proximity maps and the proof of the existence of an algorithm to find a set S_{MD} are conveniently implemented through the barycentric coordinate system. Before proving the results on the MDS for PE-PCDs, we give the following proposition on barycentric coordinates of points in a simplex $\mathfrak{S}$.

Proposition 3.5.1.1. *Let $\mathcal{Y} = \{\mathsf{y}_1, \mathsf{y}_2, \dots, \mathsf{y}_{d+1}\} \subset \mathbb{R}^d$ be a set of non-coplanar points for $d > 0$, and let $\mathfrak{S} = \mathfrak{S}(\mathcal{Y})$ be the d-simplex given by the set $\mathcal{Y}$. For $x, x^* \in \mathfrak{S}^o$, we have $d(x, f_i) < d(x^*, f_i)$ if and only if $w_{\mathfrak{S}}^{(i)}(x) < w_{\mathfrak{S}}^{(i)}(x^*)$ for all $i = 1, \dots, d+1$, where $d(x, f_i)$ is the distance between point x and the face f_i.*

Proof: For $i = 1, \dots, d+1$, note that f_i is the face of the simplex $\mathfrak{S}$ opposite to the vertex y_i. Let $L(\mathsf{y}_i, x)$ be the line through points x and y_i, and let $z \in f_i$ be the point that $L(\mathsf{y}_i, x)$ and f_i cross at. Also, recall that $\eta(\mathsf{y}_i, x)$ denotes the hyperplane through the point x, and parallel to f_i. Hence, for $\alpha \in (0,1)$,

$$x = \alpha \mathsf{y}_i + (1-\alpha) z,$$

and since z is a convex combination of the set $\{\mathsf{y}_k\}_{k \neq i}$,

$$x = \alpha \mathsf{y}_i + \left(\sum_{k=1;k\neq i}^{d+1} (1-\alpha)\beta_k \mathsf{y}_k \right),$$

for $\beta_k \in (0,1)$. Thus, $w_{\mathfrak{S}}^{(i)}(x) = \alpha$ by the uniqueness of $\mathbf{w}_{\mathfrak{S}}(x)$. Observe that $\alpha = d(x,z)/d(\mathsf{y}_i,z) = d(x,f_i)/d(\mathsf{y}_i,f_i)$ since distances $d(x,z)$ and $d(x,f_i) = d(\eta(\mathsf{y}_i,x),f_i)$ are directly proportional. In fact, points that are on the same line parallel to f_i have the same i'th barycentric coordinate $w_{\mathfrak{S}}^{(i)}(x)$ corresponding to the vertex y_i. Also, recall that with decreasing α, the point x gets closer to f_i ($x \in f_i$ if $\alpha = 0$, and $x = \mathsf{y}_i$ if $\alpha = 1$). Then, for any two points $x, x^* \in \mathfrak{S}^o$, we have $w_{\mathfrak{S}}^{(i)}(x) < w_{\mathfrak{S}}^{(i)}(x^*)$ if and only if $d(\eta(\mathsf{y}_i,x),f_i) < d(\eta(\mathsf{y}_i,x^*),f_i)$ if and only if $d(x,f_i) < d(x^*,f_i)$. ■

Barycentric coordinates of a set of points in simplex $\mathfrak{S}$ help one characterize the set of *local extremum* points, where a subset of local extremum points constitute the MDS S_{MD}. We use the Proposition 3.5.1.1 to prove the following theorem on S_{MD} of a PE-PCD D.

Theorem 3.5.1.1. *Let $\mathcal{Z} = \{z_1, z_2, \ldots, z_n\} \subset \mathbb{R}^d$ and $\mathcal{Y} = \{\mathsf{y}_1, \mathsf{y}_2, \ldots, \mathsf{y}_{d+1}\} \subset \mathbb{R}^d$ for $d > 0$, and let $\mathfrak{S} = \mathfrak{S}(\mathcal{Y})$ be the d-simplex given by the set $\mathcal{Y}$ such that $\mathcal{Z} \subset \mathfrak{S}^o$. Hence, given the map $\mathcal{N}_{PE}(\cdot, r)$, we have $\gamma(D) \leq d+1$ for PE-PCD D with vertex set $\mathcal{V} = \mathcal{Z}$.*

Proof: Let $x, x^*, M \in \mathfrak{S}^o$. For $i = 1, \ldots, d+1$, we show that there exists a point $x_{[i]} \in \mathcal{Z} \cap R_M(\mathsf{y}_i)$ such that $\mathcal{Z} \cap R_M(\mathsf{y}_i) \subset N_{PE}(x_{[i]}, r)$ for all $r \in (1,\infty)$. It is easy to see that $d(x^*,f_i) < d(x,f_i)$ if and only if $\mathcal{N}_{PE}(x^*,r) \subset N_{PE}(x,r)$. Hence, $d(x_{[i]},f_i) = \min_{z\in\mathcal{Z}} d(z,f_i)$ if and only if $\mathcal{N}_{PE}(z,r) \subset N_{PE}(x_{[i]},r)$ for all $z \in \mathcal{Z} \cap R_M(\mathsf{y}_i)$. Also, by Proposition 3.5.1.1, note that $d(x_{[i]},f_i) \leq \min_{z\in\mathcal{Z}} d(z,f_i)$ if and only if $w_{\mathfrak{S}}^{(i)}(x_{[i]}) \leq \min_{z\in\mathcal{Z}} w_{\mathfrak{S}}^{(i)}(z)$. Thus, the *local extremum* point $x_{[i]}$ is given by

$$x_{[i]} := \operatorname*{argmin}_{x \in \mathcal{Z}\cap R_M(\mathsf{y}_i)} w_{\mathfrak{S}}^{(i)}(x).$$

Finally, observe that $\mathcal{Z} \subset \cup_{i=1}^{d+1} \mathcal{N}_{PE}(x_{[i]}, r)$. Hence, the set of all local extremum points $\{x_{[1]}, \ldots, x_{[d+1]}\}$ is a dominating set of the points $\mathcal{Z} \subset \mathfrak{S}^o$, so $\gamma(D) \leq d+1$. ■

MDSs of PE-PCDs are found by locating the *local extremum* point $x_{[i]}$ of the vertex region $R_{M_C}(\mathsf{y}_i)$ for all $i = 1, \ldots, d+1$. By Theorem 3.5.1.1, in $R_{M_C}(\mathsf{y}_i)$, the point $x_{[i]}$ is the closest points to the face f_i. For a set of d-simplices given by the Delaunay tessellation of $\mathcal{X}_{1-j}$, Algorithm 4 identifies all the local extremum points of each d-simplex in order to find the (exact) MDS $S_j = S_{MD}$.

Let $D_j = (\mathcal{V}_j, \mathcal{A}_j)$ be a PE-PCD with vertex set $\mathcal{V} = \mathcal{X}_j$. In Algorithm 4, we partition $\mathcal{X}_j$ into such subsets that each subset falls into a single d-simplex of the Delaunay tessellation of the set $\mathcal{X}_{1-j}$. Let $\mathcal{S}_{1-j}$ be the set of all d-simplices associated with $\mathcal{X}_{1-j}$. Moreover, for each $\mathfrak{S} \in \mathcal{S}_{1-j}$, we further partition the subset $\mathcal{X}_j \cap \mathfrak{S}$ into subsets that each subset falls into a single vertex region of $\mathfrak{S}$. In each vertex region $R_{M_C}(\mathsf{y}_i)$, we find the *local extremum* point $x_{[i]}$. Let $S(D)$ denote the MDS and $\gamma(D)$ denote the domination number of a digraph D. Also, let $D_j[\mathfrak{S}]$ be the digraph induced by points of $\mathcal{X}_j$ inside the d-simplex $\mathfrak{S}$, i.e. $\mathcal{X}_j \cap \mathfrak{S}$. Recall that, as a result of Theorem 3.5.1.1, $\gamma(D_j[\mathfrak{S}]) \leq d+1$ since $\mathcal{X}_j \cap \mathfrak{S} \subset \cup_{i=1}^{d+1} \mathcal{N}_{PE}(x_{[i]}, r)$. To find $S(D_j[\mathfrak{S}])$, we check all subsets of the set of local extremum points, from smallest cardinality to highest, and check if $\mathcal{X}_j \cap \mathfrak{S}$ is in the union of proximity regions of these subsets of local extremum points. For example, $S(D_j[\mathfrak{S}]) = \{x_{[l]}\}$ and $\gamma(D_j[\mathfrak{S}]) = 1$ if $\mathcal{X}_j \cap R_{M_C}(\mathsf{y}_i) \subset N_{PE}(x_{[l]}, r)$ for some $l = 1, 2, 3$; else $S(D_j[\mathfrak{S}]) = \{x_{[l_1]}, x_{[l_2]}\}$ and $\gamma(D_j[\mathfrak{S}]) = 2$ if $\mathcal{X}_j \cap R_{M_C}(\mathsf{y}_i) \subset N_{PE}(x_{[l_1]}, r) \cup N_{PE}(x_{[l_2]}, r)$ for some $\{l_1, l_2\} \in \binom{\{1,2,3\}}{2}$; or else $S(D_j[\mathfrak{S}]) = \{x_{[1]}, x_{[2]}, x_{[3]}\}$ and $\gamma(D_j[\mathfrak{S}]) = 3$ if $\mathcal{X}_j \cap R_{M_C}(\mathsf{y}_i) \subset \cup_{l=1,2,3} \mathcal{N}_{PE}(x_{[l]}, r)$. The resulting MDS of D_j for $\mathcal{X}_j \cap C_H(\mathcal{X}_{1-j})$ is the union of these sets, i.e., $S_j = \cup_{\mathfrak{S} \in \mathcal{S}_{1-j}} S(D_j[\mathfrak{S}])$ and $\gamma(D_j) = |S_j|$. Observe that $S(D_j[\mathfrak{S}]) = \emptyset$ if $\mathcal{X}_j \cap \mathfrak{S} = \emptyset$. This algorithm is guaranteed to terminate, as long as n_0 and n_1 are both finite.

The level of reduction depends also on the magnitude of the expansion parameter r. In fact, the larger the magnitude of r, the more likely the $S(D_j[\mathfrak{S}])$ have smaller cardinality, i.e. the more the reduction in the data set. Thus, we have a stochastic

Algorithm 4 The algorithm for finding the (exact) MDS S_j of a PE-PCD D_j induced by $\mathcal{X}_j \cap C_H(\mathcal{X}_{1-j})$.

```
Input: The target class $\mathcal{X}_j$, a set of $d$-simplices of the non-target class $\mathcal{S}_{1-j}$, and the
  PE proximity map $\mathcal{N}_{PE}(\cdot, r)$.
Output: The MDS, $S_j$
  $S_j = \emptyset$
  for all $\mathfrak{S} \in \mathcal{S}_{1-j}$ where $\mathcal{X}_j \cap \mathfrak{S} \neq \emptyset$ do
    $\mathcal{X}_j^* \leftarrow \mathcal{X}_j \cap \mathfrak{S}$ and let $\{\mathsf{y}_1, \ldots, \mathsf{y}_{d+1}\}$ be the vertices of $\mathfrak{S}$.
    for $i = 1, \ldots, d+1$ do
      Let $x_{[i]} \leftarrow \underset{x \in \mathcal{X}_j^* \cap R_{M_C}(\mathsf{y}_i)}{\operatorname{argmin}} w_{\mathfrak{S}}^{(i)}(x)$.
    end for
    for $t = 1, \ldots, d+1$ do
      if there exists a set $\{l_1, \ldots, l_t\} \in \binom{\{1, \ldots, d+1\}}{t}$ s.t. $\mathcal{X}_j^* \subset \cup_{a=1}^{t} \mathcal{N}_{PE}(x_{[l_a]}, r)$
      then
        $S_j \leftarrow S_j \cup \{x_{[l_1]}, \ldots, x_{[l_t]}\}$
        break
      end if
    end for
  end for
```

ordering as follows:

Theorem 3.5.1.2. *Let $\gamma(D_j[\mathfrak{S}], r)$ be the domination number of the PE-PCD $D_j(\mathfrak{S})$ with expansion parameter r. Then for $r_1 < r_2$, we have $\gamma(D_j[\mathfrak{S}], r_2) \leq^{ST} \gamma(D_j[\mathfrak{S}], r_1)$ where $\leq^{ST}$ stands for "stochastically smaller than".*

Proof: Suppose $r_1 < r_2$. Then in a given simplex $\mathfrak{S}_k$ for $k = 1, \ldots, K$, let $\gamma_k(r) := \gamma(\mathfrak{S}_k, r)$ be the domination number of the component of the PE-PCD D_j whose vertices are restricted to the interior of $\mathfrak{S}_k$. Let $\mathcal{Z} = \{Z_1, Z_2, \ldots, Z_n\}$ be a set of i.i.d. random variables drawn from a continuous distribution F whose support is $\mathfrak{S}_k$, and let $Z_{[i]}$ be the local extremum point of $\mathcal{Z} \cap R_{M_C}(\mathsf{y}_i)$ where y_i being the i'th vertex of

$\mathfrak{S}_k$. Also, let $\text{Vol}(N_{PE}(x,r))$ be the volume of the $\mathcal{N}_{PE}(x,r)$ of a point $x \in \mathfrak{S}_k^o$. Note that,

$$\text{Vol}(N_{PE}(x,r_1)) < \text{Vol}(N_{PE}(x,r_2)).$$

Hence, since $\mathcal{N}_{PE}(x,r_1) \subset \mathcal{N}_{PE}(x,r_2)$,

$$\text{Vol}(\mathcal{N}_{PE}(Z_{[i]},r_1)) \leq^{ST} \text{Vol}(\mathcal{N}_{PE}(Z_{[i]},r_2)).$$

Now, let $\{l_1,\ldots,l_t\} \in \binom{\{1,\ldots,d+1\}}{t}$ be any set of indices associated with a subset of all local extremum points $t = 1,\ldots,d+1$. Thus,

$$\text{Vol}\left(\cup_{a=1}^{t}\mathcal{N}_{PE}(Z_{[l_a]},r_1)\right) \leq^{ST} \text{Vol}\left(\cup_{a=1}^{t}\mathcal{N}_{PE}(Z_{[l_a]},r_2)\right).$$

Hence, given that the event $\mathcal{Z} \subset \left(\cup_{a=1}^{t}\mathcal{N}_{PE}(Z_{[l_a]},r)\right)$ implies $\gamma_k(r) \leq t$, we can show that

$$P\left(\mathcal{Z} \subset \left(\cup_{a=1}^{t}\mathcal{N}_{PE}(Z_{[l_a]},r_2)\right)\right) \leq P\left(\mathcal{Z} \subset \left(\cup_{a=1}^{t}\mathcal{N}_{PE}(Z_{[l_a]},r_1)\right)\right),$$

and

$$P(\gamma_k(r_2) \leq t) \leq P(\gamma_k(r_1) \leq t)$$

for $t = 1,\ldots,d+1$. ■

Algorithm 4 ignores the target class points outside the convex hull of the non-target class. This is not the case with Algorithm 1, since the map $\mathcal{N}_S(\cdot,\theta)$ is defined over all points $\mathcal{X}_j$ whereas the original PE proximity map $\mathcal{N}_{PE}(\cdot,r)$ is not. Hence, the prototype set S_j only yields a reduction in the set $\mathcal{X}_j \cap C_H(\mathcal{X}_{1-j})$. Solving this issue requires different approaches. One solution is to define covering methods with two proximity maps that are the PE proximity map and the other which does not require the target class points to be inside the convex hull of the non-target class points, e.g. spherical proximity regions (proximity maps $N_S(\cdot,\theta)$).

Algorithm 5 uses both maps $\mathcal{N}_{PE}(\cdot,r)$ and $\mathcal{N}_S(\cdot,\theta)$ to generate a prototype S_j for the target class $\mathcal{X}_j$. There are two separate MDSs, $S_j^{(1)}$ which is exactly minimum, and $S_j^{(2)}$ which is approximately minimum. Each of the two proximity maps is associated with two distinct digraphs such that $\mathcal{X}_j \cap C_H(\mathcal{X}_{1-j})$ constitutes the vertex set of one digraph and $\mathcal{X}_j \setminus C_H(\mathcal{X}_{1-j})$ constitute the vertex of another, where the non-target class is always $\mathcal{X}_{1-j}$. Algorithm 4 finds a prototype set $S_j^{(1)}$ for $\mathcal{X}_j \cap C_H(\mathcal{X}_{1-j})$, and then the prototype set $S_j^{(2)}$ for $\mathcal{X}_j \setminus C_H(\mathcal{X}_{1-j})$ is appended to the overall prototype set $S_j = S_j^{(1)} \cup S_j^{(2)}$ as in Algorithm 5. Note that the set S_j is an approximate MDS since $S_j^{(2)}$ is approximately minimum.

Algorithm 5 The algorithm for finding the MDS S_j of PCD D_j defined by the proximity maps $\mathcal{N}_{PE}(\cdot, r)$ and $\mathcal{N}_S(\cdot, \theta)$.

```
Input: The target class $\mathcal{X}_j$, a set of $d$-simplices of the non-target class $\mathcal{S}_{1-j}$, and the
    proximity maps $\mathcal{N}_{PE}(\cdot, r)$ and $\mathcal{N}_S(\cdot, \theta)$.
Output: The approximate MDS, $S_j$
  $S_j^{(1)} = \emptyset$ and $S_j^{(2)} = \emptyset$
  Find the MDS of $\mathcal{X}_j \cap C_H(X_{1-j})$ in Algorithm 2 and assign it to $S_j^{(1)}$
  $\mathcal{X}_j' = \mathcal{X}_j \setminus C_H(\mathcal{X}_{1-j})$.
  Find the approximate MDS as in Algorithm 1 where the target class is $\mathcal{X}_j'$ and the
  non-target class is $\mathcal{X}_{1-j}$, and assign it to $S_j^{(2)}$
  $S_j = S_j^{(1)} \cup S_j^{(2)}$
```

Algorithm 6 uses only the PE proximity map $\mathcal{N}_{PE}(\cdot, r)$ with the original version inside $C_H(\mathcal{X}_{1-j})$ and extended version outside $C_H(\mathcal{X}_{1-j})$. The cover is a mixture of d-simplices and d-polytopes. Given a set of d-simplices $\mathcal{S}_{1-j}^{(1)}$ and a set of outer simplices $\mathcal{S}_{1-j}^{(2)}$, we find the respective local extremum points of each d-simplex and outer simplex. Local extremum points of d-simplices are found as in Algorithm 4, and then we find the local extremum points of the remaining points to get the prototype set of the entire target class $\mathcal{X}_j$. The following theorem provides a result on the local extremum points in an outer simplex $\mathscr{F}$. Note that, in Algorithm 6, the set S_j is the exact MDS since both $S_j^{(1)}$ and $S_j^{(2)}$ are exact MDSs for the PE-PCDs induced by $\mathcal{X}_j \cap C_H(\mathcal{X}_{1-j})$ and $\mathcal{X}_j \setminus C_H(\mathcal{X}_{1-j})$, respectively.

Theorem 3.5.1.3. *Let $\mathcal{Z} = \{z_1, z_2, \ldots, z_n\} \subset \mathbb{R}^d$, let $\mathcal{F}$ be a facet of $C_H(\mathcal{X}_{1-j})$ and let $\mathscr{F}$ be the associated outer simplex such that $\mathcal{Z} \subset \mathscr{F}^o$. Hence, the local extremum point and the S_{MD} of the PE-PCD D restricted to $\mathscr{F}^o$ is found in linear time and is* *equal to 1.*

Proof: We show that there is a point $s \in \mathcal{Z}$ such that $\mathcal{X} \subset \mathcal{N}_{PE}(s,r)$ for all $r \in (1,\infty)$. As a remark, note that $\eta(x,\mathcal{F})$ denotes the hyperplane through x, and is parallel to $\mathcal{F}$. Thus, for $x, x^* \in \mathcal{F}^o$, observe that $d(x,\mathcal{F}) < d(x^*,\mathcal{F})$ if and only if $d(\eta(x,\mathcal{F}),\mathcal{F}) < d(\eta(x^*,\mathcal{F}),\mathcal{F})$ if and only if $\mathcal{N}_{PE}(x,r) \subset \mathcal{N}_{PE}(x^*,r)$. Thus, the *local extremum* point s is given by

$$s := \operatorname*{argmax}_{x\in\mathcal{Z}} d(x,\mathcal{F}).$$

Therefore, $S_{MD} = \{s\}$ yields the result. ■

Algorithm 6 The algorithm for finding the (exact) MDS S_j of PE-PCD D_j with vertex set $\mathcal{X}_j$.

Input: The target class $\mathcal{X}_j$, the set of d-simplices $\mathcal{S}^{(1)}_{1-j}$, the set of outer simplices $\mathcal{S}^{(2)}_{1-j}$.

Output: The MDS, S_j

```
  $S^{(1)}_j = \emptyset$ and $S^{(2)}_j = \emptyset$
  Find the MDS in Algorithm 2 and assign it to $S^{(1)}_j$
  $\mathcal{X}'_j = \mathcal{X}_j \setminus C_H(\mathcal{X}_{1-j})$.
  for all $\mathscr{F} \in \mathcal{S}^{(2)}_{1-j}$ where $\mathcal{X}'_j \cap \mathscr{F} \neq \emptyset$ do
    $\mathcal{X}^*_j \leftarrow \mathcal{X}'_j \cap \mathscr{F}$
    Let $s \in \mathcal{X}^*_j$ be the local extremum point in $\mathscr{F}$
    $S^{(2)}_j \leftarrow S^{(2)}_j \cup \{s\}$
  end for
  $S_j = S^{(1)}_j \cup S^{(2)}_j$
```

Given Theorems 3.5.1.1 and 3.5.1.3, Algorithm 6 may be the most appealing algorithm since it gives the exact MDS for the complete target class $\mathcal{X}_j$. However, the following theorem show that the cardinality of such sets increase exponentially on dimensionality of the data set, even though it is polynomial on the number of observations.

Theorem 3.5.1.4. *Algorithm 6 finds an exact MDS S_j of the target class $\mathcal{X}_j$ in $\mathcal{O}(d^k n_{1-j}^2 + 2^d n_{1-j}^{\lceil d/2 \rceil})$ time for $k > 1$ where $|S_j| = \mathcal{O}(d n_{1-j}^{\lceil d/2 \rceil})$.*

Proof: A Delaunay tessellation of the non-target class $\mathcal{X}_{1-j} \subset \mathbb{R}^d$ is found in $\mathcal{O}(d^k n_{1-j}^2)$ time with the Bowyer-Watson algorithm for some $k > 1$, depending on the complexity of the algorithm that finds the circumcenter of a d-simplex (Watson, 1981). The resulting tessellation with n_{1-j} vertices has at most $\mathcal{O}(n_{1-j}^{\lceil d/2 \rceil})$ simplices and at most $\mathcal{O}(n_{1-j}^{\lfloor d/2 \rfloor})$ facets (Seidel, 1995). Hence the union of sets of d-simplices $\mathcal{S}_j^{(1)}$ and outer simplices $\mathcal{S}_{1-j}^{(2)}$ is of cardinality at most $\mathcal{O}(n_{1-j}^{\lceil d/2 \rceil})$. Now, for each simplex $\mathfrak{S} \in \mathcal{S}_j^{(1)}$ or each outer simplex $\mathcal{F} \in \mathcal{S}_j^{(2)}$, the local extremum points are found in linear time. Each simplex is divided into $d+1$ vertex regions, having their own set of local extremum points. A minimum cardinality subset of the set of local extremum points is of cardinality at most $d+1$ and found in a brute force fashion. For outer simplices, however, the local extremum point is the farthest point to the associated facet of the Delaunay tessellation. Thus, it takes at most $\mathcal{O}(2^d)$ and $\mathcal{O}(1)$ time to find the exact minimum subsets of local extremum points for each simplex and outer simplex, respectively. Then the result follows. ■

Theorem 3.5.1.4 shows the exponential increase of the number of prototypes as dimensionality increases. Thus, the complexity of the class cover model also increases exponentially, which might lead to overfitting. We will investigate this issue further in Sections 5.4 and 5.5.

Part II

Classification

Chapter 4

CLASSIFICATION OF IMBALANCED DATA WITH CLASS COVER CATCH DIGRAPHS

4.1 Introduction

Class imbalance problem has recently become a topic of extensive research. In a two-class setting, imbalance in class(es) occurs when one class is represented by far more observations (points) than the other class in the data set (Chawla et al., 2004; López et al., 2013). Class imbalance problem is observed in many areas such as medicine, fraud detection and education. Some examples are clinical trials in which only 5% of patients in the data set have a certain disease, such as cancer (Mazurowski et al., 2008); detecting fraudulent customers where most individuals are law-abiding in insurance, credit card and telecommunications industries (Phua et al., 2004); and archives of college students where mostly the ones who have fair results are kept (Thai-Nghe et al., 2009). In these and many other real life cases, majority class (i.e., the class with larger size) confounds the classifier performance by hindering the detection of subjects from the minority class (i.e., the class with fewer points).

The classification methods in machine learning usually suffer from the imbalance of class sizes in the data sets because most of these methods work on the assumption that class sizes are balanced (Japkowicz and Stephen, 2002). For example, the commonly used k-nearest neighbor (k-NN) classification algorithm is highly influenced by the class imbalance problem. In the k-NN approach, a new point is classified as the class of the most frequent one from its first k nearest neighbors (Cover and Hart, 1967; Evelyn Fix, 1989). As a result, in a two-class setting where one class substantially outnumbers the other, a point is more likely to be classified as the majority class by the k-NN classifier. In literature, sensitivity of k-NN classifier to the class imbalance problem and some solutions on choosing the appropriate k have been discussed in cases of imbalanced classes (García et al., 2008; Hand and Vinciotti, 2003; Mani and Zhang, 2003). Decision trees and support vector machines (SVM) are also some of the well known classifiers that are sensitive to the class imbalance in a data set (Japkowicz and Stephen, 2002; Tang et al., 2009). SVMs are among the most commonly used algorithms in the machine learning literature due to their well understood theory and high performance among popular algorithms (Fernández-Delgado et al., 2014; Wu et al., 2008), but these methods have been demonstrated to be inefficient against highly imbalanced data sets, although SVMs are still robust to moderately imbalanced data sets (Akbani et al., 2004; Raskutti and Kowalczyk, 2004).

In this chapter, we study the effects of class imbalance on two CCCD classifiers, P-CCCD and RW-CCCD. Moreover, we report on the effects of class overlapping problem (which is defined as deterioration of classification performance when class supports overlap) along with the class imbalance problem to further investigate the performance of CCCD classifiers when imbalance and overlapping between classes co-exist. Thus, we show that when there is a considerable amount of class imbalance, whether class supports overlap or not, the CCCD classifiers perform better than the k-NN classifier. We show the robustness of CCCD classifiers to the class imbalance by simulating cases having increasing levels of class imbalance. We also compare CCCD classifiers with SVM classifiers which are potentially robust to moderate levels of class imbalance but not to high levels. With respect to class imbalance problem, the k-NN, SVM and decision tree classifiers may be referred to as “weak” classifiers; that is, these methods perform weakly when there is imbalance in the data set.

Among the two variations of CCCD classifiers, we show that the RW-CCCD is more appealing in many aspects. For both simulated and real life examples, RW-CCCDs perform better than P-CCCDs and other classifiers when the classes of data sets are imbalanced and/or overlapping. Moreover, we report on the complexity of the these two CCCD classifiers and demonstrate that RW-CCCDs reduce the data sets substantially more than the other classifiers, thus increasing the testing speed. But most importantly, while reducing the majority class to mitigate the effects of class imbalances, CCCDs preserve the information on the discarded points of the majority class. CCCDs provide a novel solution to the class imbalance problem; that is, they capture the density around prototype points (i.e., members of the dominating sets) as radii of the covering balls. Hence, CCCDs preserve the information while reducing the data set In the literature, only classifiers based on ensemble classifiers achieve a similar task which requires multiple classifiers to be employed, and thus, result in lengthy training and testing time. However, CCCDs define single classifiers that undersample the data set with, possibly, a slight loss of information.

4.2 Methods for Handling Class Imbalance Problem

Solving the class imbalance problem received considerable attention in the machine learning literature (Chawla et al., 2004; Kotsiantis et al., 2006; Longadge and Dongre, 2013). Almost all algorithms designed to mitigate the effects of class imbalance incorporate a “weak” classifier which is modified to show some level of robustness to the class imbalance problem. The weak algorithm is modified either (i) in data level which involves a pre-processing of the data set being used in training, or (ii) in algorithmic level such that a “strong” classifier is constructed with a decision rule suited for the imbalances in the data set. Many modern algorithms are hybrids of both types; but in particular, there are mainly three of them: resampling methods, cost-sensitive methods, and ensemble methods (He and Garcia, 2009).

Resampling methods are commonly employed to remove the effects of class imbalance in the classification process. Resampling methods provide solutions to the class imbalance problem by (i) downsizing the majority class (undersampling) or (ii) generating new (synthetic) points for the minority class (oversampling). Hence, such methods modify the classifiers only at the data level. It might be useful to clean or erase some points in the majority class to balance the data (Drummond et al., 2003; Liu et al., 2009). However, in some cases, all points from both classes may be valuable/important, and hence, should be kept despite the differences in the class sizes. Oversampling methods generate synthetic points similar to the minority class to mitigate the class imbalance problem while preserving the information (Han et al., 2005). On the other hand, Batista et al. (2004) suggest that the combination of both over and undersampling methods can further improve the classification performance. One such method is the SMOTE+ENN method where the oversampling method SMOTE of Chawla et al. (2002) and edited nearest neighbors (ENN) method of Wilson (1972) are applied to an imbalanced data set, consecutively. While SMOTE balances the classes of the data set by generating artificial points between members of the minority class, ENN cleans the data set to further increase the classification performance of the weak classifier. Here, ENN method is an undersampling method that primarily aims to remove noisy points from the data set but not to balance the classes.

Another family of methods, namely cost-sensitive learning methods, has originated from real life: the cost of misclassifying a minority and a majority class member is usually not the same (Elkan, 2001). Frequently, the minority class has higher misclassification cost than the majority class. Classification methods such as decision trees (e.g., C4.5), can be modified to take these costs into account (Ling et al., 2004; Zadrozny et al., 2003). C5.0 is an extended version of C4.5 incorporating the cost of each class (Kuhn and Johnson, 2013). Most weak classifiers can be easily modified so as to recognizing misclassification costs. The constrained violation cost C of SVM classifiers can be adjusted to individual class costs (Chang and Lin, 2011). As for k-NN, one solution is to appoint weights to all points of the data set with respect to their classes. Hence, such weights are the costs of classes giving precedence to minority class points (Barandela et al., 2003). On the other hand, for those algorithms that costs are not inherently recognizable or available, meta-learning schemes can be used along with weak classifiers without modifying the classifiers. Such learning methods are similar to ensemble learning methods (Domingos, 1999).

A fast developing field called ensemble learning also contributes to the family of methods handling the class imbalance problem (Galar et al., 2012). The idea is to combine several classifiers to create a new classifier which has significantly better performance than its constituents (Rokach, 2010). AdaBoost is a popular algorithm among this family of learning methods (Freund and Schapire, 1997; Wu et al., 2008). AdaBoost assigns weights to each of the points in the data set and updates these weights in accordance with how well the points are estimated by each classifier. Galar et al. (2012) provide a survey of the most important ensemble learning methods that solve the class imbalance problem. However, it has been observed in some studies that ensemble learning methods work best when used together with resampling methods (López et al., 2013). In fact, ensembling and resampling schemes compensate the shortcomings of each other. The EasyEnsemble is a classifier with two levels of ensembles. First, a random undersampled majority class and the original minority class are used to train an ensemble classifier, then another random sample is drawn in the same way to train a second ensemble. This process is repeated several times to mitigate the effects of information loss as each ensemble would be applied on a different random subset of the majority class.

4.3 Classification with Class Cover Catch Digraphs

We employ two families of CCCDs, pure CCCDs (P-CCCDs) and random walk CCCDs (RW-CCCDs) that differ in the definition of the radius $r(x)$. In these two digraphs, the (approximate) MDS S and the classifier are defined in slightly different ways; with the main distinction between the two being the way the covers are defined. The covering balls of P-CCCDs do not contain any non-target class point whereas RW-CCCDs possibly allow some non-target class points inside of the class cover of the target class so as to avoid overfitting. Moreover, some target class points may also be excluded from the covers of RW-CCCDs.

4.3.1 Classification with P-CCCDs

In P-CCCDs, the covering balls $B = B(x, r(x))$ exclude all non-target class points. Thus, for a target class point $x \in \mathcal{X}_j$, which is the center of a ball B, the radius

$r(x)$ should be smaller than the distance between x and the closest non-target point $y \in \mathcal{X}_{1-j}$: $r(x) < \min_{y \in \mathcal{X}_{1-j}} d(x,y)$. Given $\theta \in (0,1]$, the radius $r(x)$ is defined as follows (Marchette, 2010):

$$r(x) := (1-\theta)d(x,l(x)) + \theta d(x,u(x)), \tag{4.1}$$

where

$$u(x) := \operatorname{argmin}_{y \in \mathcal{X}_{1-j}} d(x,y)$$

and

$$l(x) := \operatorname{argmax}_{z \in \mathcal{X}_j} \{d(x,z) : d(x,z) < d(x,u(x))\}.$$

The effect of parameter θ on the radius $r(x)$ is illustrated in Figure 4.1 (DeVinney, 2003). The ball with radius $r(x)$ catches the neighboring target class points, and for any $\theta \in (0,1]$, the ball $B(x,r(x))$ catches the same points as well. Hence, the choice of θ does not effect the structure of digraph but might affect the classification performance which will be shown later in Section 4.5. On the other hand, for all $x \in \mathcal{X}_j$, the definition of $r(x)$ in Equation (4.1) keeps any non-target point $y \in \mathcal{X}_{1-j}$ out of the ball $B(x,r(x))$, that is $\mathcal{X}_{1-j} \cup B(x,r(x)) = \emptyset$ for all $B(x,r(x)) \in C_j$. Here, $B(x,r(x))$ is an open ball: $B(x,r(x)) = \{z \in \mathbb{R}^d : d(x,z) < r(x)\}$. The digraph D is "pure" since the balls contain only the target class points; hence, the name pure-CCCD. Once all balls are constructed, so is the digraph D_j. Therefore, we have to find a covering set C_j which is equivalent to finding a MDS S_j. The greedy algorithm of finding an approximate MDS of a P-CCCD is given in Algorithm 1. At each iteration, the vertex which has the largest neighborhood (i.e., highest number of arcs) is removed from the graph together with its neighbors. Then, the process is repeated until all vertices of D_j are removed. The algorithm adds elements to the dominating set until all points are either dominated or dominate some other points.

Before Algorithm 1 finds an approximate solution, we should first construct the digraph D_j. The P-CCCD cover C_j and the P-CCCD D_j depend on the distances between points of the target class $\mathcal{X}_j$, denoted by the matrix $\mathcal{M}_j$, and the distances

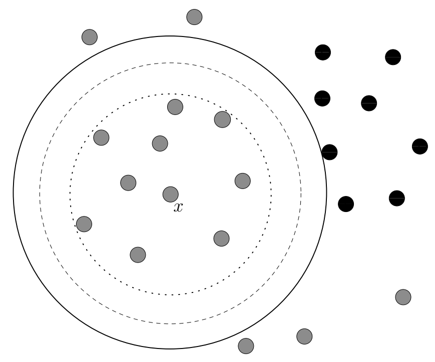


Figure 4.1: The effect of θ on the radius $r(x)$ of the target class point x in a two-class setting. Grey and black points represent the target and non-target classes, respectively. The solid circle centered at x is constructed with the radius associated with $\theta = 1$ and the dashed one with $\theta = 0.0001$ (DeVinney, 2003).

from all points of $\mathcal{X}_j$ to all points of $\mathcal{X}_{1-j}$, denoted by matrix $\mathcal{M}_{j,1-j}$. Later, we construct the set of balls $\mathcal{B}_j = \{B(x, r(x)) : x \in \mathcal{X}_j\}$, and get the set of arcs $\mathcal{A}_j$ where $\mathcal{V}_j = \mathcal{X}_j$. Hence, the minimum cardinality ball cover problem is reduced to a MDS problem. We find such a cover with Algorithm 7 which runs in quadratic time and, in addition, depends on the dimensionality of the training set $\mathcal{X} = \mathcal{X}_0 \cup \mathcal{X}_1$. The P-CCCD of one class, its associated class cover (constructed by the elements of the dominating set), and covers of both classes are illustrated in Figure 4.2.

Algorithm 7 The greedy algorithm for finding an approximate minimum cardinality ball cover C_j of the target class points $\mathcal{X}_j$ given a set of non-target class points $\mathcal{X}_{1-j}$.

```
Input: The target class $\mathcal{X}_j$, the non-target class $\mathcal{X}_{1-j}$ and the P-CCCD parameter $\theta \in (0,1]$
Output: An approximate minimum cardinality ball cover $C_j$
1: $r(x) := (1-\theta)d(x, l(x)) + \theta d(x, u(x))$ for all $x \in \mathcal{X}_j$
2: Construct the digraph $D_j$ with the set $\mathcal{B}_j$.
3: Find the approximate MDS $S_j$ of digraph $D$ by Algorithm 1.
4: $C_j := \cup_{s \in S} B(s, r(s))$
```

Theorem 1 *Algorithm 7 is an $\mathcal{O}(\log n_j)$-approximation algorithm and finds an approximate minimum cardinality ball cover C_j of the target class $\mathcal{X}_j$ in $\mathcal{O}(n_j(n_j +$*

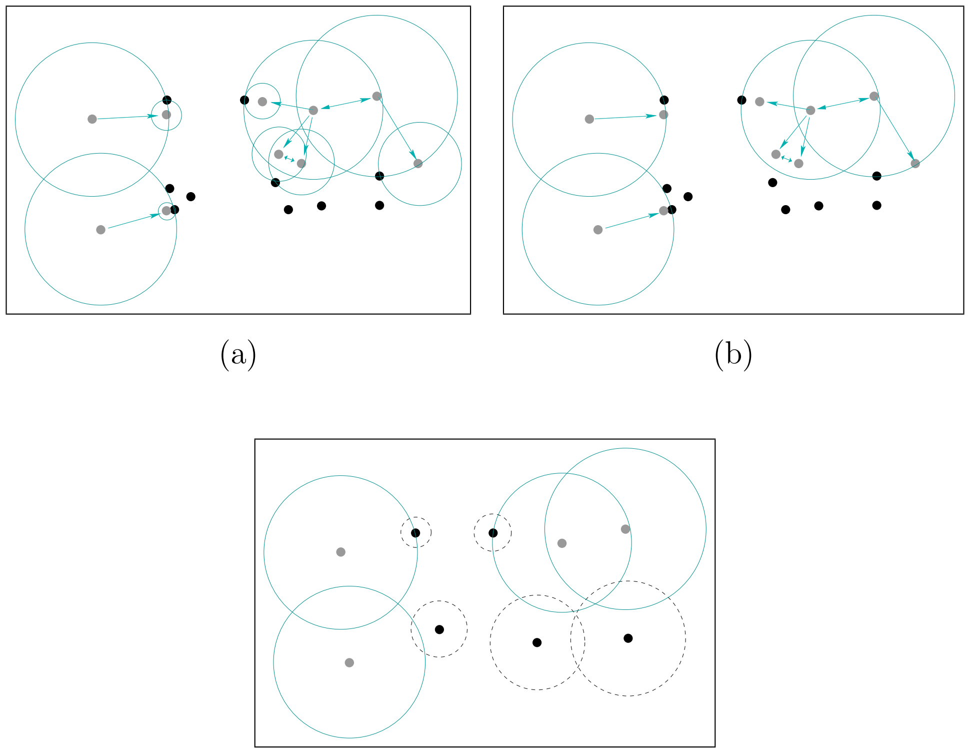

(a) (b) (c)

Figure 4.2: An illustration of the CCCDs (with the grey points representing the points from the target class) in a two-class setting. (a) All covering balls and the digraph $D_j = (\mathcal{V}_j, \mathcal{A}_j)$ and (b) the balls that constitute a class cover for the target class and are centered at points which are the elements of the dominating set $S_j \subseteq V_j$. (c) The dominating sets of both classes and their associated balls which establish the class covers. The class cover of grey points is the union of solid circles, and that of black points is the union of dashed circles.

$n_{1-j})d)$ *time.*

Proof. The algorithm is polynomial time reducible to a greedy minimum set cover algorithm which finds an approximate solution with size at most $\mathcal{O}(\log n_j)$ times of the optimum solution (Cannon and Cowen, 2004; Chvatal, 1979). We first calculate the distance matrices $\mathcal{M}_j$ and $\mathcal{M}_{j,1-j}$ which take $\mathcal{O}(n_j^2 d)$ and $\mathcal{O}(n_j n_{1-j} d)$ time, respectively. Constructing the digraph D_j requires computing $l(x)$ and $u(x)$ in Equation (4.1) for all $x \in \mathcal{X}_j$, taking $\mathcal{O}(n_j^2 + n_j n_{1-j})$ time in total. Then, we set the arc set $\mathcal{A}_j$ in $\mathcal{O}(n_j^2)$ time. Finally, the algorithm finds a solution for the digraph D_j in

$\mathcal{O}(n_j^2)$ time, hence the total running time of the algorithm is $\mathcal{O}(n_j(n_j + n_{1-j})d)$. ■

When $\mathcal{X}_{1-j}$ is the target class, observe that the time complexity is $\mathcal{O}(n_{1-j}(n_j + n_{1-j})d)$, and an approximate solution is of size at most $\mathcal{O}(\log n_{1-j})$ times the optimal solution by Theorem 1, since $n_{1-j} = |\mathcal{X}_{1-j}|$. A P-CCCD classifier consists of the covers of all classes, hence the total training time of finding CCCDs of a data set with two-class setting is $\mathcal{O}((n_j + n_{1-j})^2 d)$.

After establishing both class covers C_j and C_{1-j}, any new data point can be classified in $\mathbb{R}^d$ according to where it resides. Here, there are three cases according to the location of the given point, z, to be classified: z is (i) only in C_j or C_{1-j}, (ii) in both C_j and C_{1-j} or (iii) in neither of C_j and C_{1-j}. The case (i) is straightforward: z belongs to class $\mathcal{X}_j$ if $z \in C_j \setminus C_{1-j}$ or to class $\mathcal{X}_{1-j}$ if $z \in C_{1-j} \setminus C_j$. For cases (ii) and (iii), we need to find a way to decide the class of the point in a reasonable way. In fact, for all the cases, the estimated class of a given point z is determined by

$$\operatorname{argmin}_{C \in \{C_j, C_{1-j}\}} \left[\min_{x : B(x,r) \in C} \rho(z,x) \right] \tag{4.2}$$

where $\rho(z,x) = d(z,x)/r(x)$ (Marchette, 2010). The dissimilarity measure $\rho(x,z)$ indicates whether or not the point z is in the ball of radius $r(x)$ with center x, since $\rho(x,z) \leq 1$ if z is inside the (closure of the) ball and > 1 otherwise. The measure $\rho : \Omega \times \Omega \to \mathbb{R}_+$ is simply a scaled dissimilarity measure, since Euclidean distance between two points, $d(x,y)$, is divided (or scaled) with the radius, $r(x)$ or $r(y)$. This measure violates the symmetry axiom among metric axioms since $\rho(x,y) \neq \rho(y,x)$ whenever $r(x) \neq r(y)$. However, Priebe et al. (2003a) showed that the dissimilarity measure ρ satisfies the continuity condition, i.e., under the assumptions that both F_0 and F_1 are continuous and strictly separable ($\inf_{x \in \mathcal{X}_j, y \in \mathcal{X}_{1-j}} d(x,y) = \delta > 0$), P-CCCD classifiers are consistent; that is, their misclassification error approaches to the Bayes optimal classification error as $n_0, n_1 \to \infty$. The measure ρ favors points with bigger radii; that is, for example, for a new point z equidistant to two points, the point with bigger radius is closer in terms of this scaled dissimilarity measure; for example, $\rho(x,z) < \rho(y,z)$ when $d(x,z) = d(y,z)$ and $r(x) > r(y)$. The radius $r(x)$ can be viewed as an indicator of the density around the point x. Thus, a point x with bigger radius might suggest that the point z is more likely be drawn from the same distribution (or class) where x is drawn (i.e., from the denser class).

4.3.2 Classification with Random Walk CCCDs

For P-CCCDs, the class covers defined by CCCDs were "pure" of non-target class points; that is, no member of the non-target class was allowed inside the cover of the target class. As in Figure 4.1, the ball centered at the point x cannot expand any further since its radius is restricted by the distance to the closest non-target class point. This strategy may cause the cover to overfit or be sensitive to noise or outliers in the non-target class. By allowing some neighboring non-target class points inside the cover and some target class points outside the cover, the random walk CCCDs (RW-CCCDs) catch as much target class points as possible with an adaptive strategy of choosing the radii (DeVinney et al., 2002). For $x \in \mathcal{X}_j$, $|\mathcal{X}_j| = n$ and $|\mathcal{X}_{1-j}| = m$, RW-CCCDs define a function on radius of a ball given by

$$\begin{aligned} R_x(r) &= R_x(r; \mathcal{X}_j, \mathcal{X}_{1-j}) \\ &:= \frac{n_1}{n_0} |\{z \in \mathcal{X}_j : d(x,z) \leq r\}| - |\{z \in \mathcal{X}_{1-j} : d(x,z) \leq r\}|. \end{aligned} \tag{4.3}$$

where second and third arguments in $R_x(r; \mathcal{X}_j, \mathcal{X}_{1-j})$ are suppressed when there is no ambiguity. The function $R_x(r)$ can be viewed as a one-dimensional random walk. When the ball centered at $x \in \mathbb{R}^d$ expands, it hits either a target class point or a non-target class point which increases or decreases the random walk by one unit, respectively. The ratio n_1/n_0 is included in the first term as to avoid the bias resulted by unequal sample sizes (i.e., class imbalance). An illustration is given in Figure 4.3 for the case $n_0 = n_1$. The function $R_x(r)$ aims to find such radii that it contains a few non-target class points and sufficiently many target class points. In addition, we also want to avoid balls with large radii. Hence, the radius of x is the value maximizing $R_x(r)$ with an additional penalty function $P_x(r)$ which biases toward small radii:

$$r_x := \text{argmax}_{r \in \{d(x,z) : z \in \mathcal{X}_j \cup \mathcal{X}_{1-j}\}} R_x(r) - P_x(r). \tag{4.4}$$

Although a penalty function seems fit, DeVinney (2003) pointed out that the choice of

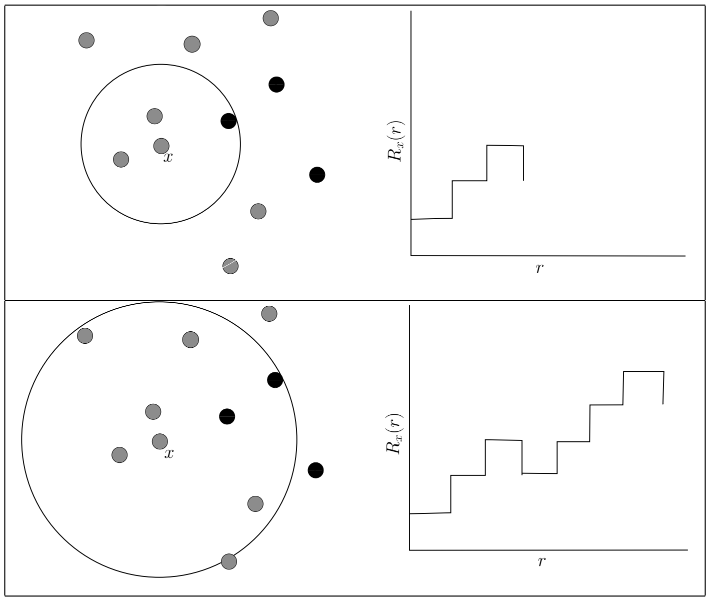


Figure 4.3: Two snapshots of $R_x(r)$ associated with the ball B_x centered at x for $m = n$.

$P_x(r) = 0$ usually works sufficiently well in practice. As in P-CCCDs, the radius of a ball represents the density of its center's neighborhood. Maximizing $R_x(r)$ determines the best possible radius. Moreover, unlike P-CCCDs, the balls of RW-CCCDs are closed balls: $B(x, r(x)) = \{z \in \mathbb{R}^d : d(x, z) \leq r(x)\}$.

Similar to P-CCCDs, finding a cover, or a dominating set, of a RW-CCCD is an NP-hard problem. However, RW-CCCDs find the MDSs in a slightly different fashion. We first locate the vertex x^* (a target class point) which has maximum of some score, T_{x^*}, and remove all target and non-target class points covered with the ball of this vertex, B_{x^*}. In the next iteration, we recalculate the radii of remaining target class points, find the next point with the maximum score and continue until all target class points are covered. This greedy method of finding dominating set(s) S_j of RW-CCCDs is given in Algorithm 8. The resulting dominating set S_j has approximate minimum cardinality. For each target class point $x \in \mathcal{X}_j$, the score T_x is associated with $R_x(r_x)$ and is given by

$$T_x = R_x(r_x) - \frac{r_x n_u}{2d_m(x)} \tag{4.5}$$

where n_u is the number of uncovered target class points in the current iteration, and $d_m(x) = \max_{z \in \mathcal{X}_j} d(x,z)$. The term which is linear in r_x of the right hand side of Equation (4.5) is similar to $P(r)$ in Equation (4.4): it biases the scores toward choosing dominating points with smaller radii. On the other hand, Algorithm 8 is likely to choose dominating points with radius $r = 0$. These points only dominate themselves but they are thought of being not covered since their balls have radii $r = 0$.

Algorithm 8 The greedy algorithm for finding an approximate MDS for RW-CCCDs of points $\mathcal{X}_j$ from the target class given non-target class points $\mathcal{X}_{1-j}$.

```
Input: Target class points $\mathcal{X}_j$ and non-target class points $\mathcal{X}_{1-j}$
Output: An approximate minimum cardinality ball cover $C_j$
 1: $H_0 = \mathcal{X}_j$, $H_1 = \mathcal{X}_{1-j}$ and $S_j = \emptyset$
 2: $\forall x \in \mathcal{X}_j$, $d_m(x) = \max_{z \in \mathcal{X}_j} d(x,z)$
 3: while $H_0 \neq \emptyset$ do
 4:     $n_u = |H_0|$
 5:     for all $x \in \mathcal{X}_j$ do
 6:         $r(x) = \mathrm{argmax}_r\, R_x(r; H_0, H_1)$ for $r \in \{d(x,z) : z \in H_0 \cup H_1\}$
 7:     end for
 8:     $x^* = \mathrm{argmax}_{x \in H_0}\, R_x(r(x); H_0, H_1) - \frac{r(x) n_u}{2 d_m(x)}$
 9:     $S_j = S_j \cup \{x^*\}$
10:     $\bar{N}(v^*) \leftarrow \{u \in \mathcal{V}(D) : (v^*, u) \in \mathcal{A}(D)\} \cup \{v^*\}$
11:     $H_0 = H_0 \setminus (\bar{N}(x^*) \cap \mathcal{X}_j)$ and $H_1 = H_1 \setminus (\bar{N}(x^*) \cap \mathcal{X}_{1-j})$
12: end while
13: $C_j := \cup_{s \in S_j} B(s, r(s))$
```

Algorithm 8 is similar to Algorithm 7, however after each iteration, a point is added to the set S and the random walk $R_x(r)$ is recalculated for all uncovered $x \in H_0$. Hence, we need an additional sweep on the training set which makes Algorithm 8 run in cubic time.

Theorem 2 *Algorithm 8 finds covers C_j of the target class $\mathcal{X}_j$ in $\mathcal{O}((n_j + d + \log(n_j + n_{1-j}))(n_j + n_{1-j})^2)$ time.*

Proof. In Algorithm 8, the matrix of distances between points of training set $\mathcal{X}_j \cup \mathcal{X}_{1-j}$ should be computed since, for all $x \in \mathcal{X}_j \cup \mathcal{X}_{1-j}$, the entire data set is swept to maximize $R_x(r)$. This takes $\mathcal{O}((n_j + n_{1-j})^2 d)$ time. The algorithm runs until all target class points are covered, but for each iteration, the random walk $R_x(r)$ is recalculated. The maximum $R_x(r_x)$ could be found by sorting the distances for all $x \in H_0$ which could be done prior to the while loop. This sorting takes $\mathcal{O}((n_j + n_{1-j})^2 \log(n_j + n_{1-j}))$ time. Since H_0 and H_1 are updated at each iteration, we can just erase the distances corresponding to points covered by $\bar{N}(x^*)$ which does not change the order of sorted list provided before the while loop. Hence, argmax $R_x(r_x)$ is found and the covered points erased in $\mathcal{O}((n_j + n_{1-j})^2)$ time. The while loop iterates n_j times in the worst case, and hence the algorithm runs in a total of $\mathcal{O}((n_j + d + \log(n_j + n_{1-j}))(n_j + n_{1-j})^2)$ time. ■

Note that Algorithm 8 finds a cover of $\mathcal{X}_{1-j}$ in $\mathcal{O}((n_j + d + \log(n_j + n_{1-j}))(n_j + n_{1-j})^2)$ time which makes a RW-CCCD classifier trained in $\mathcal{O}((n_j + n_{1-j})^3)$ time for $d < n$ and $\log(n_j + n_{1-j}) < n_j$. RW-CCCD classifiers are much better classifiers that potentially avoid overfitting, but with a cost of being much slower compared to the P-CCCD classifiers.

P-CCCD classifiers tend to overfit (DeVinney, 2003). In RW-CCCDs, covering balls allow some points of $\mathcal{X}_{1-j}$ inside C_j to increase average classification performance. In that case, Algorithm 8 cannot be reduced to a minimum set cover problem since the definition of sets change after adding a single point to the dominating set. Hence, the upper bound $\mathcal{O}(\log n_j)$ does not apply to RW-CCCDs. However, we expect to get bigger balls in RW-CCCDs compared the ones in P-CCCDs which intuitively suggests that the covers of RW-CCCDs are lower in cardinality. We conduct empirical studies to show that RW-CCCDs, in fact, produce dominating sets with lower size compared to P-CCCDs in some cases.

In RW-CCCD, once the class covers (or dominating sets) are determined, the scaled dissimilarity measure in Equation (4.2) is a good choice for estimating the class of a new point z. However, DeVinney (2003) incorporates the scores of each ball to produce better performing class covers in classification. Hence, the class of a new point z is determined by

$$\operatorname{argmin}_{C \in \{C_j, C_{1-j}\}} \left[\min_{x : B(x,r) \in C} \rho(z,x)^{T_x^e} \right]$$

where $\rho(z,x)$ is defined as in Equation (4.2). Here, $e \in [0,1]$ controls at what level the score T_x is incorporated. We observe that for $d(z,x) < r(x)$, $\rho(z,x) = d(z,x)/r(x)$ decreases as T_x increases. Hence, if a new point z is in both covers, $z \in C_j \cap C_{1-j}$, the score T_x is a good indicator to which class the new point z belongs since the bigger the T_x, the more likely the ball contains more target class points. For $e = 1$, we fully incorporate each score T_x of covering balls and with $e = 0$, we ignore the scores. By introducing a value for the parameter e in $(0,1)$, it is possible to further improve the performance of RW-CCCD classifiers.

4.4 Balancing the Class Sizes with CCCDs

The CCCD classifiers substantially reduce the number of majority class observations in a data set. The reason is that balls of majority class members are more likely to catch neighboring points of the same class. The greedy algorithm given in Algorithm 1 selects vertices with the largest closed neighborhood. Similarly, Algorithm 8 selects vertices so that their balls are as dense as possible (i.e., target class points are abundant in the balls) with some contaminating non-target class points. Both algorithms choose balls with a large number of target class points, and hence substantially reduce the data set (in particular, majority class points). Points of the MDS correspond to the centers of balls that establish the class covers. Hence CCCD classifiers can also be viewed as *prototype selection* methods where the objective is finding a set of points, or *prototypes*, S_j; from the training set to preserve or increase the classification performance while substantially reducing the sample size. However, the radii of dominating set(s) are also stored and used in the classification process.

In Figure 4.4, we illustrate the behavior of balls associated with P-CCCDs and RW-CCCDs. Note that in both families of digraphs, balls of the majority class tend to be larger and hence are more likely to catch more majority class points. Since the majority class has much more members than the minority class, balls of the majority points are more likely to catch the neighboring majority points. CCCD classifiers keep the information of ball centers and their associated radii. Larger cardinality of the majority class allows the construction of bigger balls and hence, larger values of radii are more likely to correspond to larger number of catched class members. As a result, CCCDs balance the data set and, at the same time, preserve the information of the local density by retaining the radii. The data set becomes balanced since the center of balls are the points of the new training data set which will be employed later in classification.

The loss of information in undersampling schemes are of course inevitable, however it is possible to preserve a portion of that discarded information by other means. EasyEnsemble is an ensemble classifier used for that very purpose; however, it needs multiple classifiers to be employed. Each classifier is trained on a different balanced subset of the original training data set, and hence the ensemble classifier preserves the information on the entire data set given by a collection of unbiased classifiers. On the other hand, CCCDs achieve the same goal by transforming the density around points into the radii. CCCDs resemble cluster based resampling methods in that regard. Instead of randomly sampling the data set, cluster based sampling schemes divide each class into clusters, and then, oversample the minority class or undersample the majority class proportional to each subclass. Covering balls of CCCDs have a similar purpose which has also been discussed in Priebe et al. (2003b). They use the covering balls of the MDSs to explore the latent subclasses of each class of gene expression data sets. In fact, the balls of CCCDs may correspond to clusters. Hence, sets of points associated with each cluster is undersampled to a single point (i.e., a prototype or a dominating point), and the information on the cluster is provided by the radius which represents the density of that cluster. The bigger the radius, the more influence a prototype has over the domain. In P-CCCDs, the radii may be sensitive to noise, but RW-CCCDs ignore noisy points to avoid overfitting. Moreover, in RW-CCCDs, we

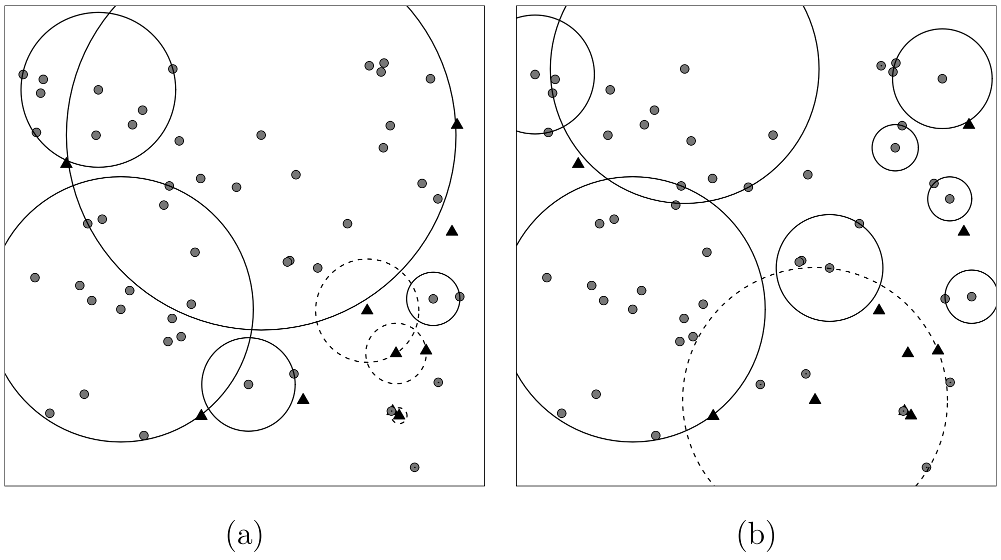


Figure 4.4: An illustration of the covering balls associated with majority and minority (a) P-CCCD ($\theta = 0.0001$) and the corresponding (b) RW-CCCDs of an imbalanced data set in a two-class setting where majority and minority class points are represented by grey dots and black triangles, respectively.

have an additional statistic provided by each cluster, the score given in Equation (4.5) based on the random walk. We use both the radii and these scores to define the RW-CCCD classifiers, and thus achieve better performing classifiers with more reduction and less information loss.

We approach the problem of class imbalance from the perspective of class overlapping problem as well. Several researchers on class imbalance revealed that overlap between the class supports degrade the classification performance of imbalanced data sets even more (Batista et al., 2004, 2005; Galar et al., 2012; Prati et al., 2004). Let $E \subset \mathbb{R}^d$, and let $s(F_0)$ and $s(F_1)$ be the supports of the classes $\mathcal{X}_0$ and $\mathcal{X}_1$, respectively. We define E as the overlapping region of these two class supports, $E := s(F_0) \cap s(F_1)$. Moreover, let $q(E) := |\mathcal{X}_{1-j} \cap E| / |\mathcal{X}_j \cap E|$ be ratio of class sizes restricted to the region $E \subset \mathbb{R}^d$. We say $q(E)$ is the "local" imbalance ratio with respect to E. Also, let the "global" imbalance ratio be $q = q(\mathbb{R}^d) = n_{1-j}/n_j$. Throughout this chapter, in both simulated and real data examples, we study and discuss the local imbalance ratio $q(E)$ restricted to the overlapping region E and the global imbalance ratio q.

We specifically illustrate the performance of several classifiers for various levels of class imbalance (local or global) and class overlapping, and assess the performance of CCCD classifiers compared to k-NN, SVM and C4.5 classifiers.

4.5 Monte Carlo Simulations and Experiments

In this section, we compare the CCCD-based classifiers, namely P-CCCD and RW-CCCD, with k-NN, support vector machines (SVM) and C4.5, on simulated data sets. These classifiers are listed in Table 4.1. We employ the `cccd`, `e0171` and `RWeka` packages in R to classify test data sets with the P-CCCD, SVM (with Gaussian kernel) and C4.5 classifiers, respectively (Marchette, 2013; Meyer et al., 2014; R Core Team, 2015).

For each of four classification methods other than C4.5, we assign the optimum parameter values which are the best performing values among all considered parameters. For example, an optimum P-CCCD parameter θ is found in a preliminary (pilot) Monte Carlo simulation study associated with the main simulation setting (i.e., the same setting of the main simulation). We use the area under curve (AUC) measure to evaluate the performance of the classifiers on the imbalanced data sets (López et al., 2013). AUC measure is often used on imbalanced real data classes. This measure has been shown to be better than the correct classification rate in general (Huang and Ling, 2005). In the pilot study, we perform a Monte Carlo simulation with 200 replications and count how many times a θ value has the maximum AUC among $\theta = 0.0, 0.1, \cdots, 1.0$ in 200 trials. Note that, since $\theta \in (0, 1]$, we denote $\theta = \epsilon$ (machine epsilon) as $\theta = 0$ for the sake of simplicity. For each replication of the pilot simulation, we (i) classify the test data set with all θ values, (ii) record the θ values with maximum AUC and (iii) update the count of the recorded θ values. Finally, we appoint the one that has the maximum count (the best performing θ) as the θ^*, the optimum θ. Then, we use θ^* as the parameter of P-CCCD classifier in our main simulation. The parameters of optimal k-NN, SVM and RW-CCCD classifiers are defined similarly. SVM methods often incorporate both a kernel parameter γ and a constrained violation cost C. We only optimize γ since the selection of an optimum C parameter will be more crucial for cost-sensitive SVM methods. Moreover, we consider two versions of the C4.5 classifier where both incorporate Laplace smoothing. The first tree classifier, C45-LP, prunes the decision tree with %25 confidence level but the second classifier, C45-LNP, does not use pruning at all.

We first consider a simulation setting similar to the one in DeVinney et al. (2002) where CCCD classifiers showed relatively good performance compared to the k-NN classifier. Here, we simulate a two-class setting where observations from both classes are drawn from separate multivariate uniform distributions: $F_0 = U(0,1)^d$ and $F_1 = U(0.3, 0.7)^d$ for $d = 2, 3, 5, 10$. Notice that $s(F_1) \subset s(F_0)$; i.e., $E = s(F_1)$. We perform Monte Carlo replications where on each replication, we train the data with equal sizes of observations ($n = n_0 = n_1$) from each class for $n = 50, 100, 200, 500$. On each replication, we record the AUC measures of the classifiers on the test data set with 100 observations from each class, resulting a test data set of size 200. We simulate test data sets until AUCs of all classifiers achieve a standard error below 0.0005. Average of AUCs of all classifiers in Table 4.1 are given in Figure 4.5 for all (n, d) combinations. Additionally, in Figure 4.6, we report the θ values of best performing P-CCCD classifiers in our pilot simulation study for all (n, d) combinations. In Figure 4.6, there are separate histograms for each combination. Each histogram represents the number of times a θ value has the maximum AUC. Also in Figure 4.7, we report the e values of the best performing RW-CCCD classifiers of the same pilot simulation study for $e = 0, 0.1, \cdots, 1.0$.

We start by investigating the effect of θ and e on CCCD classifiers. The relationship between θ, n and d can also be observed in Figure 4.6. The higher the θ value, the better the performance of P-CCCD classifier with increasing d and decreasing n. This may indicate that balls with $\theta = 0$ (i.e., $\theta = \epsilon$) represent the density around their centers better for low dimensional data sets. However, with increasing dimensionality and lower class sizes, the set of points gets sparser in $\mathbb{R}^d$. In the case of RW-CCCD, classifiers with high e values are either better or comparable to those with lower e

Method	Description
P-CCCD	P-CCCD with the optimum θ (in the pilot study) among $\theta = 0, 0.1, \cdots, 1.0$
RW-CCCD	RW-CCCD with the optimum e (in the pilot study) among $e = 0, 0.1, \cdots, 1.0$
k-NN	k-NN with optimum k (in the pilot study) among $k = 1, 2, \cdots, 30$
SVM	SVM with the radial basis function (Gaussian) kernel with the optimum γ (in the pilot study) among $\gamma = 0.1, 0.2, \cdots, 3.9, 4.0$ (Joachims, 1999)
C45-LP	C4.5 with Laplace smoothing and reduced error pruning (%25 confidence)
C45-LNP	C4.5 with Laplace smoothing and no pruning

Table 4.1: The description of classifiers employed.

values. The scores T_x of covering balls are definitely beneficial to the performance of the RW-CCCD classifiers, however with increasing n and decreasing d (especially for $n = 500$ and $d = 2$) RW-CCCD with lower e is better since the radii successfully represent the density around the prototype points due to the high number of observations in the data set.

Figure 4.5 illustrates the AUCs of all classifiers along with the Bayes optimal performance (the best possible performance) given with the dashed line. Comparing the performance of CCCD classifiers with other classification methods, we observe that RW-CCCD and P-CCCD classifiers outperform the k-NN classifier when the support of one class is entirely embedded inside that of the other class. These results are similar to the conclusions of DeVinney et al. (2002): with increasing dimensionality, the difference between k-NN and CCCD classifiers becomes more apparent, i.e., CCCD classifiers have nearly 0.20 AUC more than k-NN. On the other hand, the SVM classifier has about 0.05 more AUC than P-CCCD and RW-CCCD classifiers, especially for lower class sizes. Although, both versions of CCCD classifiers outperform the k-NN and C4.5 classifiers with increasing dimensionality, the gap between these two classifiers and CCCD classifier is getting narrower with increasing class sizes. The RW-CCCD classifier is slightly better than the P-CCCD classifier for lower n. In addition, C45-LNP achieves slightly better results than C45-LP.

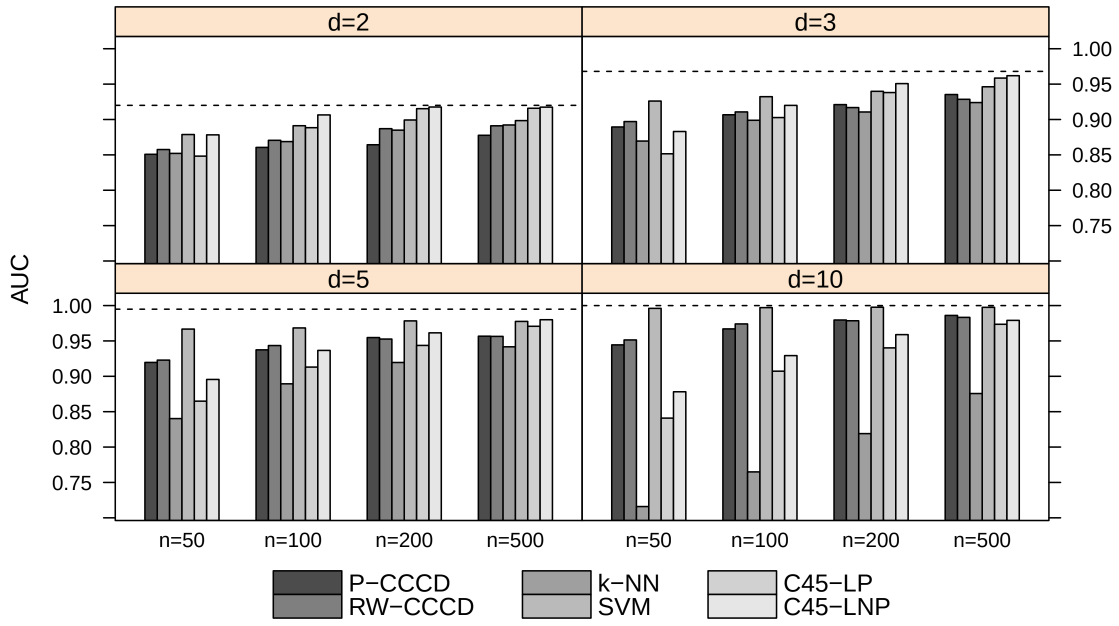


Figure 4.5: AUCs in the two-class setting, $F_0 = U(0,1)^d$ and $F_1 = U(0.3,0.7)^d$ under various simulation settings, with $d = 2,3,5,10$ and equal class sizes $n = n_0 = n_1 = 50,100,200,500$.

In the setting presented in Figure 4.5, apparently, two classes overlap on the region $E = s(F_1) = [0.3,0.7]^d$ which is the entire support of the class $\mathcal{X}_{1-j}$. For equal class sizes, $q = n_1/n_0 = 1$ but $q(E) \approx (1/0.4)^d = \operatorname{Vol}(s(F_1))/\operatorname{Vol}(s(F_0))$, where $\operatorname{Vol}(\cdot)$ is the volume functional. The classes are clearly imbalanced in E, although $n_0 = n_1$. Hence, class $\mathcal{X}_0$ becomes the minority and class $\mathcal{X}_1$ becomes the majority class with respect to E. However, readjusting the class sizes n_1 and n_0 might change the performance of P-CCCD and RW-CCCD classifiers compared to the k-NN and C4.5 classifiers. Therefore, we conduct another simulation study with classes from the same uniform distributions, but we set $n_1 = 50$ and $n_0 = 200$ for $d = 2,3$, and $n_1 = 50$ and $n_0 = 1000$ for $d = 5,10$. In this experiment, we simulated 4 times more $\mathcal{X}_0$ class members than $\mathcal{X}_1$ for $d = 2,3$, and 20 times more for $d = 5,10$. Results of this second experiment is given in Figure 4.8. k-NN and C4.5 classifiers outperform P-CCCD classifier in all d cases and has comparable AUC with SVM. However, only

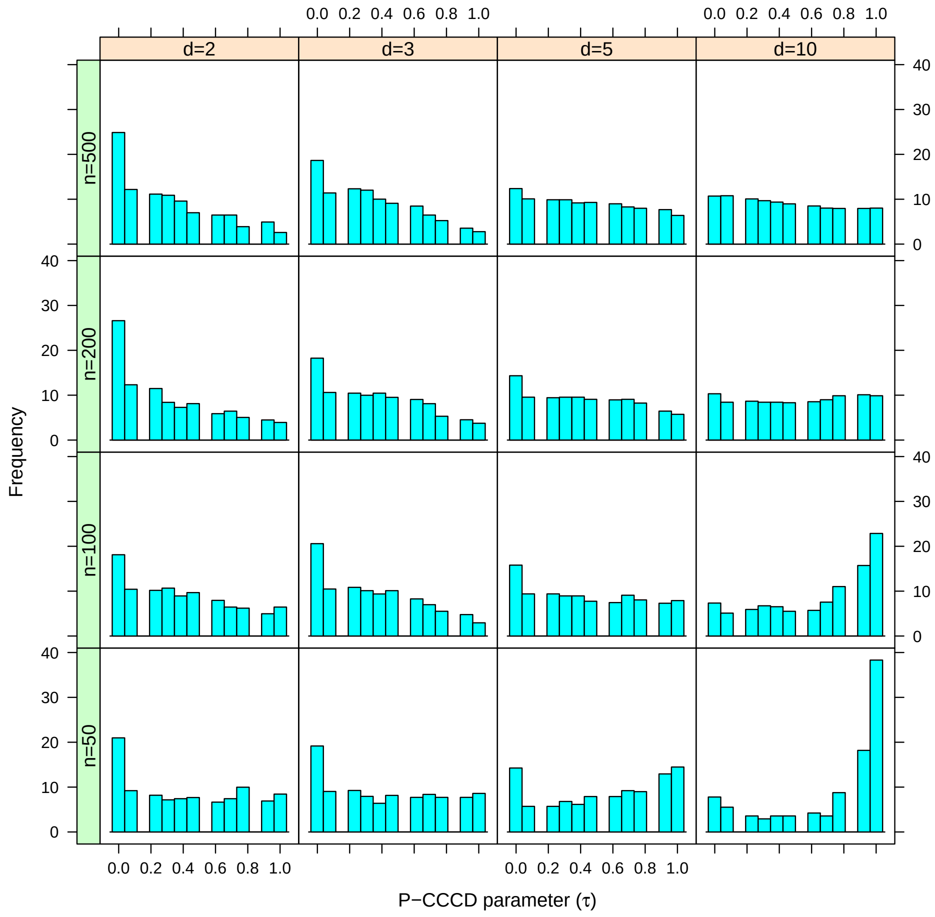


Figure 4.6: Frequencies of the best performing θ values among $\theta = 0.0, 0.1, \cdots, 1.0$ in our pilot study (this is used to determine the optimal θ used in P-CCCD). The simulation setting is same as to the one presented in Figure 4.5.

for $d = 2, 5$, RW-CCCD classifier achieves considerably more or comparable AUC compared to other classifiers. In this example, k-NN classifiers have nearly 0.05 more AUC than P-CCCDs, and also RW-CCCDs have, in general, 0.05 more AUC than k-NN classifiers.

Results from Figures 4.5 and 4.8 seem conflicting to each other, even though $E = s(F_1)$. In the simulation setting of Figure 4.8, we draw more samples from the class $\mathcal{X}_0$ to balance the class sizes with respect to E. In fact, the effect on the difference of AUCs between CCCD, k-NN and C4.5 classifiers depends heavily on the

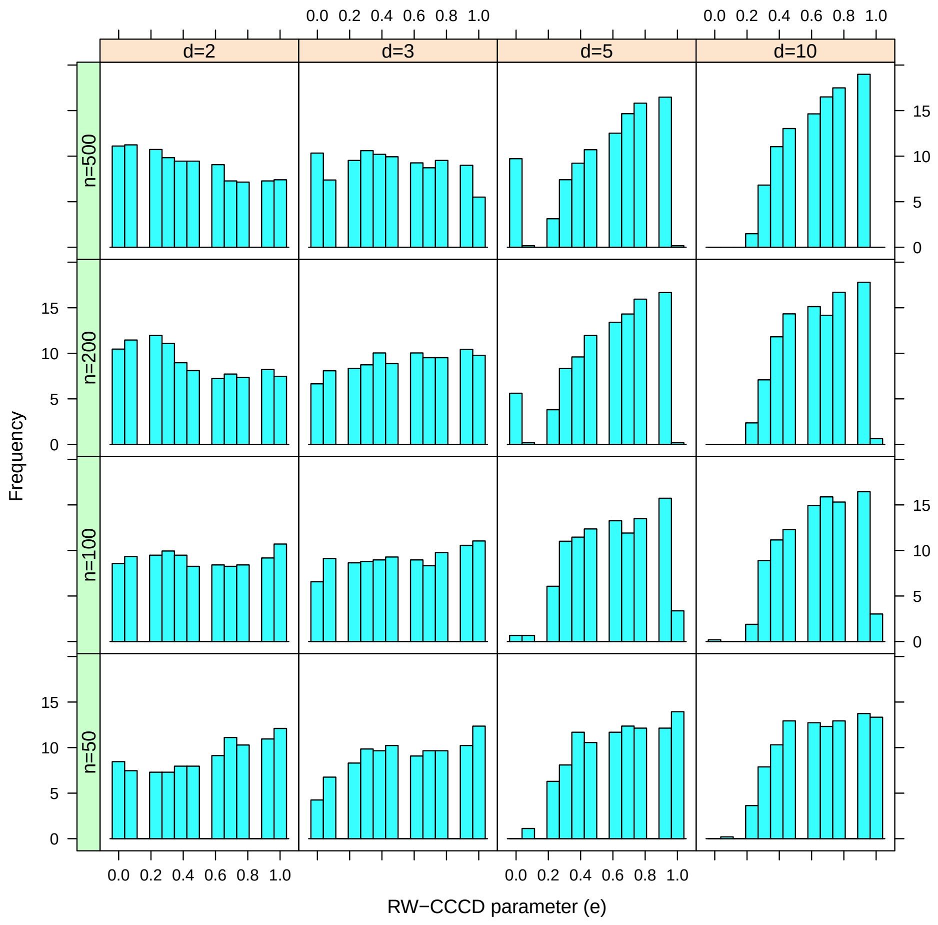


Figure 4.7: Frequencies of the best performing e values among $\theta = 0.0, 0.1, \cdots, 1.0$ in our pilot study (this is used to determine the optimal e used in RW-CCCD). The simulation setting is same as to the one presented in Figure 4.5.

local class imbalance restricted to the overlapping region E. The classes in region E are less imbalanced in setting of Figure 4.8 than in the setting of Figure 4.5. Observe that $q(E) \approx (1/0.4)^d/4$ when $(n_1, n_0) = (50, 200)$, $q(E) \approx (1/0.4)^d/20$ when $(n_1, n_0) = (50, 1000)$, and $q(E) \approx (1/0.4)^d$ in $(n_1, n_0) = (50, 50)$. Hence, d does also affect the balance between classes. With increasing d, the region E gets smaller in volume compared to $s(F_0)$ and, as a result, fewer points of the class $\mathcal{X}_0$ falls in E. Thus, we need to draw more samples from $\mathcal{X}_0$ as dimensionality increases, in order to balance the classes with respect to E. These results suggest that, the more

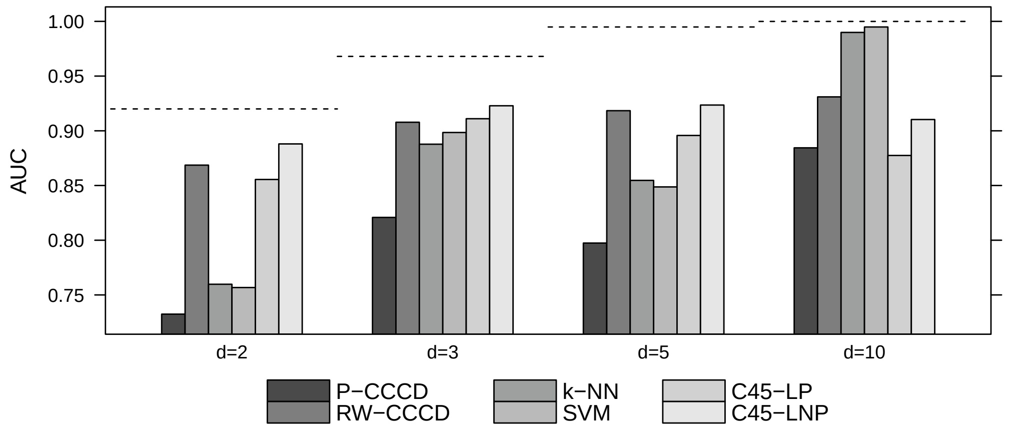


Figure 4.8: AUCs in a two-class setting, $F_0 = U(0,1)^d$ and $F_1 = U(0.3, 0.7)^d$ with fixed $n_0 = 200$ and $n_1 = 50$ in $d = 2, 3$, and with fixed $n_0 = 1000$ and $n_1 = 50$ in $d = 5, 10$.

imbalanced the data set in overlapping region E, the worse the performance of k-NN and C4.5 classifiers while CCCD classifiers preserve their classification performance. So, CCCD classifiers exhibit robustness (to the class imbalance problem). On the other hand, in Figure 4.5, we observe that the AUC of k-NN classifier approaches to the AUC of CCCD classifiers with increasing class sizes. Because, when q and $q(E)$ are fixed, the classification performance still depends on individual values of n_0 or n_1. This result is in line with the results of Japkowicz and Stephen (2002) who reported that the effect of class imbalance on the classification performance diminishes if both class sizes are sufficiently large. Furthermore, SVM classifier performs better than all classifiers in Figure 4.5, and performs worse than RW-CCCD classifiers only for $d = 2, 5$ in Figure 4.8. This might be an indication that SVM classifier is also not affected by the local class imbalance with respect to E, and performs usually better than both P-CCCD and RW-CCCD classifiers if the support of one class is inside the other. For the C4.5 classifier, on the other hand, it is known for quite some time that the pruning is detrimental for classifying imbalanced data sets (Cieslak and Chawla, 2008). In any case, C45-LNP has more AUC than C45-LP in all simulation settings.

In a two-class setting with an overlapping region E, we should expect CCCD classifiers to outperform k-NN classifiers in cases of (global or local) class imbalance. Let $F_0 = U(0,1)^d$ and $F_1 = U(\delta, 1+\delta)^d$ for $\delta, q = 0.05, 0.10, \cdots, 0.95, 1.00$; $d = 2, 3, 5, 10$; $n = 400$ and $n_1 = qn_0$. Here, the shifting parameter δ controls the level of overlap. The class supports get more overlapped with decreasing δ. Since $E = (\delta, 1)^d$ and the supports of both classes are unit boxes, observe that $q(E) \approx q$. The closer the value of q to 1, more balanced the classes are. We aim to address the relationship between the classifiers for various combinations of overlapping and global class imbalance ratios.

Figure 4.9 illustrates the difference between AUCs of CCCD and other classifiers (k-NN, SVM and C4.5) in separate heat maps for $d = 2, 3, 5, 10$. We use the unpruned C4.5 classifier C45-LNP, since it tends to perform better for imbalanced data sets, and we refer to C45-LNP as C4.5 for simplicity. Each cell of a single heat map is associated with a combination of δ and q values. Lighter tone cells indicate that CCCD classifiers are better than the other classifiers in terms of AUC, and vice versa for the darker tones. When the classes are imbalanced and moderately overlapping, RW-CCCD classifier has at least 0.05 more AUC than all other non-CCCD classifiers but P-CCCD classifier is only better than all others provided that $d = 10$. If the classes are balanced or their supports are not considerably overlapping, there seem to be no visible difference between CCCD and the other classifiers. Thus, the other classifiers suffer from the imbalance of the data while CCCD classifiers show robustness to the class imbalance. But more importantly, this difference is getting more apparent with increasing dimensionality. When d is high, fewer points of the minority class fall in E although $q(E)$ is fixed. Even though the classes are imbalanced, if the minority class have substantially small size, the class imbalance problem becomes more detrimental (Japkowicz and Stephen, 2002). Under the conditions that the data set has substantial imbalance and overlapping, AUC of RW-CCCD classifier is followed, in order by, the AUC of C4.5, SVM and k-NN classifiers.

Unlike the comparison of CCCD and SVM classifiers in Figures 4.5 and 4.8, SVM

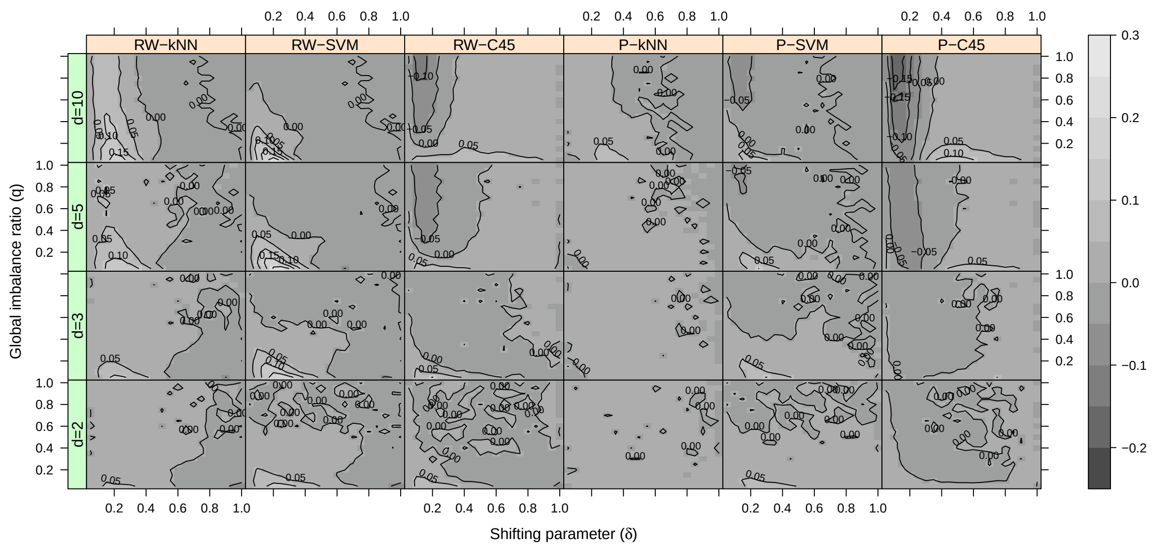


Figure 4.9: Differences between the AUCs of CCCD and other classifiers. For example, the panel titled with "RW-kNN" presents AUC(RW-CCCD)-AUC(kNN). In this two-class setting, classes are drawn from $F_0 = U(0,1)^d$ and $F_1 = U(\delta, 1+\delta)^d$ with $d = 2, 3, 5, 10$. Each cell of the grey scale heat map corresponds to a single combination of simulation parameters $\delta, q = 0.05, 0.1, \cdots, 0.95, 1.00$ with $n_0 = 400$ and $n_1 = qn_0$.

classifier has less AUC than CCCD classifiers with low δ and low q values in Figure 4.9. In this setting, n_0 is fixed to 400 and the lowest value of n_1 is 20. Compared to our experiments in Figures 4.5 and 4.8, this setting produces highly imbalanced data sets (one class has far more observations than the other, $n_1 << n_0$). Akbani et al. (2004) conducted a detailed investigation and listed some reasons of SVM classifier being sensitive to highly imbalanced UCI data sets (Bache and Lichman, 2013). They did not, however, address the problem of overlapping class supports but offered a modification to SMOTE algorithm in order to improve the robustness of SVM. On the other hand, especially for $d = 5$ and $d = 10$, SVM, k-NN and C4.5 classifiers have more AUC than CCCD classifiers with increasing q and decreasing δ. This may indicate that other other classifiers are better than CCCD classifiers for balanced classes.

The effects of class imbalance might also be observed when the class supports are well separated. If the class supports are disjoint, that is $s(F_0) \cap s(F_1) = \emptyset$, the AUC is fairly high. However, it might still be affected by the global imbalance level, q. Therefore, we simulate a data set with two classes where $F_0 = U(0,1)^d$ and $F_1 = U(1+\delta, 2+\delta) \times U(0,1)^{d-1}$. Figure 4.10 illustrates the results of this simulation study. Both class supports are d dimensional unit boxes as in the previous simulation setting, however they are now disjoint (separated along the first dimension). In addition, the parameter δ controls the smallest distance between the class supports where $\delta = 0.05, 0.10, \cdots, 0.45, 0.50$. With increasing δ, the points of class $\mathcal{X}_{1-j}$ move further away from the points of $\mathcal{X}_j$. Figure 4.10 illustrates the difference between AUCs of CCCD and other classifiers under this simulation setting.

In Figure 4.10, unlike the performance of CCCD classifiers in Figure 4.9, P-CCCD classifiers have more AUC than RW-CCCD classifiers. When classes are imbalanced and supports are close, P-CCCD classifiers outperform both SVM and k-NN classifiers for all d values, but RW-CCCD classifiers have nearly 0.03 more AUC than these classifiers only in $d = 10$. However, this is not the case with C4.5 classifier since none of the classifiers outperform C4.5; that is, C4.5 yields over 0.04 more AUC than CCCD classifiers. A well separated data set is more likely to be classified better with C4.5 tree classifier because a single separating line exists between the two class supports. Hence, C4.5 locates such a line and efficiently classifies points regardless of the distance between class supports as long as the distance is positive. On the other hand, the balls of P-CCCD classifiers establish appealing covers for the class supports because the supports do not overlap. P-CCCD classifiers establish covering balls, big enough to catch substantial amount of points from the same class. Similarly, RW-CCCD classifiers establish pure covers, and this is the result of the separation between class supports. However, P-CCCD classifiers achieve better classification performance than RW-CCCD classifiers. When the classes are well separated, the radii of a ball in random walk, say from class $\mathcal{X}_j$, is likely $\max_{z \in \mathcal{X}_j} d(z,x)$ but in P-CCCD classifiers, it is $\min_{z \in \mathcal{X}_{1-j}} d(z,x)$. In fact, the RW-CCCD classifiers are nearly equivalent to P-CCCD classifiers. Thus, when $\theta > 0$, P-CCCD classifiers are more likely to produce bigger balls than RW-CCCD classifiers, and potentially avoid overfitting.

In Figure 4.10, RW-CCCD classifiers have slightly or considerably less AUC than other classifiers when data sets are imbalanced and the supports are slightly far away from each other. The random walk contaminates the class cover with some non-target class points to improve the classification performance. However, since the classes are well separated and one class has substantially fewer points than the other, random walks are likely to yield balls to cover some points from the support of the non-target class, resulting in a degradation in the performance of RW-CCCD classifiers. On the other hand, P-CCCD classifiers outperform both k-NN and SVM classifiers for lower q and lower δ. The closer and more imbalanced the data, the better the performance of P-CCCDs than other classifiers. Although the classes do not overlap, the effect of class imbalance is still observed when the supports are close. When there is mild imbalance between classes, CCCD classifiers have either comparable or less AUC. In addition, note that the performances of SVM and k-NN classifiers deteriorate but P-CCCD classifiers preserve their AUC with increasing d. Let $E \subset \mathbb{R}^d$ be some region that contains points of both classes which are sufficiently close to the decision boundary. With increasing d, fewer minority class points are in this region, and hence fewer members of this class fall in E. As a result, the performance of both SVM and k-NN classifiers suffer from local class imbalance with respect to E.

Finally, we investigate the effect of dimensionality when classes are balanced (i.e., $q = 1$) and their supports are overlapping. In this setting, $F_0 = U(0,1)^d$ and $F_1 = U(\delta, 1+\delta)^d$. Here, let $q(E) \approx q = 1$, hence the classes are also locally balanced with respect to E as well as being globally balanced. Also, δ controls the level of overlap between two classes. However, we define δ in such a way that the overlapping ratio $\alpha \in [0,1]$ is fixed for all dimensions. When α is 0, the supports are well separated, and when α is 1, the supports of classes are the same, i.e., $s(F_0) = s(F_1)$. The closer α to 1, the more the supports overlap. Observe that $\delta \in [0,1]$ can be expressed in terms of the overlapping ratio α and dimensionality d:

$$\alpha = \frac{\mathrm{Vol}(s(F_0) \cap s(F_1))}{\mathrm{Vol}(s(F_0) \cup s(F_1))} = \frac{(1-\delta)^d}{2-(1-\delta)^d} \quad \Longleftrightarrow \quad \delta = 1 - \left(\frac{2\alpha}{1+\alpha}\right)^{1/d}. \tag{4.6}$$

Hence, we calculate δ for each (d, α) combination by the Equation (4.6). In Fig-

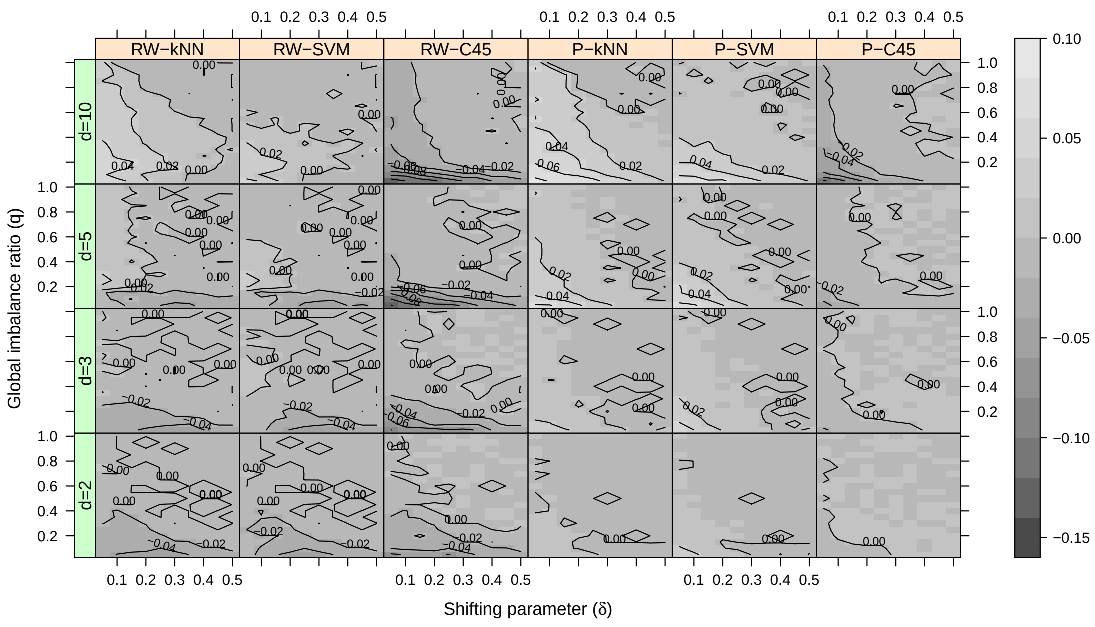


Figure 4.10: Differences between the AUCs of CCCD and other classifiers (see Figure 4.9 for details). In this two-class setting, classes are drawn from $F_0 = U(0,1)^d$ and $F_1 = U(1+\delta, 2+\delta) \times U(0,1)^{d-1}$ where $d = 2, 3, 5, 10$, $\delta = 0.05, 0.1, \cdots, 0.45, 0.50$ and $q = 0.05, 0.1, \cdots, 0.95, 1.00$ with $n_0 = 400$ and $n_1 = qn_0$. AUCs of all classifiers are over 88% since the class supports are well separated.

ure 4.11, each cell of the grey scale heat map corresponds to a single combination of simulation parameters $\alpha = 0.05, 0.1, \cdots, 0.95, 1.00$ and $d = 2, 3, 4, \cdots, 20$. In Figure 4.11, the differences between the AUCs of CCCD classifiers and other classifiers are up to 0.20. The k-NN and SVM classifiers have comparable performance with CCCD classifiers, or outperform both CCCD classifiers. However, C4.5 has more AUC with increasing d. Employing CCCD classifiers do not considerably increase the classification performance over other classifiers when classes are balanced.

4.6 Real Data Examples

In this section, we compare the performance of CCCD classifiers and all other classifiers on several data sets from UC Irvine (UCI) Machine Learning and KEEL repositories (Alcalá-Fdez et al., 2011; Bache and Lichman, 2013). To test the difference

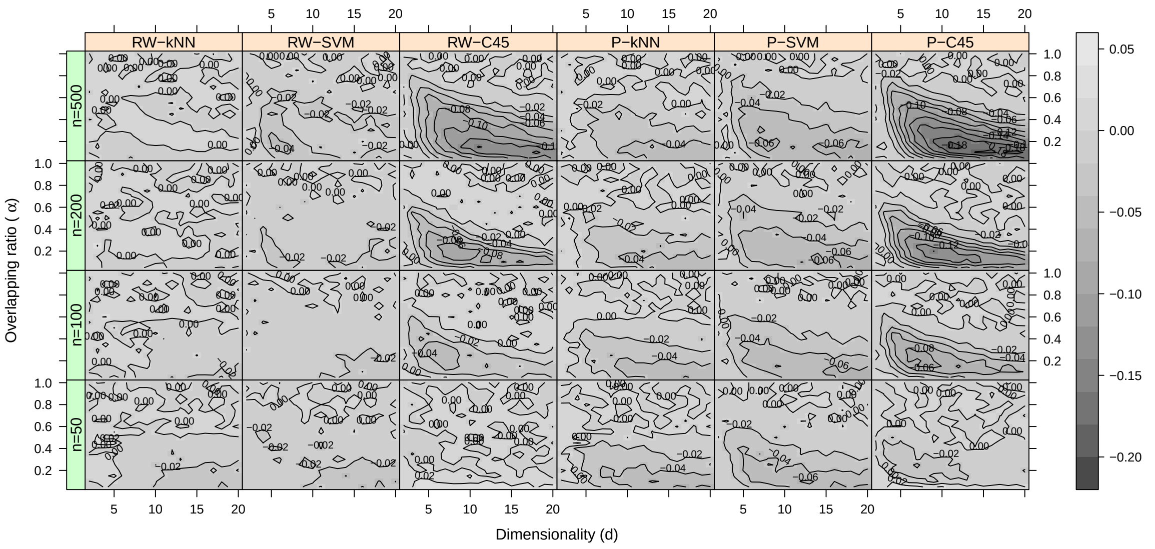


Figure 4.11: Differences between the AUCs of CCCD and other classifiers (see Figure 4.9 for details). In this two-class setting, classes are drawn from $F_0 = U(0,1)^d$ and $F_1 = U(\delta, 1+\delta)^d$ where $n = 50, 100, 200, 500$, $\alpha = 0.05, 0.1, \cdots, 0.45, 1.00$ and $d = 2, 3, 4, \cdots, 20$.

between the AUC of classifiers, we employ the 5x2 cross validation (CV) paired t-test (Dietterich, 1998) and the combined 5x2 CV F-test (Alpaydın, 1999). The 5x2 CV test has been devised by Dietterich (1998) and found to be the most powerful test among those with acceptable type-I error. However, the test statistics of 5x2 t-tests depend on which one of the ten folds is used. Hence, Alpaydın (1999) offered a combined 5x2 CV F-test which works as an omnibus test for all ten possible 5x2 t-tests (for each five repetitions there are two folds, hence ten folds in total). Basically, if a majority of ten 5x2 t-tests suggests that two classifiers are significantly different in terms of performance, the F-test also suggests a significant difference. Hence, an F-test with high p-value suggests that some of the ten t-tests fail to have low p-values.

We also provide the overlapping ratios and imbalance levels of these data sets. In a simulation study such as the one in Section 4.5, we have control on the overlapping region of two classes since we can choose the supports of the classes, hence their overlapping region is exactly known. However, in real data sets where the support of classes are neither defined nor available, we need methods to estimate the supports and hence estimate the overlapping ratios for the two classes. We employ the support vector data description (SVDD) method of Tax and Duin (2004) for this purpose. The method finds a description (or a region) of a data set, which covers a desired percentage of the points. SVDDs are also used in novelty or outlier detection. It has been inspired by the SVM classifiers and is based on defining a sphere around the data set. Similar to SVM, kernel functions can be employed to define more relaxed regions. SVDD is also a one-class learning method where the goal is to decide if a new point belongs to this particular class or not (Juszczak et al., 2002). By using SVDD approach, Xiong et al. (2010) found the SVDD regions of each class and its overlapping region. We also use SVDD to find the overlapping region E of each pair and report on the imbalance ratio with respect to E. The overlapping ratio is the percentage of points from both classes that reside in A. We use the Ddtools toolbox of MATLAB environment to produce the SVDDs of classes (Tax, 2014). Our choice of the kernel is the same as we have used with SVM classifiers, the radial basis (i.e., Gaussian) kernel; for consistency. However, the selection of σ in the kernel is crucial for the SVDD region.

In Table 4.2, we present the overlapping ratios and the imbalance in the overlapping areas of all data sets for $\sigma = 2, 3, \cdots, 10$. Although the value of σ produces different overlapping ratios, it is apparent that classes of data sets Ionosphere, Abalone19, Yeast4, Yeast6 and Yeast1289vs7 have more overlap than others, and these overlapping data sets have substantial local class imbalance in their respective overlapping regions. Other data sets have almost no overlapping nor imbalance in the overlapping regions even though their classes are globally imbalanced. One of these data sets is Yeast5 which has a imbalance ratio of $q = 32.70$ but has no imbalance in the small overlapping region.

In Table 4.3, we give the average AUC measures and their standard deviations of all CCCD-based and other classifiers according to the 5x2 CV scheme for the data sets. All other classifiers, have been two-way tested with 5x2 CV F-test against both

Table 4.2: Overlapping ratios and (local) imbalance ratios in the overlapping region of data sets. "IR" stands for the imbalance ratio in the overlapping region and "OR" stands for the overlapping ratio which is the percentage of points from both classes residing in the overlapping region. IR="NA" indicates that one of the classes has no members in the intersections of SVDD regions of classes.

Data	$q = n_1/n_0$	$n_0 + n_1$	d		$\sigma = 2$	$\sigma = 3$	$\sigma = 4$	$\sigma = 5$	$\sigma = 6$	$\sigma = 7$	$\sigma = 8$	$\sigma = 9$	$\sigma = 10$
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
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
Segment0	6.02	2308	20	OR	0%	0%	0%	0%	0%	0%	0%	0%	0%
				IR	NA	NA	NA	NA	NA	NA	NA	NA	NA
Page-Blocks0	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
Vowel0	9.98	988	14	OR	0%	0%	0%	0%	0%	0%	0%	0%	0%
				IR	NA	NA	NA	NA	NA	NA	NA	NA	NA
Shuttle0vs4	13.87	1829	10	OR	0%	0%	0%	0%	0%	0%	0%	0%	0%
				IR	NA	NA	NA	NA	NA	NA	NA	NA	NA
Yeast4	28.10	1484	9	OR	45%	27%	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	947	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
Yeast5	32.70	1484	9	OR	6%	3%	0%	0%	0%	0%	6%	7%	0.1%
				IR	NA	1.15	NA	NA	NA	NA	NA	NA	NA
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
Abalone19	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

RW-CCCD and P-CCCD classifiers. Their p-values are also provided in Table 4.3. For each of five repetitions, we divide the data into two folds. The AUC of fold 1 is given by using fold 1 as a training set and fold 2 as the test set. For fold 2, the process is similar. We repeated these experiments five times for all three classifiers. Looking at results from 11 data sets, RW-CCCD usually performs better than P-CCCD classifiers. On these data sets, RW-CCCD have significantly more AUC than the other classifiers. Thus, these results from real data sets seem to resonate with the results from our simulations and further support the robustness of CCCD classifiers to the class imbalance problem.

Table 4.3: Average of AUC values of ten folds, and standard deviations, of CCCD and other classifiers for real data sets. The p-values of 5x2 CV F-tests show the results of two-way tests comparing both CCCDs with other classifiers. Some of best performers are given in bold.

		Ionosphere	Sonar	Yeast6	Yeast5	Yeast4	Yeast1289vs7
RW-CCCD	AUC	0.917∓0.023	0.722∓0.050	**0.866∓0.051**	**0.898∓0.063**	**0.807∓0.048**	**0.643∓0.057**
P-CCCD	AUC	**0.934∓0.032**	**0.805∓0.045**	0.755∓0.053	0.793∓0.094	0.602∓0.051	0.556∓0.038
	AUC	0.803∓0.019	**0.804∓0.027**	0.786∓0.032	0.839∓0.063	0.619∓0.038	0.562∓0.04
k-NN	p-value (vs RW)	0.000	0.107	0.072	0.573	0.009	0.087
	p-value (vs P)	0.005	**0.473**	0.552	0.209	0.650	0.177
	AUC	**0.949∓0.010**	0.719∓0.057	0.710∓0.045	0.741∓0.069	0.527∓0.024	0.507∓0.014
SVM	p-value (vs RW)	0.088	0.763	0.043	0.096	0.002	0.047
	p-value (vs P)	**0.582**	0.091	0.086	0.467	0.129	0.084
	AUC	0.851∓0.024	0.712∓0.049	0.742∓0.048	0.803∓0.075	0.613∓0.057	0.570∓0.043
C4.5	p-value (vs RW)	0.008	0.747	0.015	0.429	0.084	0.270
	p-value (vs P)	0.039	0.358	0.634	0.734	0.819	0.508

		Vowel0	Shuttle0vs4	Abalone19	Segment0	Page-Blocks0
RW-CCCD	AUC	0.877∓0.046	**0.996∓0.003**	**0.603∓0.065**	0.895∓0.011	**0.875∓0.019**
P-CCCD	AUC	**0.958∓0.025**	0.988∓0.016	0.506∓0.019	**0.957∓0.010**	0.869∓0.009
	AUC	0.971∓0.031	0.996∓0.004	0.512∓0.015	**0.988∓0.004**	0.863∓0.009
k-NN	p-value (vs RW)	0.037	0.693	0.104	0.000	0.604
	p-value (vs P)	0.504	0.534	0.735	**0.026**	0.600
	AUC	0.956∓0.039	0.984∓0.012	0.500∓0.000	0.564∓0.006	0.901∓0.013
SVM	p-value (vs RW)	0.095	0.459	0.101	0.000	0.249
	p-value (vs P)	0.700	0.632	0.535	0.000	0.016
	AUC	0.948∓0.032	**0.999∓0.001**	0.503∓0.009	0.982∓0.004	**0.917∓0.012**
C4.5	p-value (vs RW)	0.090	**0.242**	0.088	0.000	**0.229**
	p-value (vs P)	0.137	0.533	0.535	0.100	0.020

4.7 Conclusions and Discussion

We assess the classification performance of various classifiers such as RW-CCCD, P-CCCD, k-NN, SVM and C4.5 classifiers and their variants when class imbalance occurs, and we illustrate the robustness of CCCD classifiers to the class imbalance in data sets. This imbalance often occurs in real life data sets where, in two-class settings, minority class (the class with fewer number of observations) is usually dwarfed by the majority class. Class imbalances hinder the performance of many classification algorithms. We studied the performance of CCCD classifiers under class imbalance problem by first simulating a two-class setting similar to the one used in DeVinney (2003). In this setting, the support of one class is entirely embedded in the support of the other. Drawing equal number of observations from both class supports results in an imbalance between two classes with respect to their overlapping region, called *local class imbalance*. This difference in the class sizes was also the case in the example of DeVinney (2003), and it is the reason that CCCD classifiers show better results than the k-NN classifier. We show that P-CCCD classifiers with lower θ values tend to perform better than the ones with higher θ values. This is merely a result of balls with $\theta = 0$ representing the local density of the target class points better. Similarly, the RW-CCCD classifiers with lower e values are better when the dimensionality is low and the class sizes are high. This might indicate that the denser the data set in $\mathbb{R}^d$, the less useful the scores T_x. However, fully utilizing the scores usually increases the classification performance.

Analysis of both simulated and real data sets indicate that both CCCD classifiers show robustness to the class imbalance problem. We demonstrated this by studying the effects of the class overlapping problem together with the class imbalance problem. In fact, there are studies in the literature focusing on the performance of classification methods when class overlapping and class imbalance problems occur simultaneously (Denil and Trappenberg, 2010; Prati et al., 2004). Overlapping of classes is an important factor in the classification of imbalanced data sets; that is, it drastically affects the classification performance of most algorithms. When classes are both imbalanced and overlapping, performance of k-NN, SVM and C4.5 classifiers deteriorate whereas CCCD classifiers are not affected as severely as these methods. We use two alternatives of C4.5 classifiers where we prune the decision tree in one and do not in the other. It is known for some time that pruning deteriorates the performance of tree classifiers under class imbalance. Moreover, SVM is robust to moderately imbalanced class sizes but demonstrates no robustness in highly imbalanced cases. However, whether the data set is highly or moderately imbalanced, CCCD classifiers seem to preserve their AUC compared to k-NN, SVM and C4.5 classifiers. Hence, our study suggests that CCCD classifiers are appealing alternatives when data have class imbalance. In addition, we mention the effect of the individual class sizes on the class imbalance problem (Japkowicz and Stephen, 2002). Whatever the ratio between class sizes is, if the minority class has a substantially high number of points, the effect of imbalances between classes tend to diminish.

We conduct simulation studies to determine how the classification performance jointly depends on both (global) class imbalance and class overlapping, parameterized as q and δ, respectively, and we apply all classifiers on several UCI and KEEL data sets. By using the SVDD method of Tax and Duin (2004), we estimated the overlapping ratios of all these data sets. We show that CCCD classifiers outperform or perform comparable to k-NN, SVM and C4.5 classifiers for some overlapping and imbalance ratios in both simulated and real data sets. In particular, CCCDs are better than SVM classifiers in highly imbalanced cases. The effect of high class imbalance on SVM classifier is also studied in Akbani et al. (2004) and Raskutti and Kowalczyk (2004). However, when no imbalance occurs between classes, CCCD classifiers usually show either comparable or slightly worse performance than the other classifiers.

We also investigate the performance of CCCD classifiers under different conditions. Specifically, in two different experiments, we simulate two classes where (i) classes are imbalanced but supports are not overlapping (well separated) and (ii) classes are balanced and supports are overlapping with increasing dimensionality. P-CCCD classifiers are better than RW-CCCD classifiers in experiment (i). Both CCCD classifiers mostly outperform k-NN and SVM classifiers when classes are imbalanced and not overlapping, however RW-CCCD classifiers outperform these classifiers only when dimensionality is sufficiently high. In experiment (ii), the classification performance of CCCD classifiers slightly degrade compared to k-NN and SVM classifiers, especially with increasing d. Among CCCD classifiers, RW-CCCDs appear to be better when classes are both overlapping and imbalanced, however our results suggest the use of P-CCCDs when classes are imbalanced and well separated (i.e., not overlapping).

In fact, class supports are often overlapping in real life data sets, hence RW-CCCD classifiers seem to be more appealing in practice.

In practice, classifiers based on CCCD classifiers resemble prototype selection methods. CCCDs balance the class sizes by defining balls that catch surrounding points of the same class, and discard these points from the training set. The resulting data set is composed of the centers of these balls and associated radii which are used in scaled dissimilarity measures. Although, CCCD classifiers remove substantial amount of observations from the majority class, they preserve (most of) the information with the radii. The bigger the radius, the more likely that the balls of CCCD classifiers contain more points. The radii could be considered as an indicator of the local density of the target class. The real advantage of CCCD classifiers are these prototype sets which are of (approximately) minimum cardinality, although training time and space of P-CCCDs and RW-CCCDs may be considerably high. CCCDs preserve important information regarding the data sets while substantially increasing the testing speed. In literature, many classifiers have been devised to preserve the information on the deleted majority class points, however they are all ensemble based classifiers which substantially increase both training and testing time complexities. In that regard, CCCDs offer a novel approach to this particular problem.

Eveland et al. (2005) modified RW-CCCD classifiers as to increase the speed of the face detection in which imbalances between classes occur naturally. They did only refer to the real life applications which consist of class imbalances. They did not, however, investigate the relationship between class imbalance and overlapping problems as thoroughly as our study does. On the other hand, establishing class covers with Euclidean balls raise the possibility of using different regions (the regions are Euclidean hyperballs around target class points in CCCD) to balance the data and, thus, construct non-parametric classifiers with more classification performance. Along this line, CCCDs can be generalized using PCDs (Jaromczyk and Toussaint, 1992). In the following chapter, we will show how PCDs can be used to derive new graph-based classifiers which are also robust to the class imbalance problem.

Chapter 5

PROTOTYPE-BASED CLASSIFICATION WITH PROXIMITY CATCH DIGRAPHS

5.1 Introduction

Classification methods based on set covering algorithms received considerable attention because of their use in prototype selection (Angiulli, 2012; Bien and Tibshirani, 2011; Cannon and Cowen, 2004). Prototypes are selected members of a data set so as to attain various tasks including reducing, condensing or summarizing a data set. Many learning methods aim to carry out more than one of these tasks, thereby building efficient learning algorithms (Bien and Tibshirani, 2011; Pękalska et al., 2006). A desirable prototype set reduces the data set in order to decrease running time, condenses the data set to preserve information, and summarizes the data set for better exploration and understanding. The methods we discuss are considered as decision boundary generators where decisions are made based on class conditional regions, or *class covers*, that are composed of a collection of convex sets, each associated with a prototype (Toussaint, 2002). The union of such convex sets constitute a region for the class of interest, estimating the support of this class (Schölkopf et al., 2001). Support estimates have uses in both supervised and unsupervised learning schemes offering solutions to many problems of machine learning literature (Marchette, 2004). We propose supervised learning methods, or classifiers, based on these estimates of the supports constructed with PCDs.

In this chapter, we employ PCDs in statistical classification and investigate their performance. The PCDs of concern are classifiers based on a two families of proximity maps called PE and CS proximity maps. The corresponding PCDs are called PE-

PCDs and CS-PCDs, respectively, and are defined for target class (i.e. the class of interest) points inside the convex hull of non-target points (Ceyhan, 2005). However, this construction ignores the target class points outside the convex hull of the non-target class. We mitigate this shortcoming by partitioning the region outside of the convex hull into unbounded regions, called outer simplices, which may be viewed as extensions of outer intervals in $\mathbb{R}$ (e.g. intervals with infinite endpoints) to higher dimensions. We attain proximity regions in these outer simplices by extending PE and CS proximity maps to outer simplices. We establish two types of classifiers based on PCDs, namely *hybrid* and *cover* classifiers. The first type incorporates the PCD covers of only points in the convex hull and use other classifiers for points outside the convex hull of the non-target class, hence we have some kind of a hybrid classifier; the second type is further based on two class cover models where the first is a hybrid of PE-PCDs (or CS-PCDs) and CCCDs (composite covers) whereas the second is purely based on either PE-PCDs or CS-PCDs (standard covers).

One common property of most class covering (or set covering) methods is that none of the algorithms find the exact minimum number of covering sets in polynomial time, and solutions are mostly provided by approximation algorithms (Vazirani, 2001). However, for PE-PCDs, the exact minimum number of covering sets (equivalent to prototype sets) can be found much faster; that is, the exact minimum solution is found in a running time polynomial in size of the data set but exponential in dimensionality. PE-PCDs have computationally tractable (exact) MDSs in $\mathbb{R}^d$ (Ceyhan, 2010). On the other hand, Ceyhan (2005) listed some properties CCCDs in $\mathbb{R}$ that can be used to define new PCDs in $\mathbb{R}^d$ for $d > 2$. CS proximity maps satisfy all these properties in $\mathbb{R}^d$ that may potentially provide better estimations of the class supports than PE proximity maps. Although the complexity of class covers based on this family of proximity maps exponentially increases with dimensionality, we apply dimension reduction methods (e.g. principal components analysis) to substantially reduce the number of features and to reduce the dimensionality. Hence, based on the transformed data sets in the reduced dimensions, the PCD (PE-PCD and CS-PCD) based hybrid and, in particular, cover classifiers become more appealing in terms of both prototype selection and classification performance (in the reduced dimension). We use simulated and real data sets to show that these two types of classifiers based on PCDs have either comparable or slightly better classification performance than other classifiers when the data sets exhibit the class imbalance problem.

5.2 PCD covers

We establish class covers with the PE proximity map $\mathcal{N}_{PE}(\cdot, r)$, CS proximity map $\mathcal{N}_{CS}(\cdot, \tau)$, and spherical proximity map $\mathcal{N}_S(\cdot)$. We define two types of class covers: one type is called *composite covers* where we cover the points in $\mathcal{X}_j \cap C_H(\mathcal{X}_{1-j})$ with either PE or CS proximity maps and the points in $\mathcal{X}_j \setminus C_H(\mathcal{X}_{1-j})$ with spherical proximity maps, and the other is called *standard cover* incorporating either PE or CS proximity maps for all points in $\mathcal{X}_j$. We use these two types of covers to establish a specific type of classifier that is more appealing in the sense of prototype selection.

Our composite covers are mixtures of simplicial and spherical proximity regions. Specifically, given a set of simplices and a set of spheres, the composite cover is the union of both these sets which constitute proximity regions of two separate PCD families, hence the name *composite cover*. The Q_j is partitioned into two: the cover $Q_j^{(1)}$ of points inside the convex hull of non-target class points, i.e., $\mathcal{X}_j \cap C_H(\mathcal{X}_{1-j})$, and the cover $Q_j^{(2)}$ of points outside, i.e., $\mathcal{X}_j \setminus C_H(\mathcal{X}_{1-j})$. Let $Q_j^{(1)} := \cup_{x \in \mathcal{X}_j \cap C_H(\mathcal{X}_{1-j})} \mathcal{N}_I(x)$ and $Q_j^{(2)} := \cup_{x \in \mathcal{X}_j \setminus C_H(\mathcal{X}_{1-j})} \mathcal{N}_O(x)$ such that $Q_j := Q_j^{(1)} \cup Q_j^{(2)}$. Here, $\mathcal{N}_I(\cdot)$ and $\mathcal{N}_O(\cdot)$ are proximity maps associated with sets $\mathcal{X}_j \cap C_H(\mathcal{X}_{1-j})$ and $\mathcal{X}_j \setminus C_H(\mathcal{X}_{1-j})$, respectively. Hence, in composite covers, target class points inside $C_H(\mathcal{X}_{1-j})$ are covered with PE proximity map $\mathcal{N}_I(\cdot) = \mathcal{N}_{PE}(\cdot, r)$ or CS proximity map $\mathcal{N}_I(\cdot) = \mathcal{N}_{CS}(\cdot, \tau)$, and the remaining points are covered with spherical proximity map $\mathcal{N}_O(\cdot) = \mathcal{N}_S(\cdot, \theta)$. Given the covers $Q_j^{(1)}$ and $Q_j^{(2)}$, let $C_j^{(1)}$ and $C_j^{(2)}$ be the class covers with lower complexity associated with the dominating sets $S_j^{(1)}$ and $S_j^{(2)}$. Hence the composite cover is given by

$$C_j := C_j^{(1)} \cup C_j^{(2)} = \left\{ \bigcup_{s \in S_j^{(1)}} N_I(s) \right\} \bigcup \left\{ \bigcup_{s \in S_j^{(2)}} N_O(s) \right\}.$$

An illustration of the class covers C_0 and C_1 with $\mathcal{N}_I(\cdot) = \mathcal{N}_{PE}(\cdot, r = 2)$ and $\mathcal{N}_O(\cdot) = \mathcal{N}_S(\cdot, \theta = 1)$, and $\mathcal{N}_I(\cdot) = \mathcal{N}_{CS}(\cdot, \tau = 1)$ and $\mathcal{N}_O(\cdot) = \mathcal{N}_S(\cdot, \theta = 1)$ is given in Figure 5.1(b) and (c).

By definition, the spherical proximity map $\mathcal{N}_S(\cdot, \theta)$ yields class covers for all points in $\mathcal{X}_j$. Figure 5.1(a) illustrates the class covers of the map $\mathcal{N}_S(\cdot, \theta = 1)$. We call such covers, that only constitute a single type of proximity map, as *standard covers*. Hence the standard cover of the PCD D_j is a union of d-simplices and d-polytopes:

$$C_j := \bigcup_{s \in S_j} \mathcal{N}_{PE}(s, r).$$

Here, $\mathcal{N}_I(\cdot) = \mathcal{N}_O(\cdot) = \mathcal{N}_{PE}(\cdot, r)$ for standard covers with PE proximity maps and $\mathcal{N}_I(\cdot) = \mathcal{N}_O(\cdot) = \mathcal{N}_{CS}(\cdot, \tau)$. An illustration is given in Figure 5.1(d) and (e).

5.3 Classification with PCDs

The elements of S_j are prototypes, for the problem of modelling the class conditional discriminant regions via a collection of proximity regions (balls, simplices, polytopes, etc.). The sizes of these regions represent an estimate of the domain of influence, which is the region in which a given prototype should influence the class labelling. Our semi-parametric classifiers depends on the class covers given by these proximity regions. We define various classifiers based on the class covers (composite or standard) and some other classification methods. We approach classification of points in $\mathbb{R}^d$ in two ways:

Hybrid classifiers: Given the class covers $C_0^{(1)}$ and $C_1^{(1)}$ associated with classes $\mathcal{X}_0$ and $\mathcal{X}_1$, we classify a given point $z \in \mathbb{R}^d$ with g_P if $z \in C_0^{(1)} \cup C_1^{(1)}$, and with g_A otherwise. Here, g_P is the pre-classifier and g_A is an alternative classifier.

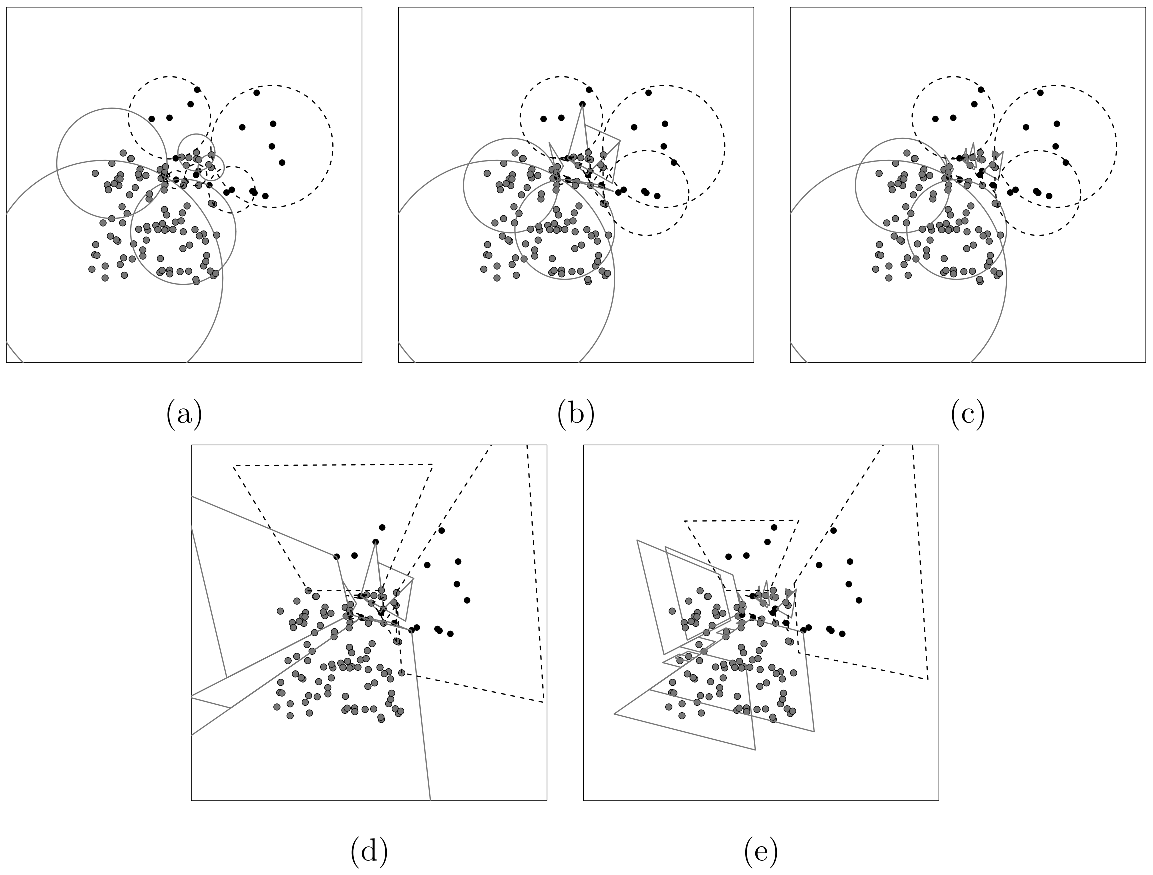


Figure 5.1: Class covers of a data set with two-class setting in $\mathbb{R}^2$ where grey and black points represent points of two separate classes. The training data set is composed of two classes $\mathcal{X}_0$ and $\mathcal{X}_1$ wherein 100 and 20 samples are drawn from multivariate uniform distributions $U([0,1]^2)$ and $U([0.5,1.5]^2)$, respectively. Cover of one class is given by solid circle and solid line segments, and the cover of the other is given by dashed circle and dashed line segments. (a) Standard class covers with $\mathcal{N}_I(\cdot) = \mathcal{N}_O(\cdot) = \mathcal{N}_S(\cdot, \theta = 1)$ (b) Composite class cover with $\mathcal{N}_I(\cdot) = \mathcal{N}_{PE}(\cdot, r = 2)$ and $\mathcal{N}_O(\cdot) = \mathcal{N}_S(\cdot, \theta = 1)$ (c) Composite class cover with $\mathcal{N}_I(\cdot) = \mathcal{N}_{CS}(\cdot, \tau = 1)$ and $\mathcal{N}_O(\cdot) = \mathcal{N}_S(\cdot, \theta = 1)$ (d) Standard class covers with $\mathcal{N}_I(\cdot) = \mathcal{N}_O(\cdot) = \mathcal{N}_{PE}(\cdot, r = 2)$ (e) Standard class covers with $\mathcal{N}_I(\cdot) = \mathcal{N}_O(\cdot) = \mathcal{N}_{CS}(\cdot, \tau = 1)$.

Cover classifiers: These classifiers are constructed by class covers only; that is, a given point $z \in \mathbb{R}^d$ is classified as $g_C(z) = j$ if $z \in C_j \setminus C_{1-j}$ or if $\rho(z, C_j) < \rho(z, C_{1-j})$, hence class of the point z is estimated as j if z is only in cover C_j, or closer to C_j than C_{1-j}. Here, $\rho(z, C_j)$ is a dissimilarity measure between point z and the cover C_j. Cover classifiers depend on the types of covers which are either *composite* or *standard* covers.

We incorporate PE-PCDs and CS-PCDs for establishing both of these types of classifiers. Hence, we will refer to them as hybrid PCD (PE-PCD or CS-PCD) and cover PCD classifiers. Since the PE and CS proximity maps were originally defined for points $\mathcal{X}_j \cap C_H(\mathcal{X}_{1-j})$, we develop hybrid PCD classifiers to account for points outside of the convex hull of the non-target class in a convenient fashion. However, as we shall see later, cover PCD classifiers, especially the ones associated PE-PCDs, have much more appealing properties than hybrid PE-PCD classifiers in terms of both efficiency and classification performance. Nevertheless, we consider and compare both types of classifiers, but first we define the PCD pre-classifier.

5.3.1 PCD Pre-classifier

Let $\rho(z, C)$ be the dissimilarity measure between z and the class cover C. The PE-PCD pre-classifier is given by

$$g_P(z) := \begin{cases} j & \text{if } z \in C_j^{(1)} \setminus C_{1-j}^{(1)} \text{ for } j = 0, 1 \\ I(\rho(z, C_1^{(1)}) < \rho(z, C_0^{(1)})) & \text{if } z \in C_0^{(1)} \cap C_1^{(1)} \\ -1 & \text{otherwise.} \end{cases} \tag{5.1}$$

Here, $I(\cdot)$ is the indicator functional and $g_P(z) = -1$ denotes a "no decision" case. Given that class covers $C_0^{(1)}$ and $C_1^{(1)}$ are the unions of simplical (PE or CS) proximity regions of points in dominating sets $S_0^{(1)}$ and $S_1^{(1)}$, the closest cover is found by, first, checking the proximity region of a cover closest to the point z:

$$\rho(z, C_j^{(1)}) = \min_{s \in S_j^{(1)}} \rho(z, \mathcal{N}(s))$$

which is expressed based on a dissimilarity measure between a point z and the region $\mathcal{N}(s)$. For such measures, we employ convex distance functions. Let H be a convex set in $\mathbb{R}^d$ with center $x \in H$. The point x may be viewed as the center of the set H. Thus, let the dissimilarity between z and H be defined by

$$\rho(z, H) := \frac{d(z, x)}{d(t, x)},$$

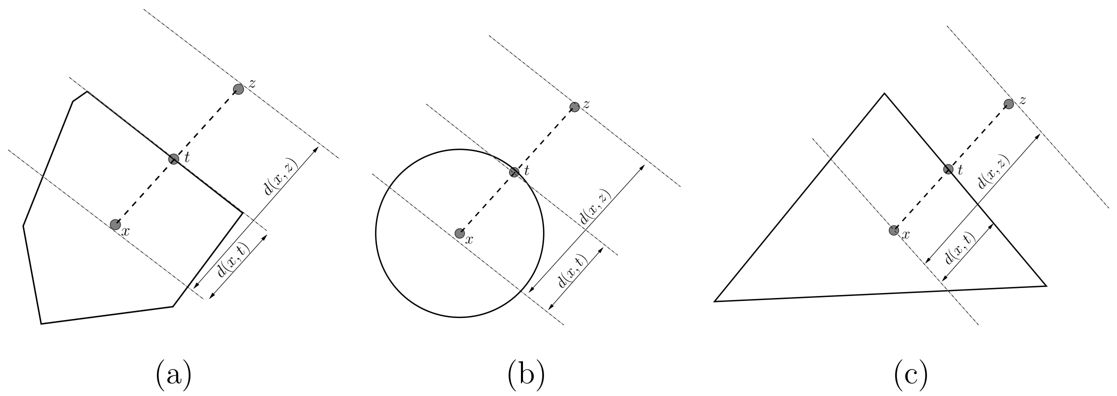


Figure 5.2: Illustration of a convex distance between a a point z and an arbitrary (a) convex set H, (b) ball and (c) 2-simplex in $\mathbb{R}^2$.

where $d(\cdot,\cdot)$ is the Euclidean distance and t is a point on the line $L(x,z) := \{x + \alpha(z-x) : \alpha \in [0,\infty)\}$ such that $t \in \partial(H)$, the boundary of the H. An illustration is given in Figure 5.2 for several convex sets, including balls and simplices in $\mathbb{R}^2$.

For spherical proximity map $\mathcal{N}_S(\cdot,\theta)$, the dissimilarity function is defined by the radius of that ball which is a spherical proximity region: $d(x,t) = \varepsilon_\theta(x)$ (Priebe et al., 2003a). However, for d-simplices, we characterize the dissimilarity measure in terms of barycentric coordinates of z with respect to d-simplex $\mathfrak{S}(x) = \mathcal{N}_{PE}(\cdot,r)$ for PE proximity maps and $\mathfrak{S}(x) = \mathcal{N}_{CS}(\cdot,\tau)$ for CS proximity maps.

Proposition 5.3.1.1. *Let $\{t_1,t_2,\ldots,t_{d+1}\} \subset \mathbb{R}^d$ be a set of non-collinear points that are the vertices of simplex $\mathfrak{S}(x) = \mathcal{N}(x)$ with the median $M_C(x) \in \mathfrak{S}(x)^o$. Then, for $z \in \mathbb{R}^d$ and $t \in \partial(\mathfrak{S}(x))$,*

$$\rho(z,\mathfrak{S}(x)) = \frac{d(M_C(x),z)}{d(M_C(x),t)} = 1-(d+1)w^{(k)}_{\mathfrak{S}(x)}(z),$$

where $w^{(k)}_{\mathfrak{S}(x)}(z)$ being the k'th barycentric coordinate of z with respect to $\mathfrak{S}(x)$. Moreover, $\rho(z,\mathfrak{S}(x)) < 1$ if $z \in \mathfrak{S}(x)^o$ and $\rho(z,\mathfrak{S}(x)) \geq 1$ if $z \notin \mathfrak{S}(x)^o$.

Proof: Let the line segment $L(M_C(x),z)$ and $\partial(\mathfrak{S}(x))$ cross at the point $t \in f_k$ for f_k being the face of $\mathfrak{S}(x)$ opposite to t_k. Thus, for $\alpha_i \in (0,1)$ and $\beta \in (0,1)$,

$$z = (1-\beta)M_C(x) + \beta t = (1-\beta)M_C(x) + \beta\left(\sum_{i=1;i\neq k}^{d+1} \alpha_i t_i\right).$$

Here, note that $\beta = d(M_C(x), z)/d(M_C(x), t) = \rho(z, \mathfrak{S}(x))$. Also, since $M_C(x)$ is the median,

$$z = (1-\beta)\frac{\sum_{i=1}^{d+1} t_i}{d+1} + \beta\left(\sum_{i=1;i\neq k}^{d+1} \alpha_i t_i\right) = \frac{1-\beta}{d+1}t_k + \sum_{i=1;i\neq k}^{d+1}\left(\frac{1-\beta}{d+1} + \beta\alpha_i\right)t_i.$$

Hence $(1-\beta)/(d+1) = w^{(k)}_{\mathfrak{S}(x)}(z)$ which implies $\beta = 1-(d+1)w^{(k)}_{\mathfrak{S}(x)}(z)$. Therefore, $z \in \mathfrak{S}(x)^o$ if and only if $\beta = 1-(d+1)w^{(k)}_{\mathfrak{S}(x)}(z) < 1$. ■

For (convex) proximity regions $\mathcal{N}_{PE}(x, r)$ and $\mathcal{N}_{CS}(x, \tau)$, the dissimilarity measure $\rho(z, \mathfrak{S}(x))$ indicates whether or not the point z is in THE proximity region, since $\rho(z, \mathfrak{S}(x)) < 1$ if $z \in \mathcal{N}_{PE}(x, r)$ and ≥ 1 otherwise. Hence, the PE-PCD pre-classifier g_P may simply be defined by

$$g_P(z) := \begin{cases} I(\rho(z, C_1^{(1)}) < \rho(z, C_0^{(1)})) & \text{if } z \in C_0^{(1)} \cup C_1^{(1)} \\ -1 & \text{otherwise} \end{cases} \tag{5.2}$$

since $z \in C_0^{(1)} \setminus C_1^{(1)}$ if and only if $\rho(z, C_0^{(1)}) < 1$. Let $\rho(z, x) := \rho(z, \mathfrak{S}(x))$ be the dissimilarity between x and z, then the dissimilarity measure $\rho(\cdot,\cdot)$ violates the symmetry axiom of the metric since $\rho(x, z) \neq \rho(z, x)$ whenever $d(x, t(x)) \neq d(z, t(z))$ where proximity regions $\mathcal{N}(x)$ and $\mathcal{N}(z)$ intersect with the lines $L(M_C(x), z)$ and $L(M_C(z), x)$ at points $t(x)$ and $t(z)$, respectively.

5.3.2 Hybrid PE-PCD Classifiers

Constructing hybrid classifiers has many purposes. Some classifiers are designed to solve harder classification problems by gathering many weak learning methods (often known as *ensemble* classifiers) while some others have advantages only when combined with another single classifier (Woźniak et al., 2014). Our hybrid classifiers are of the latter type. The PCD pre-classifier g_P is able to classify points in the overlapping region of the class supports, i.e. $s(F_0) \cap s(F_1)$, however classifying the remaining points in $\mathbb{R}^d$ requires incorporating an alternative classifier, often one that works for all points $\mathbb{R}^d$. We use the PCD pre-classifier $g_P(\cdot)$ to classify all points of the test data, and if no decision are made for some of these points, we classify them with the alternative classifier g_A. Hence, let g_H be the hybrid PCD classifier such that

$$g_H(z) := \begin{cases} g_P(z) & \text{if } z \in C_0^{(1)} \cup C_1^{(1)} \\ g_A(z) & \text{otherwise.} \end{cases} \tag{5.3}$$

For "no decision" cases where $g_P(z) = -1$, we rely on the alternative classifier g_A; we will use the k-nearest neighbor, SVM and CCCD classifiers as alternative classifiers. The parameters are k, the number of closest neighbors to make a majority vote in the k-NN classifier; γ, the scaling parameter of the radial basis function (RBF) kernel of the SVM classifier; and θ, the parameter of the CCCD classifier that regulates the size of each ball as described in Section 3.2.

Hybrid PCD classifiers depend on both the PCD pre-classifier g_P and the alternative classifier g_C. Therefore, we use some of the well known classification methods in the literature to incorporate them as alternative classifiers. All these classifiers are well defined for all points in $\mathbb{R}^d$, so we use them when the PCD pre-classifier fails to make a decision, i.e. $g_P(z) = -1$. In addition to considering these classifiers as alternative classifiers, we apply them to the entire training data set in our simulated and real data studies to compare them with our hybrid classifiers as well.

5.3.3 Composite and Standard Cover PE-PCD Classifers

We propose PCD classifiers g_C based on composite and standard covers. The classifier g_C is defined as

$$g_C(z) := I(\rho(z, C_1) < \rho(z, C_0)). \tag{5.4}$$

The cover is based on either composite covers or standard covers wherein both $\mathcal{X}_j \subset C_j$, hence a decision can be made without an alternative classifier. Note that composite cover PE-PCD classifiers are, in fact, different types of hybrid classifiers where the classifiers are only modelled by class covers but with multiple types of PCDs. Compared to hybrid PCD classifiers, cover PCD classifiers have many appealing properties. Since a reduction is done over all target class points $\mathcal{X}_j$, depending on the percentage of reduction, classifying a new point $z \in \mathbb{R}^d$ is computationally faster and more efficient, whereas an alternative classifier might not provide such a reduction. Note that, given the multi-class covers, the two-class cover PCD classifier g_C can be modified for the multi-class case as

$$g(z) = \underset{j \in J}{\operatorname{argmin}} \rho(z, C_j). \tag{5.5}$$

5.3.4 Consistency Analysis

In this section, we prove some results on the consistency of both hybrid PCD classifiers and cover PCD classifiers when two class conditional distributions are strictly δ-seperable. For $\delta \in [0, \infty)$, the regions $A, B \subset \mathbb{R}^d$ are *δ-separable* if and only if

$$\inf_{x \in A, y \in B} d(x, y) \geq \delta.$$

Moreover, let δ-separable regions A and B be the supports of continuous distributions F_A and F_B, respectively. Hence, F_A and F_B are called *δ-separable distributions*, and if $\delta > 0$, *strictly* δ-separable (Devroye et al., 1996).

We first show the consistency of cover PCD classifiers, and then, we show that the hybrid PCDs classifiers are also consistent. Cover classifiers are characterized by the PCDs associated with proximity regions $\mathcal{N}(x)$ for $x \in \mathbb{R}^d$, and thus, the consistency of such PCD classifiers depend on the map $\mathcal{N}(\cdot)$. We require the following properties for a proximity map $\mathcal{N}(\cdot)$ to satisfy:

P1 For all $x \in \mathbb{R}^d$, the proximity region $\mathcal{N}(x)$ is an open set, and x is in the interior of $\mathcal{N}(x)$.

P2 Given data sets from two classes $\mathcal{X}_0$ and $\mathcal{X}_1$ with distributions F_0 and F_1, and supports $s(F_0)$ and $s(F_1)$, and given that $x \in s(F_j)$ for $j = 0, 1$, the proximity map $\mathcal{N}(\cdot)$ associated with the target class $\mathcal{X}_j$ is a function on the non-target class points such that $\mathcal{N}(x) \cap \mathcal{X}_{1-j} = \emptyset$.

Note that $\mathcal{N}_S(\cdot, \theta)$ for $\theta \in (0, 1]$, $\mathcal{N}_{PE}(\cdot, r)$ for $r \in (1, \infty)$ and $\mathcal{N}_{CS}(\cdot, \tau)$ for $\tau \in (0, \infty)$ satisfy **P1** and **P2**. These will be useful in showing that classifiers based on our class covers attain Bayes-optimal classification performance for δ-separable classes. Thus, first, we have to show that the support of a class is almost surely a subset of the class cover for sufficiently large data sets. Note that all points of the target class reside inside the class cover C_j, i.e. $\mathcal{X}_j \subset C_j$. Hence, we have the following proposition.

Proposition 5.3.4.1. *Let $\mathcal{Z}_n = \{Z_1, Z_2, \dots, Z_n\}$ be a set of i.i.d. random variables drawn from a continuous distribution F whose support is $s(F) \subseteq \mathbb{R}^d$. Let the proximity map $\mathcal{N}(\cdot)$ satisfy **P1**, let the corresponding class cover of $\mathcal{Z}_n$ be denoted as $C(\mathcal{Z}_n)$ such that $\mathcal{Z}_n \subset C(\mathcal{Z}_n)$, and let $C^* := \liminf_{n\to\infty} C(\mathcal{Z}_n)$. Hence, we have $s(F) \subset C^*$ w.p. 1 in the sense that $\lambda(s(F) \setminus C^*) \to 0$ almost surely where $\lambda(\cdot)$ is the Lebesgue measure functional.*

Proof: Suppose, for a contradiction, $s(F) \not\subset C^*$ w.p. 1. Hence, $s(F) \setminus C^* \neq \emptyset$ w.p. 1 in such a way that $\lambda(s(F) \setminus C^*) > 0$ w.p. 1 since $\lambda(\mathcal{N}(Z)) > 0$ for all $Z \in s(F)$ by **P1**. Hence, $\mathcal{Z}_n \cap (s(F) \setminus C^*) \neq \emptyset$ w.p. 1 as $n \to \infty$, but then some $Z \in \mathcal{Z}_n \cap (s(F) \setminus C^*)$ will not be in C^*, which contradicts the fact that C^* covers $\mathcal{Z}_n$ including Z. ■

Proposition 5.3.4.1 shows that a class cover almost surely covers the support of its associated class. However, to show consistency of classifiers based on PCD class covers, we have to investigate the class covers under the assumption of separability of class supports.

Let $\mathcal{X}_0$ and $\mathcal{X}_1$ be two classes of a data set with strictly δ-separable distributions, the property **P2** of the map $\mathcal{N}(\cdot)$ establishes *pure* class covers that include none of the points of the non-target class, i.e. $C_j \cap \mathcal{X}_{1-j} = \emptyset$. In this case, we have the following proposition showing that the intersection of the cover of the target class and the support of the non-target class is almost surely empty as $n_{1-j} \to \infty$.

Proposition 5.3.4.2. *Let $\mathcal{X}_0 = \{X_1, X_2, \dots, X_{n_0}\}$ and $\mathcal{X}_1 = \{Y_1, Y_2, \dots, Y_{n_1}\}$ be two sets of i.i.d. random variables with strictly δ-separable continuous distributions F_0 and F_1. For $j = 0, 1$, let the proximity map $\mathcal{N}(\cdot)$ satisfy **P1** and **P2** such that the map $\mathcal{N}(\cdot)$ of the target class is a function on the non-target class $\mathcal{X}_{1-j}$. Then, for*

$j = 0, 1$, *we have* $C(\mathcal{X}_j) \cap s(F_{1-j}) = \emptyset$ *almost surely as* $n_{1-j} \to \infty$ *in the sense that* $\lambda(C(\mathcal{X}_j) \setminus s(F_{1-j})) \to 0$ *as* $n_{1-j} \to \infty$.

Proof: For $j = 0, 1$, note $C(\mathcal{X}_j) = \cup_{X \in S_j} \mathcal{N}(X)$ for $S_j \subset \mathcal{X}_j$ being the minimum prototype set of $\mathcal{X}_j$. We prove the proposition for $j = 0$ (as the proof of case $j = 1$ follow by symmetry). Hence, it is sufficient to show that (given $\mathcal{N}(\cdot)$ is a function on $\mathcal{X}_1$) $\mathcal{N}(x) \cap s(F_1) = \emptyset$ w.p. 1 as $n_1 = |\mathcal{X}_1| \to \infty$ for all $x \in s(F_0)$. Suppose for a contradiction, $\lambda(C(\mathcal{X}_0) \setminus s(F_1)) > 0$ w.p. 1 as $n_1 \to \infty$. Then, there exists $x \in s(F_0)$ such that $\mathcal{N}(x) \cap s(F_1) \neq \emptyset$ almost surely as $n_1 \to \infty$. Then, the region $\mathcal{N}(x) \cap s(F_1)$ has positive measure. Therefore, some $Y \in \mathcal{X}_1$ will fall in to this region w.p. 1 as $n_1 \to \infty$. This contradicts **P2** since $Y \in \mathcal{N}(x) \cap s(F_1)$ implies $\mathcal{N}(x) \cap \mathcal{X}_1 \neq \emptyset$. ■

Now, we would like to show that cover PCD classifiers are consistent when class supports are strictly δ-separable; that is, the error rate of the cover classifier $L(g_C)$ converges to the Bayes optimal error rate L^*, which is 0 for classes with δ-separable supports, as $n_0, n_1 \to \infty$ (Devroye et al., 1996). Then, we have the following theorem.

Theorem 5.3.4.1. *Suppose that the samples of the data set* $\mathcal{X}_0 \cup \mathcal{X}_1$ *are i.i.d. with distribution* $F = \pi_0 F_0 + (1 - \pi_0) F_1$ *for* $\pi_0 \in [0, 1]$, *and let class conditional distributions* F_0 *and* F_1 *are continuous with supports* $s(F_0)$ *and* $s(F_1)$, *being finite dimensional and strictly* δ*-separable. Then the cover classifier* g_C *is consistent; that is,* $L(g_c) \to L^* = 0$ *as* n_0, $n_1 \to \infty$.

Proof: Let Z_j be a random variable with distribution F_j for $j = 0, 1$. Then by Propositions 5.3.4.1 and 5.3.4.2, we have $P(Z_j \in C(\mathcal{X}_j)) \to 1$ as $n_j \to \infty$ and $P(Z_j \notin C(\mathcal{X}_{1-j})) \to 1$ as $n_{1-j} \to \infty$. Hence,

$$P(Z_j \notin C(\mathcal{X}_j) \text{ and } Z_j \in C(\mathcal{X}_{1-j})) \to 0$$

as $n_0, n_1 \to \infty$. Then, for $C_j = C(\mathcal{X}_j)$,

$$\begin{aligned} L(g_C) &= P(g_C(Z_0) \neq 0)\pi_0 + P(g_C(Z_1) \neq 1)\pi_1 \\ &= P(Z_0 \notin C_0 \text{ and } Z_0 \in C_1)\pi_0 + P(Z_1 \notin C_1 \text{ and } Z_1 \in C_0)\pi_1. \end{aligned}$$

Hence, $L(g_C) \to 0$ as $n_0, n_1 \to \infty$. ■

As a corollary to Theorem 5.3.4.1, we have that classifier g_C of standard and composite covers with maps $N_S(\cdot, \theta)$ and $N_{PE}(\cdot, r)$ for $r > 1$ are consistent. A special case occurs when $r = 1$; that is, observe that $x \in \partial(\mathcal{N}(x))$, and hence $\mathcal{N}(\cdot)$ does not satisfy **P1**.

We showed that a cover PCD classifier is consistent provided that, as $n_0, n_1 \to \infty$, support of the target class is a subset of the class cover, and the PCD cover excludes all points of the non-target class almost surely. However, to show that the hybrid PCD classifiers are consistent, we need alternative classifiers which are consistent as well.

Theorem 5.3.4.2. *Suppose that the samples of data set $\mathcal{X}_0 \cup \mathcal{X}_1$ are i.i.d. with distribution $F = \pi_0 F_0 + (1 - \pi_0) F_1$ for $\pi_0 \in [0, 1]$, and let class conditional distributions F_0 and F_1 are continuous with supports $s(F_0)$ and $s(F_1)$, being finite dimensional and strictly δ-separable. Then the hybrid classifier g_H is consistent provided that alternative classifier g_A is also consistent.*

Proof: Note that $C_j = C_j^{(1)} \cup C_j^{(2)}$ and $C_j^{(1)} \subset C_H(\mathcal{X}_{1-j})$. For $j = 0, 1$, let $Z_j \sim F_j$. Also, let Υ_j be the event that $Z_j \in C_0^{(1)} \cup C_1^{(1)}$ and let $\upsilon_j := P(\Upsilon_j)$. Note that

$$L(g_H) = P(g_H(Z_0) \neq 0)\pi_0 + P(g_H(Z_1) \neq 1)\pi_1.$$

Hence, for $j = 0, 1$;

$$\begin{aligned} P(g_H(Z_j) \neq j) &= P(g_H(Z_j) \neq j | \Upsilon_j)\upsilon_j + P(g_H(Z_j) \neq j | \Upsilon_j^c)(1 - \upsilon_j) \\ &= P(g_P(Z_j) \neq j | \Upsilon_j)\upsilon_j + P(g_A(Z_j) \neq j | \Upsilon_j^c)(1 - \upsilon_j). \end{aligned}$$

As $n_0, n_1 \to \infty$, $P(g_P(Z_j) \neq j | \Upsilon_j) \to 0$ by Theorem 5.3.4.1, and $P(g_A(Z_j) \neq j) \to 0$ since the classifier g_A is consistent. Then the result follows. ■

5.4 Monte Carlo Simulations and Experiments

In this section, we assess the classification performance of hybrid and cover PCD classifiers. We perform simulation studies wherein observations of two classes are drawn from separate distributions where $\mathcal{X}_0$ is a random sample from a multivariate uniform distribution $U([0,1]^d)$ and $\mathcal{X}_1$ is from $U([\nu, 1+\nu]^d)$ for $d = 2,3,5$ with the overlapping parameter $\nu \in [0,1]$. Here, ν determines the level of overlap between the two class supports. We regulate ν in such a way that the overlapping ratio ζ is fixed for all dimensions, i.e. $\zeta = \operatorname{Vol}(s(F_0) \cap s(F_1))/\operatorname{Vol}(s(F_0) \cup s(F_1))$. When $\zeta = 0$, the supports are well separated, and when $\zeta = 1$, the supports are identical: i.e. $s(F_0) = s(F_1)$. Hence, the closer the ζ to 1, the more the supports overlap. Observe that $\nu \in [0,1]$ can be expressed in terms of the overlapping ratio ζ and dimensionality d:

$$\zeta = \frac{\operatorname{Vol}(s(F_0) \cap s(F_1))}{\operatorname{Vol}(s(F_0) \cup s(F_1))} = \frac{(1-\nu)^d}{2-(1-\nu)^d} \quad \Longleftrightarrow \quad \nu = 1 - \left(\frac{2\zeta}{1+\zeta}\right)^{1/d}. \tag{5.6}$$

In this simulation study, we train the classifiers with $n_0 = 400$ and $n_1 = qn_0$ with the imbalance level $q = n_1/n_0 = \{0.1, 0.5, 1.0\}$ and overlapping ratio $\zeta = 0.5$. For values of q closer to zero, classes of the data set are more imbalanced. On each replication, we form a test data with 100 random samples drawn from each of F_0 and F_1, resulting a test data set of size 200. This setting is similar to a setting used in Chapter 4, where we showed that CCCD classifiers are robust to imbalance in data sets. We intend to show that the same robustness extends to PE-PCD and CS-PCD classifiers. We use the area under curve (AUC) measure to evaluate the performance of the classifiers on the imbalanced data sets (López et al., 2013). AUC measure is often used on imbalanced real data classes. This measure has been shown to be better than the correct classification rate in general (Huang and Ling, 2005). Using all classifiers, at each replication, we record the AUC measures for the test data, and also, we record the correct classification rates (CCRs) of each class of the test data separately. We perform these replications until the standard errors of AUCs of all classifiers are below 0.0005. We refer to the CCRs of two classes as “CCR0” and “CCR1”, respectively. We consider the expansion parameters $r = 1, 1.2, \ldots, 2.9, 3, 5, 7, 9$ for the PE-PCD classifiers, and $\tau = 0.1, 0.2, \ldots, 1, 2, 5, 10$ for the CS-PCD classifiers. Our hybrid PE-PCD classifiers are referred as PE-SVM, PE-kNN and PE-CCCD classifiers with alternative classifiers SVM, k-NN and CCCD, respectively. Similarly, our hybrid

CS-PCD classifiers are referred to as CS-SVM, CS-kNN and CS-CCCD classifiers.

Before the main Monte Carlo simulation, we perform a preliminary (pilot) Monte Carlo simulation study to determine the values of optimum parameters of SVM, CCCD and k-NN classifiers. The same values will be used for alternative classifiers as well. We train the g_{svm}, g_{cccd} and g_{knn} classifiers, and classify the test data sets for each classifier to find the optimum parameters. We perform Monte Carlo replications until the standard error of all AUCs are below 0.0005 and record which parameter produced the maximum AUC among the set of all parameters in a trial. Specifically, on each replication, we (i) classify the test data set with each θ value (ii) record the θ values with maximum AUC and (iii) update the count of the recorded θ values. Finally, given a set of counts associated with each θ value, we appoint the θ with the maximum count as the θ^*, the optimum θ (or the best performing θ). Later, we use θ^* as the parameter of alternative classifier g_{cccd} in our main simulations. Optimal parameter selection process is similar for classifiers g_{knn} and g_{svm} associated with the parameters k and γ.

The optimum parameters of each simulation setting is listed in Table 5.1. We consider parameters of SVM $\gamma = 0.1, 0.2, \ldots, 4.0$, of CCCD $\theta = 0, 0.1, \ldots, 1$ (here, $\theta = 0$ is actually equivalent to $\theta = \epsilon$, the machine epsilon), and of k-NN $k = 1, 2, \ldots, 30$. In Table 5.1, as q and d increases, optimal parameters γ and θ decrease whereas k increases. In Chapter 4, we showed that dimensionality d may affect the imbalance between classes when the supports overlap. Observe that in Table 5.1, with increasing d, optimal parameters are more sensitive to the changes in imbalance level q. For the CCCD classifier, $\theta = 1$ is usually preferred when the data set is imbalanced, i.e. $q = 0.1$ or $q = 0.5$. Bigger values of θ are better for the classification of imbalanced data sets, since with $\theta = 1$, the cover of the minority class is substantially bigger which increases the domain influence of the points of the minority class. For θ closer to 0, the class cover of the minority class is much smaller compared the class cover of the majority class, and hence, the CCR1 is much smaller. Bigger values of parameter k is also detrimental for imbalanced data sets, the bigger the parameter k, the more likely a new point is classified as class of the majority class since the points tend to be labelled as the class of the majority of k neighboring points. As for the parameter γ, support vectors have more influence over the domain as γ decreases (Wang et al., 2003). Note that $\gamma = 1/(2\sigma^2)$ in the radial basis function (RBF) kernel. The smaller the γ, the bigger the σ. Hence more points are classified as the majority class with decreasing γ since the majority class has more influence. Thus, bigger values of γ is better for the imbalanced data sets.

Table 5.1: Optimum parameters for SVM, CCCD and k-NN classifiers used in the hybrid PE-PCD classifiers.

d	q	θ (CCCD)	k (k-NN)	γ (SVM)
	0.1	1	1	3.8
2	0.5	1	1	4.0
	1.0	0	3	0.1
	0.1	1	1	2.3
3	0.5	1	1	0.4
	1.0	0	4	0.2
	0.1	1	1	0.9
5	0.5	1	4	0.3
	1.0	1	10	0.1

Average of AUCs and CCRs of three hybrid PE-PCD classifiers are presented in Figure 5.3. For $q = 0.1$, the classifier PE-kNN, for $q = 0.5$, the classifier PE-CCCD and, for $q = 1.0$, the classifier PE-SVM performs better than others. Especially, when the data set is imbalanced, the CCR1 determines the performance of a classifier; that is, generally, the better a method classifies the minority class, the better the method performs overall. When the data is balanced (i.e. $q = 1$), PE-SVM is expected to perform well, however it is known that SVM classifiers are confounded by the imbalanced data sets (Akbani et al., 2004). Moreover, when $q = 0.1$, PE-kNN performs better than PE-CCCD. The reason for this is hybrid PE-PCD classifiers incorporate alternative classifiers for points outside of the convex hull and kNN might perform better for these points. The kNN classifier is prone to missclassify points closer to the decision boundary when the data is imbalanced, and we expect points outside the convex hull to be far away from the decision boundary in our simulation setting.

In Figure 5.3, CCR1 increases while CCR0 decreases for some settings of q and d, and vice versa for some other settings. Recall that Theorem 3.5.1.2 shows a stochastic ordering of the expansion parameter r; that is, with increasing r, there is an increase in the probability of exact MDS being less than or equal to some $\kappa = 1, \ldots, d+1$. Hence with increasing r, the proximity region $\mathcal{N}_{PE}(x, r)$ gets bigger and the cardinality of the prototype set S_j gets lower. Therefore, we achieve a bigger cover of the minority class and more reduction in the majority class. The bigger the cover, the higher the CCR1 is in the imbalanced data sets. However, the decrease in the performance, when r increases, may suggest that alternative classifiers perform better for these settings. For example, the CCR1 of PE-SVM increases as r increases for $q = 0.1, 0.5$ and $d = 2, 3$, but CCR1 of PE-CCCD and PE-kNN decreases for $r \geq 1.6$. The higher the r, the more the reduction in data set. However, higher values of r may confound the classification performance. Hence, we choose an optimum value of r. Observe that for $d = 5$, the AUCs of all hybrid PE-PCD classifiers are equal for all r. With increasing dimensionality, the probability that a point of the target class falling in the convex hull of the non-target class decreases, hence most points remain outside of the convex hull.

In Figure 5.4, we illustrate the AUC measures of hybrid CS-PCD classifiers. Apparently, the τ values to do not change the performance when $d = 5$. Note that, with increasing dimensionality, the probability of target class points residing in the convex hull of non-target class points are quite slim. Hence, in hybrid CS-PCD classifiers, CS-PCD pre-classifier makes a "no decision" for almost all points which makes alternative classifier to classify these points. However, for $d = 2, 3$, τ values between 0.1 and 1 perform similarly while higher values of τ, i.e. $\tau = 10$, perform relatively good for imbalanced data sets and for classifiers CS-kNN and CS-CCCD. Increas-

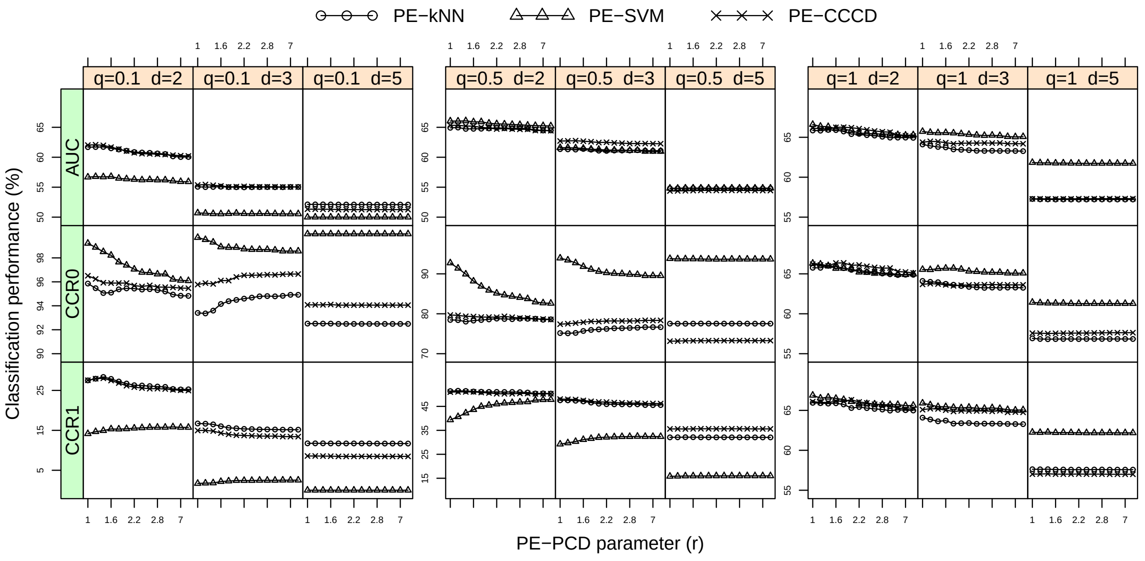


Figure 5.3: AUCs and CCRs of the three hybrid PE-PCD classifiers versus expansion parameter $r = 1, 1.2, \ldots, 2.9, 3, 5, 7, 9$ and the alternative classifiers: CCCD, k-NN and SVM. Here, the classes are drawn as $\mathcal{X}_0 \sim U([0,1]^d)$ and $\mathcal{X}_1 \sim U([\nu, 1+\nu]^d)$ with several simulation settings based on $\zeta = 0.5$ given the Equation 5.6, imbalance level $q = 0.1, 0.5, 1$, and dimensionality $d = 2, 3, 5$.

ing the τ value from 5 to 10, we observe a sudden change in CCR0 AND CCR1 for $(d,q) = (2, 0.1)$ and $(d,q) = (2, 0.5)$. However, the overall AUC increases with $\tau = 10$. Hence, the performance substantially increases while achieving more reduction in the data set. This is due to the fact that, the bigger the τ value, the bigger the proximity region, and hence the less the cardinality of the prototype set. The hybrid classifier of CS proximity maps and SVM, i.e. CS-SVM, appears to perform worst when the classes are imbalanced. When $q = 0.1$ and $q = 0.5$, CS-CCCD performs the best. When a data set is has imbalanced classes, CCR of the minority class represent the overall success of a classifier the best. Hence, looking at CCR1, we observe that CS-kNN and CS-CCCD works better in imbalanced cases. Although it is known that the performance of k-NN classifiers deteriorate with imbalanced classes, hybrid CS-PCD classifiers works fine since k-NN is only used for points outside of the class cover of CS proximity maps, where the data is not locally imbalanced.

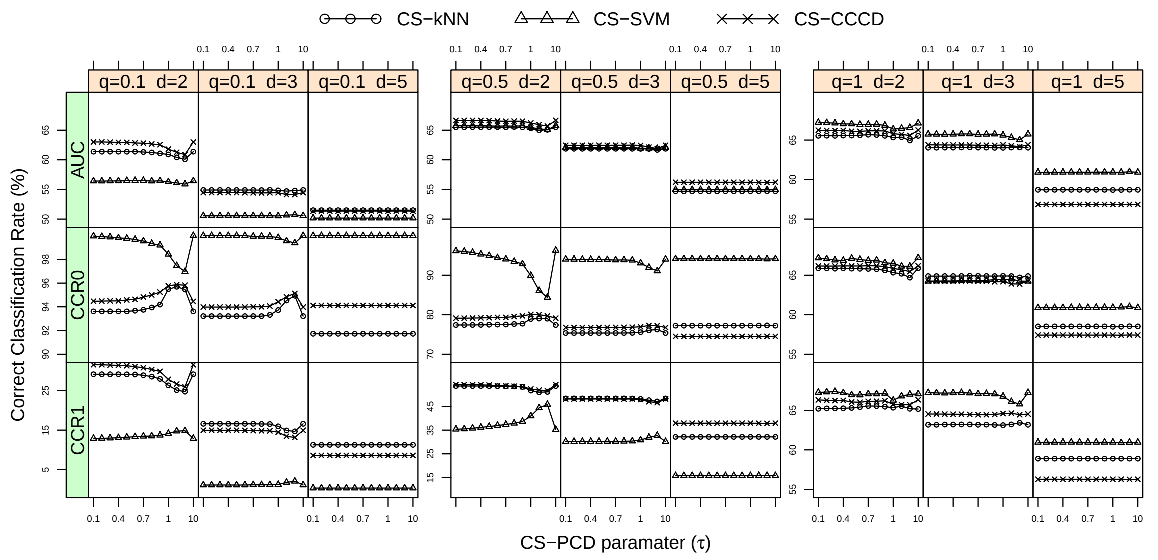


Figure 5.4: CCRs (CCRall, CCR0 and CCR1) of the three hybrid CS-PCD classifiers given with proximity maps $\mathcal{N}_{CS}(\cdot, \tau)$ and the alternative classifiers: CCCD, k-NN and SVM. Here, the class are drawn as $\mathcal{X}_0 \sim U(0,1)^d$ and $\mathcal{X}_1 \sim U(\nu, 1+\nu)^d$ with several simulation settings based on $\zeta = 0.5$, imbalance level $q = 0.1, 0.5, 1$ and dimensionality $d = 2, 3, 5$. Classification performance for all expansion parameters $\tau = 0.1, 0.2, \ldots, 1, 2, 5, 10$ is illustrated.

In Figure 5.5, we compare the composite cover PE-PCD classifier and the standard cover PE-PCD classifier. The standard cover is slightly better in classifying the minority class, especially when there is imbalance between classes. In general, the standard cover PE-PCD classifier appear to have more CCR1 than the composite cover PE-PCD classifiers. However, the composite covers are better when $d = 5$. The PE-PCD class covers are surely influenced by the increasing dimensionality. Moreover, for $q = 0.1, 0.5$, we see that the CCR1 of standard cover PE-PCD classifier slightly decreases with r, even though the data set is more reduced with increasing r. Hence, we should choose an optimum value of r that can still be incorporated to both substantially reduce the data set and to achieve a good classification performance.

In Figure 5.6, we illustrate the AUC measures of the cover CS-PCD classifiers. Lower values of τ for composite cover CS-PCD classifiers considerably perform better than other classifiers. Classifiers with low τ has high CCR1 and low CCR0 which

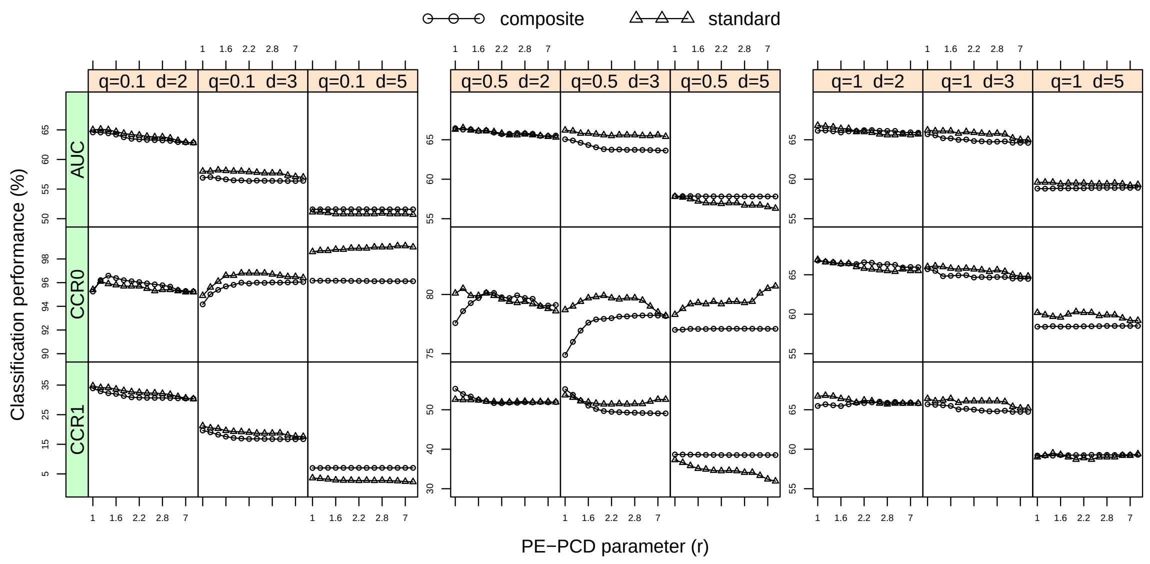


Figure 5.5: AUCs and CCRs of the two cover PE-PCD classifiers versus expansion parameter $r = 1, 1.2, \ldots, 2.9, 3, 5, 7, 9$ with composite and standard covers. Here, the classes are drawn as $\mathcal{X}_0 \sim U([0,1]^d)$ and $\mathcal{X}_1 \sim U([\nu, 1+\nu]^d)$ with several simulation settings based on $\zeta = 0.5$ given the Equation 5.6, imbalance level $q = 0.1, 0.5, 1$, and dimensionality $d = 2, 3, 5$.

indicates high classification performance for imbalanced data sets. The performance of standard classifiers do not apparently change for τ values between 0.1 and 1, but significantly change for $\tau = 2$ and higher values. It appears higher τ values performs better for imbalanced data sets, especially $\tau = 10$. However, with increasing dimensionality, composite cover CS-PCD classifiers become superior. This is of course due to same reasons why hybrid CS-PCD classifiers degrade in performance. With increasing d, convex hull of the non-target class gets sparse of target class points exponentially fast.

In Figure 5.7, we compare all five classifiers, three hybrid and two cover PE-PCD classifiers. We consider the expansion parameter $r = 3$ since, in both Figures 5.3 and 5.5, class covers with $r = 3$ perform well and, at the same time, substantially reduce the data set. For all $d = 2, 3, 5$, it appears that all classifiers show comparable performance when $q = 1$, but PE-SVM and SVM give slightly better results. However,

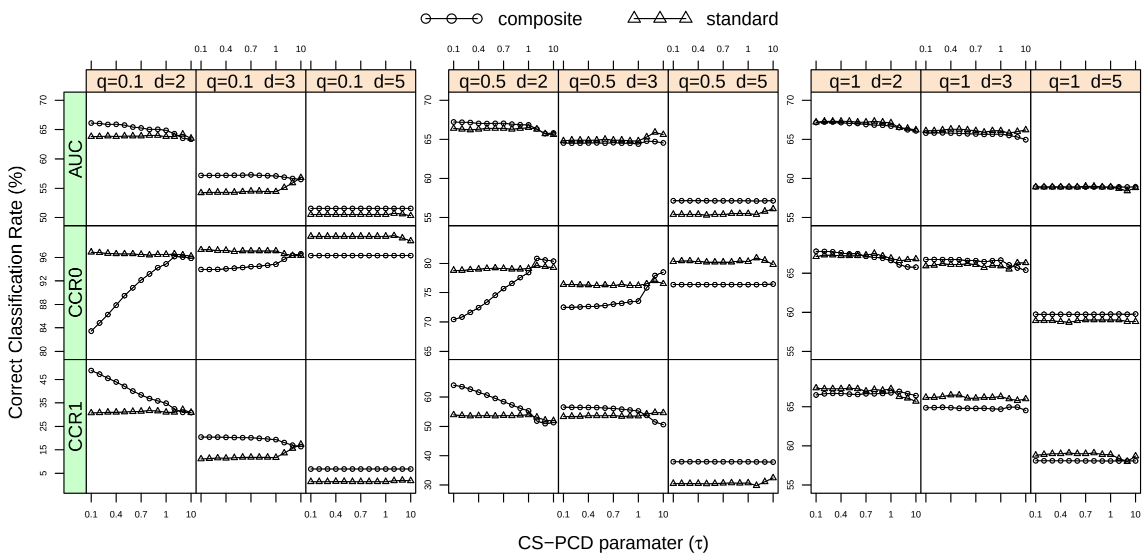


Figure 5.6: CCRs (CCRall, CCR0 and CCR1) of the two cover CS-PCD classifiers given with proximity maps $\mathcal{N}_{CS}(\cdot,\tau)$ and with composite and standard covers. Here, the class are drawn as $\mathcal{X}_0 \sim U(0,1)^d$ and $\mathcal{X}_1 \sim U(\nu,1+\nu)^d$ with several simulation settings based on $\zeta = 0.5$, imbalance level $q = 0.1, 0.5, 1$ and dimensionality $d = 2, 3, 5$. Classification performance for all expansion parameters $\tau = 0.1, 0.2, \ldots, 1, 2, 5, 10$ is illustrated.

when there is imbalance in the data sets, the performances of PE-SVM and SVM degrade, and hybrid and cover PE-PCD classifiers and CCCD classifiers have more AUC values than others. Compared to all other classifiers, on the other hand, the standard cover PE-PCD classifier is clearly the best performing one for $d = 2, 3$ and $q = 0.1, 0.5$. Observe that the standard cover PE-PCD classifier achieves the highest CCR1 among all classifiers. Apparently, the standard cover constitutes the most robust (to class imbalance) classifier. The performance of standard cover PE-PCD classifier is usually comparable to the composite cover PE-PCD classifier, but slightly better. However, for $d = 5$, the performance of standard cover PE-PCD classifier degrades and composite cover PE-PCD classifiers usually perform better. These results show that cover PE-PCD classifiers are more appealing than hybrid PE-PCD classifiers. The reason for this is that the cover PE-PCD classifiers have both good classification performance and reduce the data considerably more since

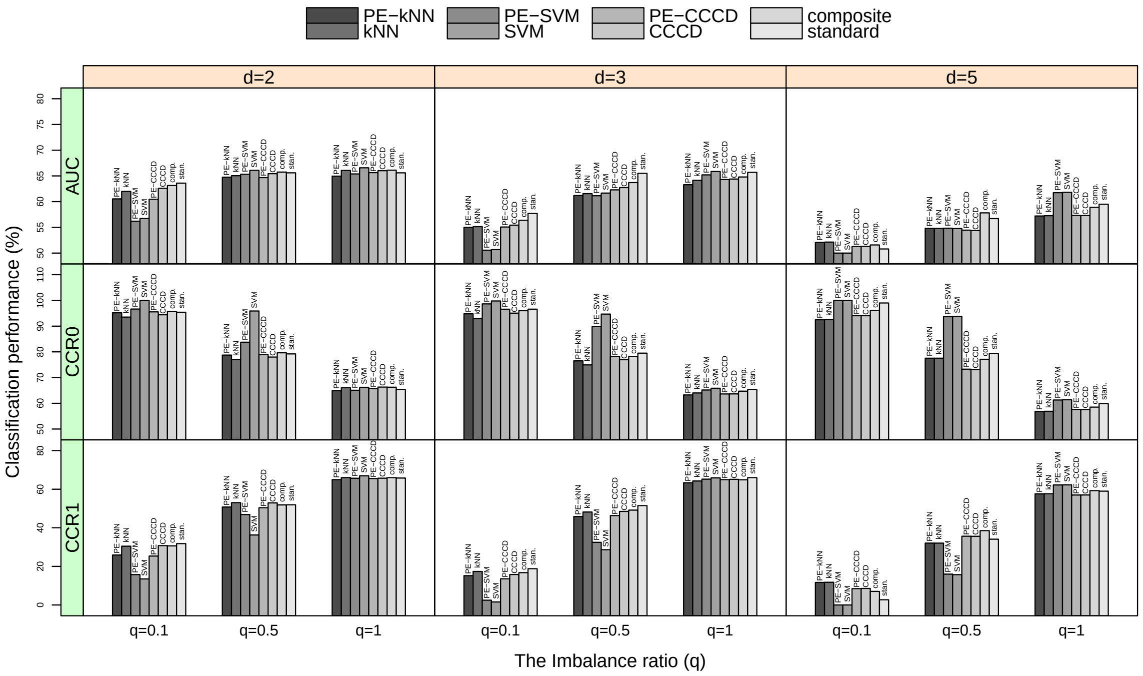


Figure 5.7: AUCs and CCRs of the two cover, three hybrid PE-PCD classifiers with expansion parameter $r = 3$, and k-NN, SVM and CCCD classifiers. The composite covers are indicated with "comp." and standard covers with "stan.". Here, the classes are drawn as $\mathcal{X}_0 \sim U([0,1]^d)$ and $\mathcal{X}_1 \sim U([\nu, 1+\nu]^d)$ with several simulation settings based on $\zeta = 0.5$, imbalance level $q = 0.1, 0.5, 1$ and dimensionality $d = 2, 3, 5$.

hybrid PE-PCD classifiers provide a data reduction for only $\mathcal{X}_j \cap C_H(\mathcal{X}_{1-j})$ whereas cover PE-PCD classifiers reduce the entire data set. The level of reduction, however, may decrease as the dimensionality of the data set increases.

In Figure 5.8, we compare all five classifiers, three hybrid and two cover CS-PCD classifiers. We consider the expansion parameter $\tau = 10$ since, in both Figure 5.4 and 5.6, class covers with higher values of τ perform well and substantially reduce the data set. Especially for $d = 3$, standard cover CS-PCD classifier with proximity map $N_{CS}(\cdot, \tau = 10)$ perform slightly better than all other classifiers since this classifier is more accurate on classifying the minority class. Hybrid and alternative classifiers based on SVMs perform the worst, and its performance is followed by classifiers based on k-NN and CCCD. However, cover CS-PCD classifiers perform the best.

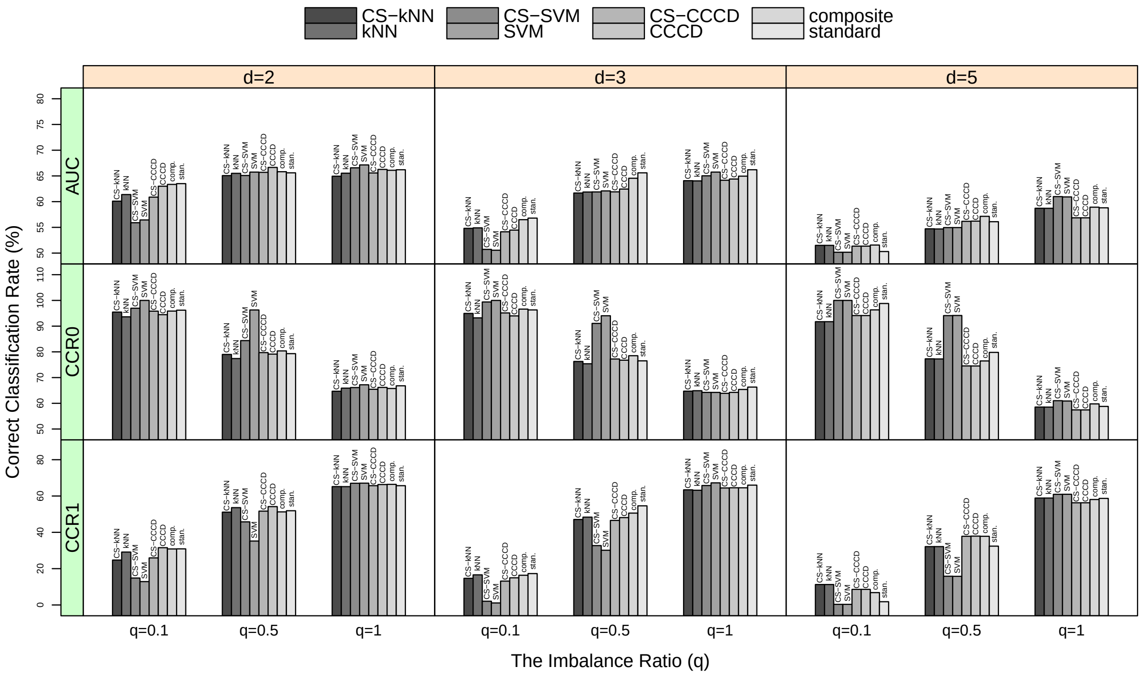


Figure 5.8: CCRs (CCRall, CCR0 and CCR1) of the two cover and three hybrid CS-PCD classifiers given with composite and standard covers for proximity maps $\mathcal{N}_{CS}(\cdot, \tau = 10)$. Here, the class are drawn as $\mathcal{X}_0 \sim U(0,1)^d$ and $\mathcal{X}_1 \sim U(\nu, 1+\nu)^d$ with several simulation settings based on $\zeta = 0.5$, imbalance level $q = 0.1, 0.5, 1$ and dimensionality $d = 2, 3, 5$. Classification performance for all expansion parameters $\tau = 0.1, 0.2, \ldots, 1, 2, 5, 10$ is illustrated.

The performance of standard cover CS-PCD classifier is usually comparable to the composite cover CS-PCD classifier, but slightly more.

In Figure 5.9, we compare all five classifiers, three hybrid and two cover PE-PCD classifiers in a slightly different simulation setting where there exists an inherent class imbalance. We perform simulation studies wherein equal number of observations $n = n_0 = n_1$ are drawn from separate distributions where $\mathcal{X}_0$ is a random sample from a multivariate uniform distribution $U([0,1]^d)$ and $\mathcal{X}_1$ is from $U([0.3, 0.7]^d)$ for $d = 2, 3, 5$ and $n = 50, 100, 200, 500$. Observe that the support of one class in entirely inside of the other, i.e. $s(F_1) \subset s(F_1)$. The same simulation setting have been used to highlight the robustness of CCCD classifiers to imbalanced data sets in Chapter 4. In Figure 5.9, the performance of kNN and PE-kNN classifiers degrade as d increases

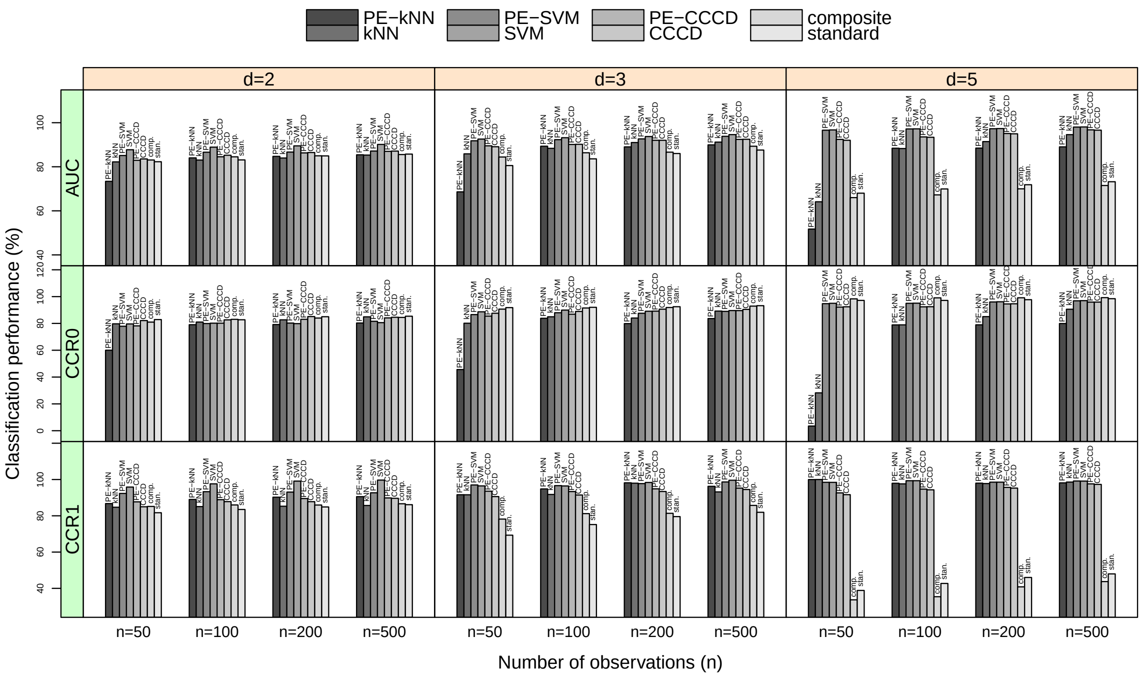


Figure 5.9: AUCs and CCRs of the two cover, three hybrid PE-PCD classifiers with expansion parameter $r = 2.2$, and k-NN, SVM and CCCD classifiers. The composite covers are indicated with “comp.” and standard covers with “stan.”. Here, the classes are drawn as $\mathcal{X}_0 \sim U([0,1]^d)$ and $\mathcal{X}_1 \sim U([0.3,0.7]^d)$ with several simulation settings based on number of observarions $n = 50, 100, 200, 500$ and dimensionality $d = 2, 3, 5$.

and n decreases. With sufficiently high d and low n, the minority class $\mathcal{X}_0$ is sparsely distributed around the overlapping region of class supports $s(F_1) \cap s(F_0)$ which is the support of $\mathcal{X}_1$. Hence, although the number of observations are equal in both classes, there exists a “local” imbalance between classses. However, CCCD and SVM classifiers, including the associated hybrid PE-PCD classifiers perform fairly good. Although the cover PE-PCD classifiers have considerably less CCR1, they perform relatively good compared to other classifiers and generally have more CCR0 than other classifiers. Similar to other simulation settings, cover PE-PCD classifiers are also affected by the increasing dimensionality of this data set.

In Figure 5.10, we compare all five classifiers, three hybrid and two cover CS-PCD classifiers in a slightly different simulation setting where there exists an inherent local class imbalance. We perform simulation studies wherein equal number of observations $n = n_0 = n_1$ are drawn from separate distributions where $\mathcal{X}_0$ is a random sample from a multivariate uniform distribution $U([0,1]^d)$ and $\mathcal{X}_1$ is from $U([0.3,0.7]^d)$ for $d = 2,3,5$ and $n = 50,100,200,500$. Observe that the support of one class in entirely inside of the other, i.e. $s(F_1) \subset s(F_1)$. Observe that the local imbalance is assumed to occur around the region of overlap $s(F_1) = s(F_1) \cap s(F_0)$ where a deterioration in the classification performance is presumably seen for sufficiently high d and low n. Hence, in this setting the class $\mathcal{X}_0$ becomes the minority class with respect to the overlapping region $s(F_1)$. In Figure 5.10, the performance of kNN and PE-kNN classifiers degrade as d increases and n decreases. However, CCCD and SVM classifiers, including the associated hybrid CS-PCD classifiers perform fairly good. Although the cover CS-PCD classifiers have considerably less CCR1, they perform relatively good compared to other classifiers and generally have more CCR0 (i.e. the CCR0 of the minority class) than other classifiers. Similar to other simulation settings, cover CS-PCD classifiers are also affected by the increasing dimensionality of this data set.

Although the PE-PCD based standard cover classifiers are competitive in classification performance, a case should be made on how much they reduce the data sets during the training phase. In Figure 5.11, we illustrate the percentage of reduction in the training data set, and separately, in both minority and majority classes, using PE-PCD for $r = 1,2,3$. The overall reduction increases with r, which is also indicated by Theorem 3.5.1.2, and the reduction in the majority class is much more than in minority class when $q = 0.1, 0.5$ since proximity regions of the majority class catch more points unlike the minority class. The majority class is reduced over nearly %60 when $q = 0.1$, and %40 when $q = 0.5$. Indeed, the more the imbalance between classes, the more the reduction in the abundantly populated classes. On the other hand, as the dimensionality increases, composite covers reduce the data set more than the standard covers. The number of the facets and simplices increases exponentially with d, and hence the cardinality of MDS (or the prototype set) also increases exponentially with d (see Theorem 3.5.1.4). As a result, composite PE-PCD covers achieve much

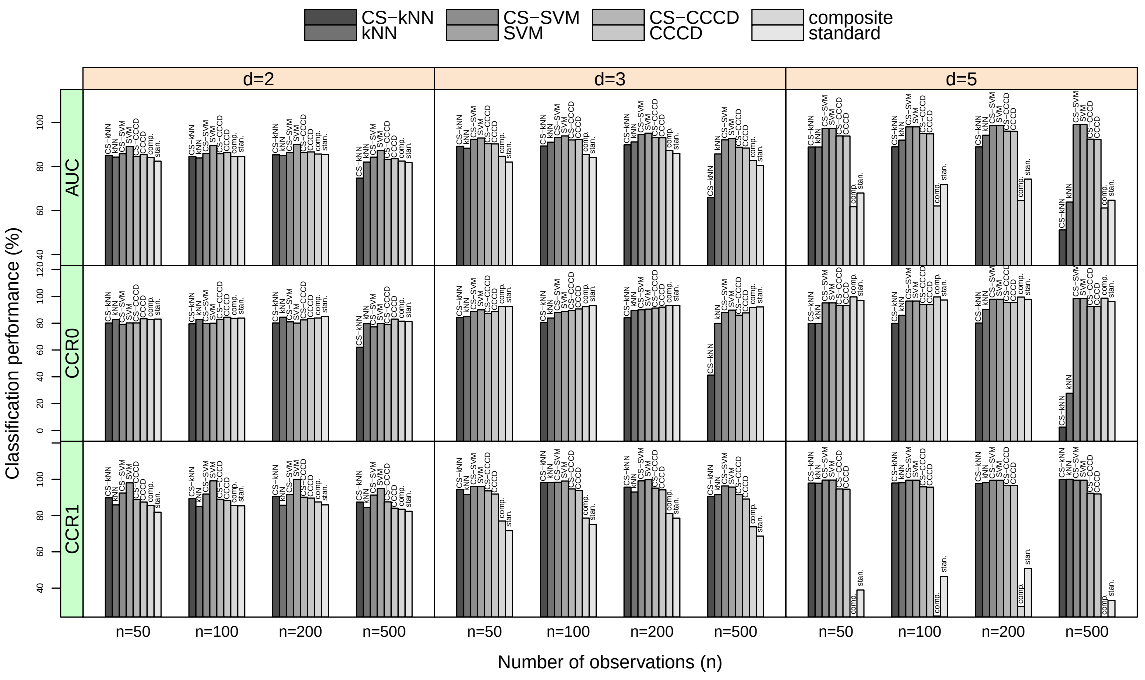


Figure 5.10: AUCs and CCRs of the two cover, three hybrid CS-PCD classifiers with expansion parameter $\tau = 10$, and k-NN, SVM and CCCD classifiers. The composite covers are indicated with "comp." and standard covers with "stan.". Here, the classes are drawn as $\mathcal{X}_0 \sim U([0,1]^d)$ and $\mathcal{X}_1 \sim U([0.3,0.7]^d)$ with several simulation settings based on number of observarions $n = 50, 100, 200, 500$ and dimensionality $d = 2, 3, 5$.

more reduction than standard PE-PCD covers.

In Figure 5.12, we plot the percentage of reduction in the training data set, and separately, in both minority and majority classes, using the map $\mathcal{N}_{CS}(\cdot, \tau)$ for $\tau = 0.5, 1, 2, 10$. The overall reduction increases with τ, and the reduction in the majority class is much more then in minority class when $q = 0.1, 0.5$ since proximity regions of the majority class catch more points unlike the minority class. The more the imbalance between classes, the more the reduction in the abundantly populated class. In Chapter 4, it has been showed that the class covers of the majority class substantially reduce the number of observations of the majority class, balancing the number of both classes. However, compared to composite covers, standard covers provide almost no reduction to either minority or majority class. CS proximity maps

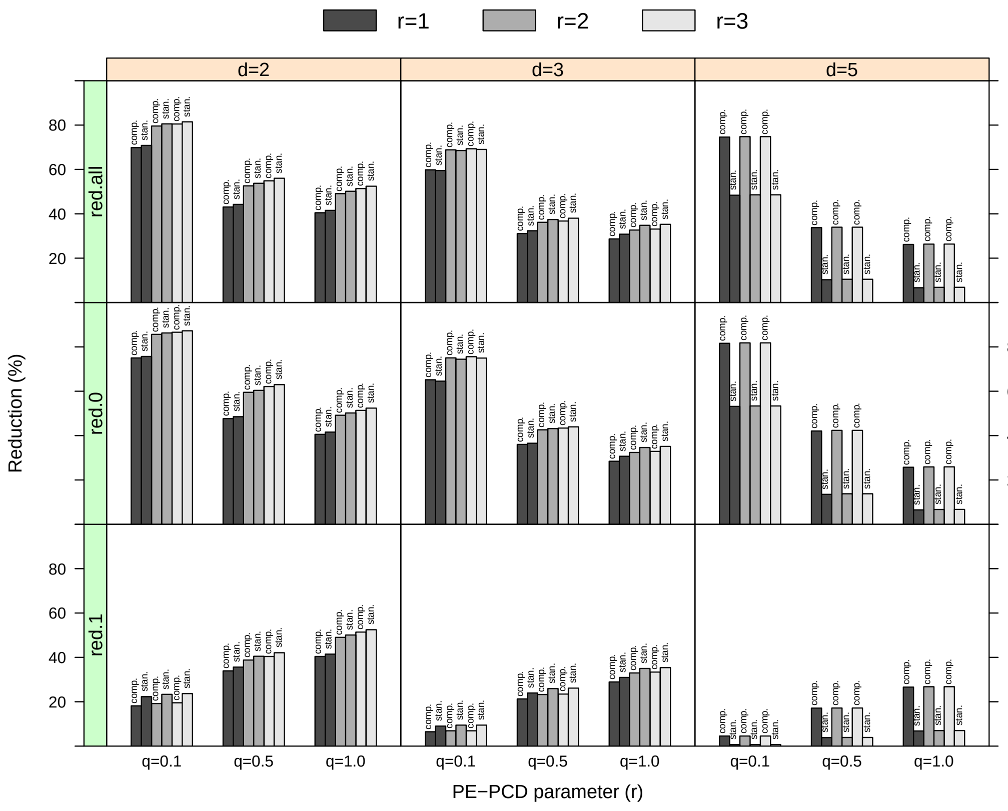


Figure 5.11: The percentage of reduction of the composite (comp.) and standard (stan.) PE-PCD covers. The "red.all" indicates the overall reduction in the training data set, $1-(|S_0+S_1|/(n_0+n_1))$, "red.0" the reduction in the $\mathcal{X}_0$ class, $1-(|S_0|/n_0)$, and "red.1" the reduction in the $\mathcal{X}_1$ class, $1-(|S_1|/n_1)$. Here, the classes are drawn as $\mathcal{X}_0 \sim U([0,1]^d)$ and $\mathcal{X}_1 \sim U([\nu, 1+\nu]^d)$ with several simulation settings based on $\zeta = 0.5$, imbalance level $q = 0.1, 0.5, 1$ and dimensionality $d = 2, 3, 5$.

do not substantially reduce the data set, even though they may potentially provide good estimations of the class supports.

5.5 Real Data Examples

In this section, we apply the hybrid and cover PCD (PE-PCD and CS-PCD) classifiers on UCI and KEEL data sets (Alcalá-Fdez et al., 2011; Bache and Lichman, 2013). We start with a trivial but a popular data set, `iris`. This data set is composed of 150 flowers classified into three types based on their petal and sepal lengths. Hence

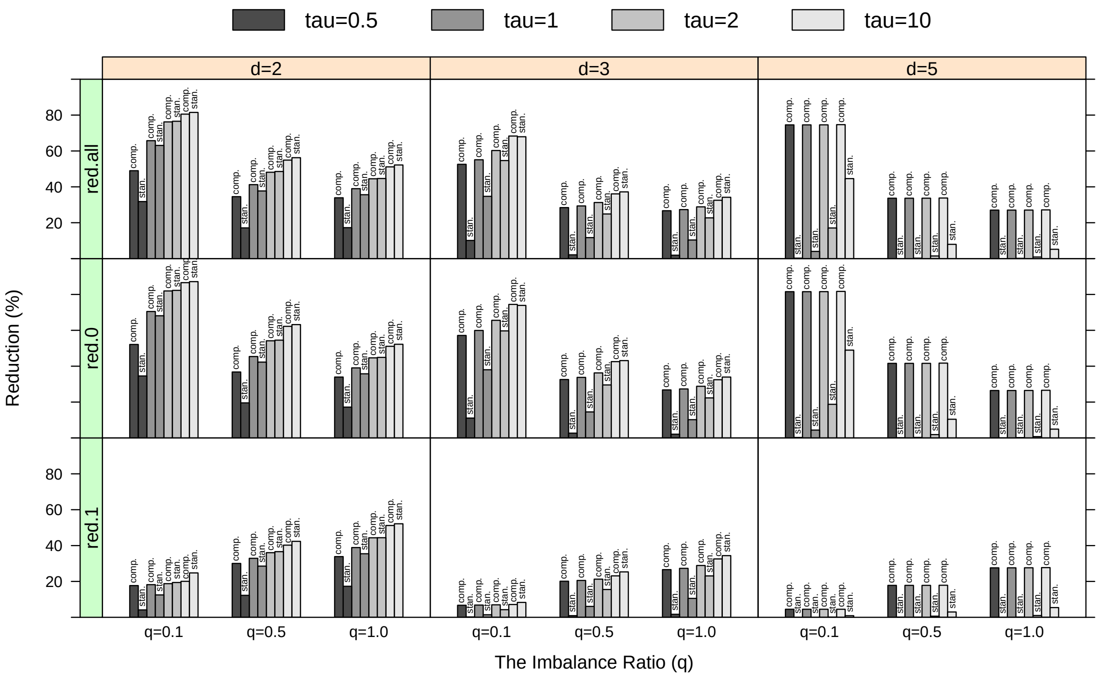


Figure 5.12: The percentage of reduction of the composite (comp.) and standard (stan.) cover CS-PCD classifiers given with proximity maps $\mathcal{N}_{CS}(\cdot, \tau)$. Here, the class are drawn as $\mathcal{X}_0 \sim U(0,1)^d$ and $\mathcal{X}_1 \sim U(\nu, 1+\nu)^d$ with several simulation settings based on $\zeta = 0.5$, imbalance level $q = 0.1, 0.5, 1$ and dimensionality $d = 2, 3, 5$.

it constitutes a nice example for class covers of multi-class data sets. In Figure 5.13, we illustrate standard and composite of PE-PCD covers, and CCCD covers of the first and the third variables of iris data set, sepal and petal lengths. We refer to this data set as `iris13`. Observe that in composite covers of Figure 5.13(c), only a few or no triangles are used to cover the setosa and virginica classes. Points of these classes are almost all outside of the convex hull of the versicolor class points, and hence covered mostly by spherical proximity regions. However, the standard covers of Figure 5.13(d) and (f) cover setosa and virginica classes with polygons since these classes are in the outer triangles of the convex hull of the versicolor class. Here, the polygons of CS-PCD covers appear to successfully estimate the class supports; that is, proximity regions, or polygons, are central and covered by only two prototypes.

To test the difference between the AUC of classifiers, we employ the 5x2 paired

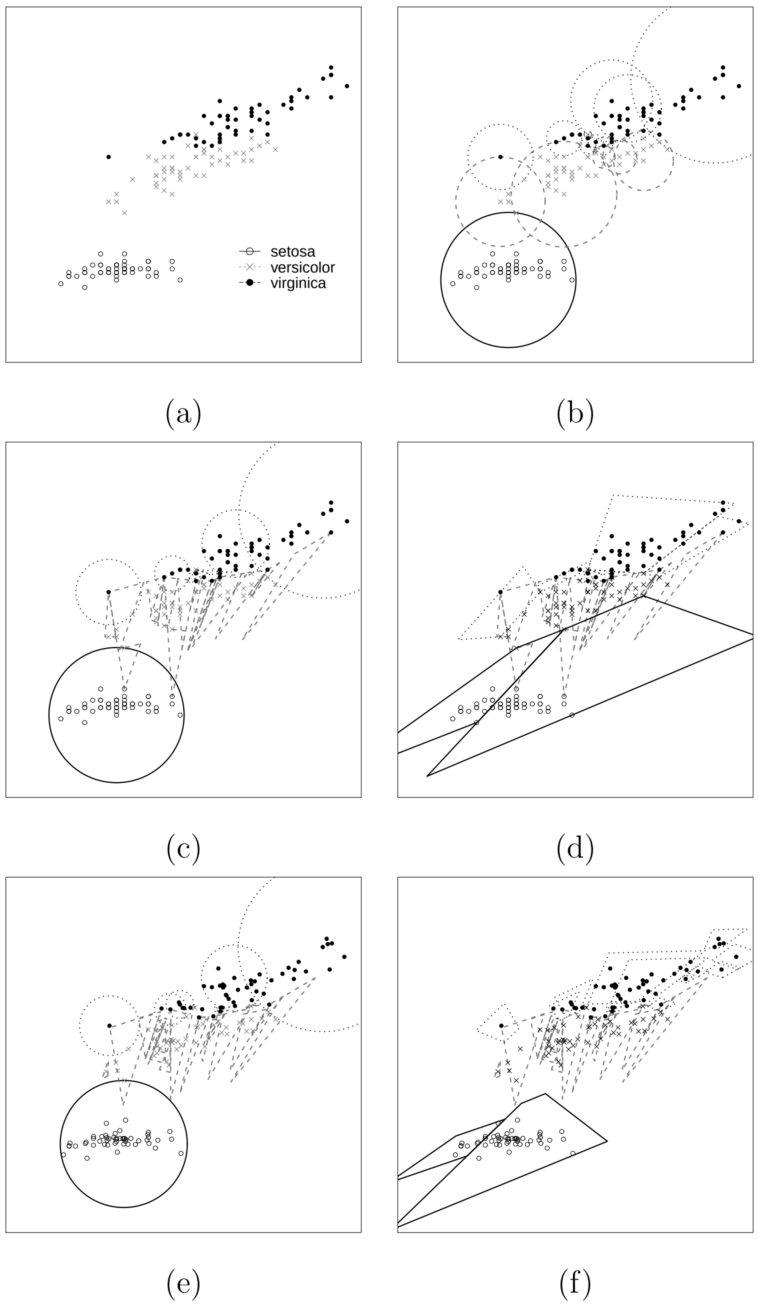


Figure 5.13: Class covers of `iris13` data set. (a) The data set with variables sepal and petal length. (b) Standard covers with $\mathcal{N}_S(\cdot, \theta = 1)$, (c) composite covers with $\mathcal{N}_I(\cdot) = \mathcal{N}_{PE}(\cdot, r = 1)$ and $\mathcal{N}_O(\cdot) = \mathcal{N}_S(\cdot, \theta = 1)$, (d) standard covers with $\mathcal{N}_I(\cdot) = \mathcal{N}_O(\cdot) = \mathcal{N}_{PE}(\cdot, r = 1)$, (e) composite covers with $\mathcal{N}_I(\cdot) = \mathcal{N}_{CS}(\cdot, \tau = 1)$ and $\mathcal{N}_O(\cdot) = \mathcal{N}_S(\cdot, \tau = 1)$ and (f) standard covers with $\mathcal{N}_I(\cdot) = \mathcal{N}_O(\cdot) = \mathcal{N}_{CS}(\cdot, \tau = 1)$.

cross validation (CV) paired t-test and the combined 5x2 CV F-test (Alpaydın, 1999; Dietterich, 1998). Refer to Section 4.6 for a review and definition of the combined 5x2 CV F-test.

Recall that the number of prototypes increases exponentially with d as shown by Theorem 3.5.1.4. Simulation studies in Section 5.4 also indicated that the dimensionality of a data set affects the classification performance. Hence, we apply dimension reduction to mitigate the effects of dimensionality. We use principal component analysis (PCA) to extract the principal components with high variance. For `iris`, let us incorporate the first two principal components with two highest variance. We refer to this new data set with two variables as `irispc2`. The information on the four variables of `iris` data set has projected onto two dimensions, and we expect that standard cover PCD classifiers works better than that in `iris` data set.

We give the AUC measures of PE-PCD classifiers compared to other classifiers on `iris13`, `iris` and `irispc2` data set in Table 5.2 and the p-values of the 5x2 CV F-test in Table 5.3. All classifiers perform well in classifying all three `iris` data sets. Although hybrid PE-PCD classifier (PE-kNN, PE-SVM and PE-CCCD) perform comparable to other kNN, SVM and CCCD classifiers, they seem to perform slightly better than the hybrid PE-PCD classifiers. Since `iris` data set and its variants in Table 5.2 are well separated and the classes are balanced, it is not surprising that kNN and SVM performs better. In `iris13` data set, standard cover PE-PCD classifier produces comparable AUC to other hybrid and cover PE-PCD classifiers. For example, standard cover PE-PCD classifier has nearly 0.05 AUC less than PE-kNN classifier in CV repetitions 1 and 3; but, on the other hand, 0.05 more AUC than PE-kNN in repetition 5. However, in `iris` data set, standard cover PE-PCD classifier has significantly much less AUC (about 0.1 AUC less) than other classifiers. Observe that $d = 2$ in `iris13` data set, but $d = 4$ in `iris` data set. Since the complexity of the class cover increases with dimensionality, the class cover of the standard cover PE-PCD classifier becomes less appealing. Although the composite cover PE-PCD classifier has substantially more AUC than standard cover PE-PCD classifier for `iris` data set, it still performs worse than the CCCD classifier. However, in `irispc2`, observe that AUC of the standard cover PE-PCD classifier has substantially increased compared to that in `iris` data set. Obviously, the increase in the performance of standard cover PE-PCD classifiers is a result of the low dimensionality. The lower the dimension, the less the complexity of the class cover and the fewer the number of prototype sets, and thus better the classification performance. Moreover, we also report on the optimum parameters of all classifiers in Table 5.2. It appears that, in general, θ increases, and k and γ decrease as expansion parameter r increases. As reviewed in Section 5.4, the smaller the values of k and γ, the higher the values of θ and r.

We give the AUC measures of of CS-PCD classifiers compared to other classifiers on `iris13`, `iris` and `irispc2` data set in Table 5.4 and the p-values of the 5x2 CV F-test in Table 5.5. All classifiers perform considerably well in classifying the iris data set. Hybrid CS-PCD classifiers perform slightly less than k-NN, SVM and CCCD classifiers, although CS-CCCD classifier achieves 0.02 more AUC than CCCD classifier in one fold. Although composite cover CS-PCD classifier performs comparably well, standard CS-PCD classifier significantly underperforms; that is, no AUC value is more than 0.85 while all other classifiers produce at least 0.90 AUC. In `iris13` and `irispc2` data set, standard cover CS-PCD classifier performs comparable to other classifier. The standard cover CS-PCD classifiers achieves even more AUC than other classifiers in some folds, and therefore, there is no significant difference in the AUC values, as seen in Table 5.3. Dimensionality of the data set shows a great deal of importance; that is, the lower the dimensionality, the lower the complexity of the standard cover, and hence the lower complexity of the associated CS-PCD classifier. We observe that optimum τ values are ¿ 2. High τ values generate big proximity regions around prototypes that have high domain of influence.

Cover PE-PCD classifiers perform better if the data has low dimensionality. Hence, we reduce the dimensionality of data sets by means of say, PCA, and then classify the data set with the cover PE-PCD classifiers trained over this data set in the reduced dimension. The `Ionosphere` data set has 34 variables. We refer to the `Ionosphere`

Table 5.2: AUC measures of the best performing (ones with their respective optimum paramaters) hybrid and cover PE-PCD classifiers for three variants of both `iris` and `Ionosphere` data sets.

		PE-kNN		kNN		PE-SVM		SVM		PE-CCCD		CCCD		Composite		Standard	
Data		Fo. 1	Fo. 2	Fo. 1	Fo. 2	Fo. 1	Fo. 2	Fo. 1	Fo. 2	Fo. 1	Fo. 2	Fo. 1	Fo. 2	Fo. 1	Fo. 2	Fo. 1	Fo. 2
	opt.	$r = 2.8$	$k = 10$	$k = 10$		$r = 2.8$	$\gamma = 3.1$	$\gamma = 3.1$		$r = 2.8$	$\theta = 0.1$	$\theta = 0.1$		$r = 2.8$	$\theta = 0.1$	$r = 2.8$	
	1	0.95	0.97	0.93	0.97	0.95	0.90	0.96	0.90	0.92	0.96	0.92	0.96	0.95	0.92	0.92	0.92
	2	0.88	0.96	0.92	0.99	0.88	0.96	0.92	0.99	0.85	0.96	0.89	0.96	0.89	0.97	0.87	0.94
`iris13`	3	0.96	0.86	0.99	0.93	0.96	0.86	1.00	0.95	0.96	0.86	0.99	0.89	0.97	0.88	0.95	0.90
	4	0.91	0.96	0.96	0.96	0.92	0.93	0.96	0.93	0.88	0.96	0.93	0.96	0.92	0.96	0.91	0.95
	5	0.96	0.88	0.95	0.93	0.91	0.88	0.87	0.92	0.96	0.88	0.92	0.89	0.95	0.88	0.93	0.91
	opt.	$r = 2$	$k = 8$	$k = 8$		$r = 2$	$\gamma = 0.1$	$\gamma = 0.1$		$r = 2$	$\theta = 0.6$	$\theta = 0.6$		$r = 2$	$\theta = 0.6$	$r = 2$	
	1	0.96	0.97	0.97	0.97	0.93	0.99	0.95	0.99	0.96	0.97	0.97	0.97	0.97	0.92	0.76	0.76
	2	0.95	0.97	0.95	0.97	0.95	0.97	0.93	0.97	0.92	0.97	0.92	0.97	0.91	0.97	0.71	0.70
`iris`	3	0.97	0.92	0.97	0.95	0.97	0.92	0.97	0.94	0.97	0.92	0.97	0.95	0.97	0.85	0.84	0.81
	4	0.96	0.96	0.96	0.97	0.95	0.92	0.95	0.92	0.96	0.92	0.96	0.93	0.96	0.96	0.71	0.76
	5	0.96	0.93	0.97	0.93	0.95	0.93	0.96	0.93	0.91	0.92	0.91	0.92	0.92	0.95	0.75	0.77
	opt.	$r = 4$	$k = 3$	$k = 3$		$r = 4$	$\gamma = 0.8$	$\gamma = 0.8$		$r = 4$	$\theta = 0.8$	$\theta = 0.8$		$r = 4$	$\theta = 0.8$	$r = 4$	
	1	0.94	0.89	0.97	0.99	0.94	0.86	0.97	0.96	0.94	0.89	0.96	0.97	0.93	0.91	0.87	0.89
	2	0.92	0.93	0.95	0.97	0.92	0.92	0.92	0.96	0.92	0.93	0.95	0.97	0.92	0.96	0.93	0.89
`irispc2`	3	0.90	0.95	0.96	0.96	0.90	0.95	0.96	0.96	0.90	0.93	0.97	0.96	0.95	0.95	0.91	0.95
	4	0.88	0.96	0.94	0.96	0.88	0.93	0.92	0.93	0.88	0.96	0.94	0.97	0.95	0.96	0.95	0.95
	5	0.97	0.90	0.97	0.95	0.97	0.90	0.99	0.95	0.95	0.90	0.95	0.95	0.93	0.93	0.91	0.93
	opt.	$r = 1.3$	$k = 9$	$k = 9$		$r = 1.3$	$\gamma = 0.9$	$\gamma = 0.9$		$r = 1.3$	$\theta = 0.1$	$\theta = 0.1$		$r = 1.3$	$\theta = 0.1$	$r = 1.3$	
	1	0.75	0.73	0.76	0.75	0.77	0.76	0.78	0.77	0.76	0.74	0.76	0.75	0.72	0.70	0.76	0.72
	2	0.72	0.74	0.73	0.78	0.71	0.76	0.73	0.79	0.71	0.76	0.74	0.76	0.71	0.74	0.73	0.76
`Ionopc2`	3	0.80	0.73	0.82	0.72	0.79	0.72	0.82	0.73	0.74	0.72	0.74	0.72	0.75	0.68	0.78	0.70
	4	0.74	0.76	0.78	0.77	0.76	0.73	0.79	0.72	0.73	0.75	0.77	0.74	0.71	0.73	0.71	0.72
	5	0.75	0.74	0.78	0.75	0.75	0.76	0.78	0.77	0.74	0.72	0.75	0.72	0.74	0.74	0.75	0.72
	opt.	$r = 1.9$	$k = 6$	$k = 6$		$r = 1.9$	$\gamma = 2$	$\gamma = 2$		$r = 1.9$	$\theta = 0.4$	$\theta = 0.4$		$r = 1.9$	$\theta = 0.4$	$r = 1.9$	
	1	0.87	0.83	0.88	0.84	0.88	0.82	0.89	0.82	0.86	0.80	0.86	0.80	0.86	0.81	0.88	0.80
	2	0.81	0.83	0.81	0.85	0.83	0.82	0.84	0.83	0.81	0.83	0.83	0.83	0.84	0.79	0.81	0.80
`Ionopc3`	3	0.83	0.81	0.84	0.81	0.82	0.86	0.84	0.86	0.81	0.84	0.83	0.83	0.85	0.84	0.83	0.84
	4	0.79	0.86	0.80	0.87	0.86	0.86	0.86	0.86	0.83	0.84	0.84	0.84	0.84	0.84	0.80	0.83
	5	0.81	0.81	0.84	0.81	0.80	0.80	0.83	0.80	0.82	0.78	0.84	0.79	0.80	0.80	0.81	0.78
	opt.	$r = 1.9$	$k = 4$	$k = 4$		$r = 1.9$	$\gamma = 4$	$\gamma = 4$		$r = 1.9$	$\theta = 0$	$\theta = 0$		$r = 1.9$	$\theta = 0$	$r = 1.9$	
	1	0.88	0.84	0.88	0.84	0.94	0.89	0.94	0.90	0.92	0.83	0.92	0.83	0.87	0.81	0.86	0.84
	2	0.85	0.85	0.85	0.85	0.91	0.89	0.91	0.89	0.93	0.86	0.93	0.86	0.91	0.83	0.88	0.83
`Ionopc5`	3	0.86	0.86	0.86	0.86	0.87	0.90	0.87	0.90	0.88	0.90	0.88	0.90	0.89	0.87	0.84	0.78
	4	0.85	0.88	0.85	0.88	0.91	0.89	0.91	0.89	0.89	0.87	0.89	0.87	0.84	0.88	0.80	0.85
	5	0.84	0.86	0.84	0.86	0.91	0.94	0.91	0.95	0.89	0.84	0.89	0.84	0.84	0.84	0.81	0.78

data set with two principal components of two highest variance as `Ionopc2`, and also, with three principal components as `Ionopc3`, and with five as `Ionopc5`. We give the AUC measures of all classifiers on these dimensionaly reduced Ionosphere data sets

Table 5.3: The p-values of the 5x2 CV F test of AUC values in Figure 5.2. The p-values below 0.1 are given in boldface.

		PE-kNN	kNN	PE-SVM	SVM	PE-CCCD	CCCD	Composite	Standard
	PE-kNN		0,315	0,442	0,404	0,454	0,690	0,389	0,526
	kNN			0,227	0,498	0,251	0,545	0,439	0,506
	PE-SVM				0,285	0,367	0,549	0,305	0,270
iris13	SVM					0,315	0,420	0,540	0,447
	PE-CCCD						0,482	0,384	0,434
	CCCD							0,780	0,719
	Composite								0,656
		PE-kNN	kNN	PE-SVM	SVM	PE-CCCD	CCCD	Composite	Standard
	PE-kNN		0,403	0,627	0,635	0,402	0,708	0,628	**0,005**
	kNN			0,535	0,617	0,227	0,391	0,532	**0,003**
	PE-SVM				0,433	0,350	0,641	0,706	**0,020**
iris	SVM					0,120	0,309	0,576	**0,014**
	PE-CCCD						0,389	0,756	**0,010**
	CCCD							0,793	**0,005**
	Composite								**0,008**
		PE-kNN	kNN	PE-SVM	SVM	PE-CCCD	CCCD	Composite	Standard
	PE-kNN		0,219	0,535	0,307	0,535	0,327	0,628	0,695
	kNN			0,184	0,205	0,122	0,386	**0,066**	**0,081**
	PE-SVM				0,224	0,535	0,279	0,484	0,694
irispr2	SVM					0,178	0,356	**0,038**	0,196
	PE-CCCD						0,186	0,495	0,676
	CCCD							0,133	0,117
	Composite								0,535
		PE-kNN	kNN	PE-SVM	SVM	PE-CCCD	CCCD	Composite	Standard
	PE-kNN		0,324	0,528	0,515	0,328	0,438	**0,000**	**0,093**
	kNN			0,282	0,521	0,294	0,424	**0,028**	**0,038**
	PE-SVM				0,398	0,439	0,435	**0,045**	0,343
Ionopr2	SVM					0,419	0,434	0,137	0,301
	PE-CCCD						0,589	0,130	0,574
	CCCD							0,182	0,467
	Composite								0,118
		PE-kNN	kNN	PE-SVM	SVM	PE-CCCD	CCCD	Composite	Standard
	PE-kNN		0,430	0,638	0,507	0,727	0,693	0,655	0,656
	kNN			0,672	0,620	0,542	0,594	0,631	0,479
	PE-SVM				0,434	0,420	0,617	0,610	0,154
Ionopr3	SVM					**0,074**	0,108	0,350	**0,014**
	PE-CCCD						0,578	0,732	0,486
	CCCD							0,659	0,282
	Composite								0,584
		PE-kNN	kNN	PE-SVM	SVM	PE-CCCD	CCCD	Composite	Standard
	PE-kNN		0,500	**0,022**	**0,020**	0,548	0,548	0,618	0,223
	kNN			**0,022**	**0,020**	0,548	0,548	0,618	0,223
	PE-SVM				0,535	0,324	0,324	**0,096**	**0,062**
Ionopr5	SVM					0,338	0,338	**0,094**	**0,061**
	PE-CCCD						0,500	0,452	0,168
	CCCD							0,452	0,168
	Composite								**0,076**

Table 5.4: AUC measures of the best performing (ones with their respective optimum paramaters) hybrid and cover CS-PCD classifiers for three variants of both `iris` and `Ionosphere` data sets.

		CS-kNN		kNN		CS-SVM		SVM		CS-CCCD		CCCD		Composite		Standard	
Data		Fo. 1	Fo. 2	Fo. 1	Fo. 2	Fo. 1	Fo. 2	Fo. 1	Fo. 2	Fo. 1	Fo. 2	Fo. 1	Fo. 2	Fo. 1	Fo. 2	Fo. 1	Fo. 2
	opt.	$\tau = 5$ $k = 10$		$k = 10$		$\tau = 5$ $\gamma = 3.1$		$\gamma = 3.1$		$\tau = 5$ $\theta = 0.1$		$\theta = 0.1$		$\tau = 5$ $\theta = 0.1$		$\tau = 5$	
	1	0.91	0.97	0.96	0.97	0.89	0.97	0.95	0.97	0.87	0.97	0.87	0.97	0.91	0.96	0.91	0.92
	2	0.85	0.97	0.93	0.99	0.85	0.97	0.91	1.00	0.84	0.97	0.91	0.99	0.89	0.97	0.91	0.93
`iris13`	3	0.95	0.89	0.97	0.92	0.95	0.89	0.99	0.95	0.91	0.88	0.93	0.93	0.95	0.90	0.95	0.93
	4	0.93	0.96	0.96	0.96	0.92	0.95	0.95	0.95	0.92	0.93	0.95	0.93	0.92	0.95	0.91	0.96
	5	0.95	0.88	0.93	0.93	0.89	0.88	0.88	0.92	0.92	0.88	0.91	0.92	0.91	0.90	0.89	0.89
	opt.	$\tau = 10$ $k = 8$		$k = 8$		$\tau = 10$ $\gamma = 0.1$		$\gamma = 0.1$		$\tau = 10$ $\theta = 0.6$		$\theta = 0.6$		$\tau = 10$ $\theta = 0.6$		$\tau = 10$	
	1	0.97	0.97	0.97	0.97	0.95	0.99	0.95	0.99	0.96	0.97	0.96	0.97	0.97	0.95	0.79	0.72
	2	0.97	0.96	0.97	0.97	0.95	0.96	0.95	0.97	0.95	0.96	0.93	0.97	0.93	0.97	0.85	0.68
`iris`	3	0.99	0.93	0.99	0.95	0.99	0.93	0.99	0.96	0.99	0.93	0.99	0.95	0.99	0.85	0.85	0.82
	4	0.96	0.96	0.96	0.97	0.95	0.92	0.95	0.92	0.97	0.93	0.97	0.94	0.97	0.96	0.79	0.85
	5	0.96	0.93	0.97	0.95	0.95	0.94	0.95	0.96	0.91	0.93	0.92	0.93	0.92	0.92	0.81	0.74
	opt.	$\tau = 10$ $k = 3$		$k = 3$		$\tau = 10$ $\gamma = 0.8$		$\gamma = 0.8$		$\tau = 10$ $\theta = 0.8$		$\theta = 0.8$		$\tau = 10$ $\theta = 0.8$		$\tau = 10$	
	1	0.95	0.92	0.96	0.99	0.95	0.88	0.97	0.94	0.95	0.92	0.97	0.97	0.96	0.89	0.92	0.86
	2	0.93	0.93	0.93	0.99	0.91	0.92	0.89	0.95	0.92	0.93	0.93	0.97	0.91	0.97	0.92	0.89
`irispc2`	3	0.92	0.92	0.95	0.96	0.91	0.89	0.95	0.93	0.91	0.92	0.96	0.95	0.95	0.92	0.91	0.89
	4	0.95	0.96	0.95	0.96	0.94	0.95	0.94	0.96	0.95	0.96	0.97	0.95	0.91	0.92	0.90	0.93
	5	0.97	0.90	0.97	0.96	0.97	0.90	0.97	0.95	0.93	0.90	0.95	0.95	0.93	0.95	0.89	0.92
	opt.	$\tau = 2$ $k = 9$		$k = 9$		$\tau = 2$ $\gamma = 0.9$		$\gamma = 0.9$		$\tau = 2$ $\theta = 0.1$		$\theta = 0.1$		$\tau = 2$ $\theta = 0.1$		$\tau = 2$	
	1	0.75	0.75	0.77	0.76	0.76	0.76	0.78	0.77	0.74	0.77	0.76	0.78	0.70	0.74	0.75	0.77
	2	0.71	0.73	0.73	0.78	0.71	0.74	0.73	0.80	0.72	0.72	0.75	0.76	0.72	0.71	0.73	0.70
`Ionopc2`	3	0.76	0.70	0.82	0.70	0.76	0.70	0.82	0.72	0.76	0.72	0.76	0.75	0.75	0.67	0.77	0.67
	4	0.73	0.75	0.77	0.75	0.74	0.75	0.78	0.72	0.71	0.76	0.77	0.75	0.68	0.77	0.70	0.78
	5	0.76	0.75	0.78	0.75	0.76	0.76	0.78	0.78	0.73	0.71	0.74	0.74	0.74	0.71	0.74	0.74
	opt.	$\tau = 10$ $k = 6$		$k = 6$		$\tau = 10$ $\gamma = 2$		$\gamma = 2$		$\tau = 10$ $\theta = 0.4$		$\theta = 0.4$		$\tau = 10$ $\theta = 0.4$		$\tau = 10$	
	1	0.85	0.81	0.87	0.82	0.87	0.81	0.89	0.82	0.85	0.79	0.85	0.80	0.87	0.82	0.84	0.83
	2	0.80	0.83	0.80	0.85	0.81	0.82	0.83	0.83	0.81	0.83	0.83	0.85	0.85	0.79	0.83	0.78
`Ionopc3`	3	0.84	0.81	0.84	0.82	0.84	0.85	0.85	0.86	0.84	0.81	0.84	0.81	0.87	0.81	0.84	0.82
	4	0.80	0.85	0.82	0.86	0.84	0.87	0.85	0.87	0.85	0.84	0.86	0.85	0.81	0.83	0.78	0.81
	5	0.80	0.80	0.83	0.80	0.79	0.80	0.83	0.80	0.82	0.77	0.86	0.77	0.83	0.80	0.76	0.82
	opt.	$\tau = 10$ $k = 4$		$k = 4$		$\tau = 10$ $\gamma = 4$		$\gamma = 4$		$\tau = 10$ $\theta = 0$		$\theta = 0$		$\tau = 10$ $\theta = 0$		$\tau = 10$	
	1	0.88	0.84	0.88	0.84	0.95	0.91	0.95	0.91	0.94	0.84	0.94	0.84	0.88	0.81	0.85	0.81
	2	0.85	0.84	0.85	0.84	0.90	0.89	0.90	0.89	0.93	0.86	0.93	0.86	0.88	0.82	0.79	0.79
`Ionopc5`	3	0.86	0.85	0.86	0.85	0.89	0.91	0.89	0.91	0.89	0.91	0.90	0.91	0.89	0.84	0.79	0.75
	4	0.84	0.88	0.84	0.88	0.92	0.88	0.92	0.87	0.85	0.87	0.85	0.87	0.83	0.87	0.69	0.84
	5	0.85	0.86	0.84	0.86	0.90	0.92	0.90	0.92	0.89	0.86	0.89	0.87	0.85	0.87	0.82	0.75

in Table 5.2 and the p-values of the 5x2 CV F-test in Table 5.3. In all three data sets, SVM classifiers seem to have the highest AUC values. Hybrid PE-PCD classifiers perform slightly worse compared to their corresponding classifiers which are used as alternative classifiers. However, for `Ionopc2` data set, both composite and standard cover PE-PCD classifiers have comparable performance to other classifiers. For `Ionopc3` and `Ionopc5`, on the other hand, the AUC of composite and standard cover PE-PCD classifiers relatively deteriorate compared to other classifiers. Although PE-PCD classifiers have computationally tractable MDSs and potentially have comparable performance to those other classifiers, the high dimensionality of the data sets are detrimental for these classifiers based on PE-PCD class covers.

In Table 5.6, we reduce the dimensionality and classify eleven KEEL and UCL data sets with all classifiers. All data sets, except Yeast6, achieved maximum AUC when reduced to two dimensions, and for these dimensionally low data sets, standard cover PE-PCD classifiers perform, in general, comparable to other classifiers. Observe that low dimensionality mitigates the effects on the complexity of the standard cover, and hence, a relatively good classification performance is achieved. Hybrid PE-PCD classifiers usually perform slightly worse then their alternative classifier counterparts. However, the hybrid PE-PCD classifier PE-kNN increases the AUC of k-NN 0.01 AUC more.

For CS-PCD classifiers, the performance is substantially high in dimensions either $d = 2$ and $d = 3$. In Segment0 and Wine data sets, other classifiers clearly achieve more AUC than composite and standard cover CS-PCD classifiers. However, in all data sets, except Yeast6, achieved maximum AUC when reduced to two dimensions, and for these dimensionally low data sets, standard cover PE-PCD classifiers perform, in general, comparable to other classifiers. Observe that low dimensionality mitigates the effects on the complexity of the standard covers, and hence, a relatively good classification performance is achieved. Hybrid CS-PCD classifiers usually perform slightly worse then their alternative classifier counterparts. However, the hybrid CS-PCD classifier PE-kNN increases the AUC of k-NN 0.01 AUC more.

5.6 Conclusions and Discussion

We use PCDs to construct semi-parametric classifiers. These families of random geometric digraphs constitute class covers of a class of interest (i.e. the target class) in order to generate decision-boundaries for classifiers. PCDs are generalized versions of CCCDs. For imbalanced data sets, CCCDs showed better performance than some other commonly used classifiers in previous studies, as also showed in Chapter 4 (DeVinney et al., 2002). CCCDs are actually examples of PCDs with spherical proximity maps. Our PCDs, however, are based on simplicial proximity maps, e.g. PE proximity maps. Our PCD class covers are extended to be unions of simplicial and polygonal regions whereas original PCD class covers were composed of only simplicial regions. The most important advantage of these family of PE proximity maps is that their respective digraphs, or namely PE-PCDs, have computationally tractable MDSs. The class covers of such digraphs are minimum in complexity, offering maximum reduction of the entire data set with comparable and, potentially, better classification performance. On the other hand, tractability of MDSs of CS-PCDs are still an open problem.

The PE-PCDs and CS-PCDs are defined on the Delaunay tessellation of the non-target class (i.e. the class not of interest). These PCDs, and associated proximity maps, were only defined for the points inside of the convex hull of the non-target class, $C_H(\mathcal{X}_{1-j})$, in previous studies. Here, we introduce the outer simplices associated with facets of $C_H(\mathcal{X}_{1-j})$ and thus extend the definition of both PE and CS proximity maps to these outer simplices. Hence, the class covers of PE-PCDs and CS-PCDs apply for all points of the target class $\mathcal{X}_j$. PE-PCDs and CS-PCDs are based on the regions of simplices associated with the vertices and faces of these simplices, called M-vertex regions and M-face regions, respectively. We characterize these vertex regions with barycentric coordinates of target class points with respect to the vertices of the d-simplices. However, the barycentric coordinates only apply for the target class points inside the $C_H(\mathcal{X}_{1-j})$. For those points outside the convex hull, we may incorporate the generalized barycentric coordinates of, for example, Warren (1996). Such coordinate systems are convenient for locating points outside $C_H(\mathcal{X}_{1-j})$ since outer simplices are similar to convex d-polytopes even though they are unbounded. However, generalized barycentric coordinates of the points with respect to these convex polytopes are not unique. Hence, properties on MDSs and convex distance measures are not well-defined.

PE-PCD class covers are low in complexity; that is, by finding the MDSs of these PE-PCDs, we can construct class covers with minimum number of proximity regions. The MDS, or the prototype set, is viewed as a reduced data set that potentially increases the testing speed of a classifier. CCCDs have the same properties, but only for data sets in $\mathbb{R}$. By extending outer intervals, i.e. intervals with infinite end points, to outer simplices in $\mathbb{R}^d$ for $d > 1$, we established classifiers having the same appealing properties of CCCDs in $\mathbb{R}$. The expansion parameter r of the PE proximity maps substantially decreases the cardinality of the MDS, but the classification performance decreases for very large r. Hence, an optimal choice of r value is in order. On the other hand, the complexity of the prototype set increases exponentially with d, the dimensionality of the data set. This fact is due to the Delaunay tessellation of the non-target class since the number of simplices and facets increases exponentially on d (see Theorem 3.5.1.4). Therefore, these class covers become inconvenient for modelling the support of the class for high d. We employ dimensionality reduction, e.g. principal components analysis, to mitigate the effects of the dimensionality. Hence, the classification performance substantially increases with these dimensionally reduced data sets as shown in Section 5.5. The Monte Carlo simulations and experiments in Section 5.4 also indicate that PE-PCDs have good reduction percentage in lower dimensions.

We define two types of classifiers based on PCDs, namely hybrid and cover PCD classifiers. In hybrid PCD classifiers, alternative classifiers are used when PCD pre-classifiers are unable to make a decision on a query point. These pre-classifiers are only defined by the simplices provided in the Delaunay tessellation of the set $\mathcal{X}_{1-j}$, hence only for target class points in $C_H(\mathcal{X}_{1-j})$. We considered alternative classifiers

k-NN, SVM and CCCD. The cover PCD classifiers, on the other hand, are based on two types of covers: *composite* covers where the target class points inside and outside of the convex hull of the non-target class are covered with separate proximity regions, and *standard* covers where all points are covered with regions based on the same family of proximity maps. For composite covers, we consider a composition of spherical proximity maps (used in CCCDs) and PE proximity maps. Results on both hybrid and cover PCD classifiers indicate that when the dimensionality is low and classes are imbalanced, standard cover PCD classifiers achieve either comparable or slightly better classification performance than others. We show that these classifiers are better in classifying the minority class in particular. This makes cover PCD classifiers appealing since they present slightly better performance than other classifiers (including hybrid PCD classifiers) with a high reduction in the data set.

PE-PCDs offer classifiers of (exact) minimum complexity based on estimation of the class supports. The MDSs are computationally tractable, and hence, the maximum reduction is achieved in polynomial time (on the size of the training data set). This property of PE-PCDs, however, achieved by partitioning of $\mathbb{R}^d$ by Delaunay tessellation, and as a result, the number of the simplices and facets of the convex hull of the non-target class determines the complexity of the model which increases exponentially fast with the dimensionality of the data set. Indeed, this leads to an overfitting of the data set. We employ PCA to extract the features with the most variation, and thus reduce the dimensions to mitigate the effects of dimensionality. PCA, however, is one of the oldest dimensionality reduction method, and there are many dimension reduction methods in the literature that may potentially increase the classification performance of PCD classifiers. Moreover, PE-PCDs are one of many family of PCDs using simplicial proximity maps investigated in Ceyhan (2010). Their construction is also based on the Delaunay tessellations of the non-target class, and similar to PE-PCDs, they enjoy some other properties of CCCDs in $\mathbb{R}$, and they can also be used to establish PCD classifiers. However, our work proves the idea that relatively good performing classifiers with minimum prototype sets can be provided with

PCDs based on partitioning schemes (e.g. Delaunay tessellations), but we believe an alternative partitioning method, say for example a rectangular partitioning scheme, that produces less partitioning than a Delaunay tessellation would be more appealing for the class cover. Such schemes could also have computationally tractable MDSs. Such classifiers and their classification performance are topics of ongoing research.

Table 5.5: The p-values of the 5x2 CV F test of AUC values in Figure 5.4. The p-values below 0.1 are given in bold font.

		CS-kNN	kNN	PE-SVM	SVM	PE-CCCD	CCCD	Composite	Standard
	PE-kNN		0.445	0.511	0.466	0.387	0.661	0.680	0.711
	kNN			0.237	0.281	0.272	0.526	0.134	0.116
	PE-SVM				0.245	0.409	0.470	0.464	0.778
iris13	SVM					0.244	0.424	0.219	0.360
	PE-CCCD						0.318	0.506	0.699
	CCCD							0.708	0.708
	Composite								0.651
		CS-kNN	kNN	CS-SVM	SVM	CS-CCCD	CCCD	Composite	Standard
	CS-kNN		0.323	0.566	0.626	0.604	0.655	0.535	**0.014**
	kNN			0.421	0.557	0.365	0.395	0.438	**0.013**
	CS-SVM				0.535	0.366	0.433	0.501	**0.038**
iris	SVM					0.207	0.154	0.468	**0.040**
	CS-CCCD						0.684	0.590	**0.043**
	CCCD							0.553	**0.050**
	Composite								**0.064**
		CS-kNN	kNN	CS-SVM	SVM	CS-CCCD	CCCD	Composite	Standard
	CS-kNN		0.390	0.323	0.550	0.535	0.430	0.651	0.379
	kNN			0.306	0.352	0.149	0.657	0.378	0.169
	CS-SVM				0.301	0.489	0.264	0.517	0.578
irispr2	SVM					0.196	0.369	0.194	0.188
	CS-CCCD						0.130	0.438	0.296
	CCCD							0.262	0.113
	Composite								0.311
		CS-kNN	kNN	CS-SVM	SVM	CS-CCCD	CCCD	Composite	Standard
	CS-kNN		0.385	0.298	0.364	0.337	0.285	0.489	0.764
	kNN			0.444	0.599	0.504	0.779	0.372	0.557
	CS-SVM				0.361	0.379	0.323	0.452	0.758
Ionopr2	SVM					0.473	0.651	0.414	0.558
	CS-CCCD						0.392	0.268	0.646
	CCCD							0.404	0.622
	Composite								0.167
		CS-kNN	kNN	CS-SVM	SVM	CS-CCCD	CCCD	Composite	Standard
	CS-kNN		0.342	0.348	0.283	0.727	0.645	0.749	0.704
	kNN			0.558	0.512	0.511	0.676	0.787	0.633
	CS-SVM				0.335	0.663	0.676	0.648	0.394
Ionopr3	SVM					0.202	0.507	0.558	0.388
	CS-CCCD						0.318	0.570	0.654
	CCCD							0.597	0.607
	Composite								0.649
		CS-kNN	kNN	CS-SVM	SVM	CS-CCCD	CCCD	Composite	Standard
	CS-kNN		0.535	**0.061**	**0.089**	0.301	0.236	0.714	0.114
	kNN			**0.055**	**0.082**	0.315	0.245	0.693	0.133
	CS-SVM				0.535	0.460	0.462	0.151	**0.090**
Ionopr5	SVM					0.508	0.514	0.182	0.104
	CS-CCCD						0.535	0.274	**0.082**
	CCCD							0.177	**0.075**
	Composite								0.154

Table 5.6: Average of ten folds of 5x2 CV F-test AUC values of all classifiers on eleven KEEL and UCL data sets. The symbol "*" indicate a difference with the AUC of standard cover PCD classifier at significant level of 0.1, and "**" at level 0.05. "PCd" indicates the number of principal components used.

Data	PC d	PE-kNN	kNN	PE-SVM	SVM	PE-CCCD	CCCD	Composite	Standard
iris	2	0,924	0,962*	0,918	0,952	0,920	0,959	0,939	0,918
Ionosphere	2	0,747*	0,763**	0,752	0,767	0,737	0,746	0,720	0,735
New-Thyroid1	2	0,962	0,965	0,947	0,950	0,960	0,963	0,966	0,963
New-Thyroid2	2	0,977	0,977	0,948	0,948	0,986	0,986	0,986	0,969
Segment0	3	0.740**	0.907**	0.896**	0.910**	0.858**	0.842**	0.752	0.745
Shuttle0vs4	2	0,995	0,995	1,000	1,000	0,998	0,998	0,998	0,997
Wine	2	0,974**	0,975**	0,971**	0,972**	0,965	0,965	0,955	0,950
Yeast4	2	0,579	0,588	0,555	0,504**	0,569	0,564	0,562	0,553
Yeast5	2	0,711	0,721	0,675	0,624	0,688	0,683	0,668	0,666
Yeast6	3	0,687*	0,676*	0,621	0,557	0,655	0,641	0,594	0,613
Yeast1289vs7	2	0,559	0,547	0,548	0,503	0,552	0,535	0,546	0,549

Data	PC d	CS-kNN	kNN	CS-SVM	SVM	CS-CCCD	CCCD	Composite	Standard
iris	2	0.935	0.962	0.922	0.945	0.929	0.957	0.931	0.903
Ionosphere	3	0.819	0.831	0.830	0.843	0.821	0.832	0.828	0.811
New-Thyroid1	2	0.965	0.965	0.950	0.950	0.963	0.963	0.967	0.976
New-Thyroid2	2	0.977	0.977	0.948	0.948	0.986	0.986	0.986	0.973
Segment0	3	0.892**	0.908**	0.893**	0.907**	0.847**	0.842**	0.722	0.723
Shuttle0vs4	3	1.000	1.000	0.995	0.995	0.997	0.997	0.997	0.998
Wine	2	0.975**	0.975**	0.972**	0.972**	0.966	0.965	0.943	0.951
Yeast4	2	0.586	0.604	0.544	0.504	0.565	0.572	0.549	0.546
Yeast5	2	0.730	0.728	0.689	0.633	0.708	0.693	0.684	0.682
Yeast6	2	0.639	0.640	0.606	0.523	0.619	0.618	0.612	0.602
Yeast1289vs7	3	0.571*	0.563*	0.521	0.500	0.551	0.539	0.527	0.524

Data	N	d	$q = m/n$	k	γ	θ	r	τ
iris	150	4	2.00	3	0.8	0.8	4.0	10
Ionosphere	351	35	1.78	9	0.9	0.1	1.3	10
New-Thyroid1	215	5	5.14	5	2.5	1.0	2.5	0.2
New-Thyroid2	215	5	5.14	4	3.5	1.0	4.0	2.0
Segment0	2308	20	6.02	30	3.3	1	1.5	10
Shuttle0vs4	1829	10	13.87	1	0.1	0.2	1.1	0.8
Wine	178	13	2.00	24	1.0	0.4	1.4	0.8
Yeast4	1484	9	28.10	1	0.7	1.0	7.0	10
Yeast5	1484	9	32.70	6	2.5	1.0	1.0	10
Yeast6	1484	9	41.40	1	2.3	1.0	9.0	10
Yeast1289vs7	1484	9	30.70	1	4.0	0.3	4.0	10

Part III

Clustering

Chapter 6

PARAMETER-FREE CLUSTERING WITH CLUSTER CATCH DIGRAPHS

6.1 Introduction

Clustering is one of the most challenging tasks in machine learning, and perhaps, discovering the exact number of clusters of an unlabelled data set is the most important problem. Many clustering methods find the clusters (or hidden classes) and the number of these clusters simultaneously (Sajana et al., 2016). Although there exist methods to validate and compare the quality of a partitioning of a data set, algorithms that provides the (estimated) number of clusters without any input parameter are still appealing. However, such methods or algorithms rely on other parameters viewed as the intensity, i.e. expected number of objects in a unit area, of presumably existing clusters. Choice of such parameters are often challenging since different values of such parameters may drastically change the result. We use unsupervised adaptations of a family of CCCDs, that showed relatively good performance in statistical classification (Priebe et al., 2003a). Unsupervised versions of CCCDs are called *cluster catch digraphs* (CCDs). Primarily, CCDs use statistics that require an intensity parameter to be specified. We approach this problem by incorporating spatial data analysis tools that estimate the spatial intensity of regions in the domain. We propose algorithms using Ripley's K function with CCDs to establish density-based clustering methods that find the optimal partitioning of unlabelled data sets, without the definition of any priori parameter.

Clustering algorithms based on CCDs are, very similar to density-based clustering methods, which can find the exact number of arbitrarily shaped cluster in a data sets (Sajana et al., 2016). Jarvis-Patrick method and DBSCAN are some of the first density-based clustering methods (Ester et al., 1996; Jarvis and Patrick, 1973). However, all of these algorithms require prior domain knowledge on the spatial intensity of the data set. We propose slight modifications to the existing CCD methods in order to drop the necessity of knowing the intensity. We estimate the second-order moments of a set of points in a domain, and tell whether the points are clustered. Ripley (1977) introduced the K function to estimate the second-order moments and introduced tests for clustering of the spatial data set. We combine Ripley's K function with the CCDs to successfully find clusters with no prior assumption on the intensity. We show that our adaptations of CCDs do not require any domain knowledge (i.e. the number of assumed clusters or the spatial intensity) and able to detect both spherical and arbitrarily shaped clusters. Our algorithms are similar to some parameter free clustering algorithms such as DBCLASS of Xu et al. (1998) and the graph-based clustering algorithm of Streib and Davis (2011). However, they only demonstrated the performance of their algorithm in two dimensional data sets, whereas we conduct Monte Carlo and real life experiments on general family of data sets.

We introduce two clustering algorithms designed specifically for data sets with either spherical shaped or arbitrarily shaped clusters. For data sets with spherical shaped clusters, we use the approximate MDSs of the CCDs to locate possible clusters. Our adaptations of CCDs successfully separate noise clusters from the true clusters; however, we use cluster validity methods to decide which clusters are either true clusters or noise clusters. Moreover, the cluster validity methods help choosing a subset of the approximate MDS in such a way that the CCDs are efficient against data sets with unimodal clusters. Hence, the method works considerably well for data sets composed of either uniformly or normally distributed clusters. For data sets with arbitrarily shaped clusters, we find the maximal disconnected components of CCDs to locate the clusters of data sets. In both algorithms, the support of the data set is estimated with a union of covering balls which allows us to explore possible partitionings of data sets with different types of clusters. We conduct Monte Carlo simulations, as well as real data experiments, to show that the new CCDs improve over the existing clustering methods based on CCDs, and to demonstrate that the new CCDs perform comparable to the some existing methods in the literature while being parameter free.

6.2 Data Clustering

Let $(\Omega, \mathcal{M})$ be a measurable space, and let $\mathcal{X} = \{X_1, X_2, \cdots, X_n\} \subset \Omega$ be a set of Ω-valued random variables with distribution F and the support $s(F)$. Here, we assume X_i is drawn from a finite mixture of distributions $\{F_1, F_2, \cdots, F_M\}$. Thus, let $X_i \sim F := \sum_{m=1}^{\mathcal{K}} F_m \pi_m$ for $\mathcal{K}$ being the number of components of F (i.e. the number of clusters) and $\pi_m \in [0,1]$ such that $\sum_m \pi_m = 1$. The goal of a partitional clustering algorithm is to divide the data set $\mathcal{X}$ into $\hat{\mathcal{K}}$, i.e. the estimate of $\mathcal{K}$, number of disjoint subsets $\mathcal{P} = \{P_1, P_2, \cdots, P_{\hat{\mathcal{K}}}\}$ where $\mathcal{X} = \cup_{m=1}^{\hat{\mathcal{K}}} P_m$, and hence to minimize (or maximize) some objective function $h(\mathcal{P})$ that gives the "best" or "optimal" partitioning. Here, $\hat{\mathcal{K}}$ is either unknown or assumed prior to the partitioning. In reality, there is no best partitioning (and thus no best function h) of a data set $\mathcal{X}$ since the objective function h depends on the distribution F, and most importantly, on the user of the algorithm. We review some of the most frequently used types of partitional clustering algorithms for data sets with both spherical shaped and arbitrarily shaped clusters.

Prototype-based, graph-based and density-based clustering algorithms are some of the most well known families of clustering algorithms available (Gan et al., 2007). Prototype-based methods, for example the famous k-means algorithms, find an optimal partitioning of the data set assuming the number of clusters are exactly k. Graph-based and density-based algorithms, however, group the objects to similar clusters of a data set if either they are close to some high density regions of the domain or if they satisfy some similarity/dissimilarity relation. These methods mostly rely on some parameter that determines the lowest density of a point that they could be viewed as members of a cluster, which could be summarized as the spatial intensity. The clustering methods we discuss in this chapter are hybrids of graph-based and density-based algorithms. We determine the regions of high density in a data set by choosing the size and the location of the covering balls that are associated with the potential cluster centers. Once the size (or the radius) of each ball is determined, we establish geometric digraphs to find the clusters and their exact number simultaneously. We highlight the relation between our methods and other methods similar to ours, thus demonstrate the advantages of our methods. Later, we briefly describe some validity indices and methods for evaluating clustering algorithms. These indices will be useful to locate and identify true clusters, and as well as noise clusters, in the data set.

6.2.1 Algorithms for Spherical and Arbitrarily Shaped Clusters

The k-means algorithm is perhaps the most known and commonly used clustering algorithm in the literature, even though it is simple and prone to errors for some common types of data sets (Ball and Hall, 1965). Let the (estimated and presumed) number of clusters $k = \hat{\mathcal{K}}$ be given. The objective function h_k of the k-means algorithm is defined as

$$h_k(\mathcal{P}) = \sum_{i}^{n} \sum_{m}^{\hat{\mathcal{K}}} d^2(X_i, \mu(P_m)).$$

Here, $d(\cdot,\cdot)$ is the dissimilarity measure (e.g. Euclidean distance measure), and $\mu(P_m) \in \mathbb{R}^d$ is the center of the cluster P_m randomly given prior to the execution of the algorithm. At each step, $\mu(P_m)$ is updated as the current center of mass of all points in P_m, and the members of P_m as the collection of all closest points to $\mu(P_m)$. This is repeated until no considerable change is observed in $\mu(P_m)$ for all $m = 1, \cdots, \hat{\mathcal{K}}$. There are many extensions of k-means such that each mitigates the effects of some difficulties observed in real life data sets. One such algorithm is the ISODATA where the number of clusters are initialized before the execution of the algorithm; however, the clusters are merged or split during the run based on some criteria (such as minimum number of points in a cluster or the minimum distance between two clusters) (Ball and Hall, 1965). But ISODATA is sensitive to weakly separated compact clusters (i.e. clusters that are close to each other), and may merge or split existing clusters, diverging from the optimal solution. Dunn (1973) introduced two adaptations of k-means, known as fuzzy c-means, where the first is a fuzzy version of ISODATA, and the latter is a completely novel fuzzy partitioning algorithm with the objective function

$$h_c(\mathcal{P}) = \sum_{i}^{n} \sum_{m}^{\hat{\mathcal{K}}} u_{ij}^q d^2(X_i, \mu(P_m)).$$

The objective function h_c is slightly different than the function h_k where u_{im} denotes the magnitude of membership of X_i in cluster P_m, and $q > 1$ is some parameter to adjust fuzziness. Clustering methods with fuzzy membership functions are more robust to weakly separated clusters; that is, they tend to converge to the optimal clustering solution when clusters are substantially close to each other.

Clustering methods such as k-means perform relatively well if the clusters in a data set are spherical and compact. However, all such algorithms requires to be initialized with a prior knowledge on the number of clusters which may not be estimated or guessed easily before the execution of the algorithm. In addition, these algorithms satisfy a partitioning criteria which may not be suitable for data sets with arbitrarily shaped clusters. These special types of data sets do not necessarily have centers. Density-based methods are based on capturing the density around points, and hence they estimate the true number of clusters (or latent classes). The Jarvis-Patrick algorithm is one of the first methods of this kind to capture arbitrarily shaped and non-arbitrarily shaped clusters (Jarvis and Patrick, 1973). The DBSCAN algorithm of Ester et al. (1996), and the OPTICS algorithm of Ankerst et al. (1999) work in similar fashion. Given two parameters, the radii ϵ and the minimum number of points *MinPts* in the neighborhood of a point, the data set is divided into several types of points wherein one set of points are called *core* points. The union of the neighborhoods (i.e. the points within the radius ϵ) of core points constitutes the clusters. The parameters ϵ and *MinPts* may change the result of the analysis drastically. Both of these parameters represent the *intensity* of the spatial distribution of clusters we are looking for in a data set. Kernel-based clustering algorithms are often employed for both estimating the true number of clusters and locating arbitrarily shaped clusters in a data set. The intensity of a spatial distribution is provided by the width parameter of the Gaussian kernel. One such algorithm is the *pdfCluster* where clustering depends on the density estimation of the data sets via Delaunay triangulation (or Delaunay tessellation in $\mathbb{R}^d$ for $d > 2$). Vertices of the triangulation, whose estimated pdf is below a certain cut value are removed, and the remaining connected components of the triangulation are set to be the clusters. A culmination of results from several choices of cut values are analyzed to decide the final partitioning. The density around each point is estimated with a kernel function Φ (mostly a Gaussian kernel with width h) as follows: for $y \in \mathbb{R}^d$,

$$f(y) = \sum_{i=1}^{n} \frac{1}{nh_{i,1} \cdots h_{i,d}} \sum_{j=1}^{d} \Phi\left(\frac{y - X_{i,j}}{h_{i,j}}\right).$$

where X_{ij} and h_{1-j} are the value and the kernel width, respectively, of j'th coordinate of the random variable associated with the i'th observation.

6.2.2 Validation of the Clustering Algorithms

A partitioning $\mathcal{P}$ can be validated by a variety of indices. These families of indices often divided into three major family of indices based on the criteria. These are referred to as *internal*, *external* and *relative* criteria (Gan et al., 2007). Some existing indices can be modified to be used as an index of either internal or external criteria. These are Hubert's Γ statistics and Goodman-Kruskal γ statistics. Moreover, Rand statistic, Jaccard statistic and Folkes-Mallows index are used as indices of external; and cophenetic correlation coefficient is used as an index of internal criteria. We focus on the relative criteria since our algorithms are based on these family of indices.

Indices of relative criteria are only meaningful when several clustering algorithms are to be compared. However, we use these indices to locate an appropriate subset of our approximate MDSs which correspond to the true clusters. We will show that relative indices are particularly appealing to find a set of distant and compact clusters in which both the true and noise clusters reside. Many indices of relative criteria indicate or measure some relationship between *inter-cluster* distances (the distances between the members of the same cluster) and *intra-cluster* distances (the distances between the members of different clusters). A simple index based on this criteria is the *silhouette* index (Gan et al., 2007). Given a certain partitioning of the data set, the silhouette index measures how well a data set is clustered. Let the point X_i appointed to the cluster P_m. Let $a(i)$ be the average distance from X_i to all other points of the same cluster P_m, and let $b(i)$ be the distance to the closest cluster to the point X_i where the distance between a cluster a point is given as the average distance to all points in the cluster to the point X_i. Hence, for X_i being a member of the cluster P_m,

$$a(i) := \frac{1}{|P_m|} \sum_{Y \in P_m} d(X_i, Y)$$

and

$$b(i) := \min_{i \neq m} \frac{1}{|P_m|} \sum_{Y \in P_m} d(X_i, Y).$$

Thus, the silhouette $sil(i)$ of the random variable X_i is denoted as

$$sil(i) := \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}.$$

Here, $sil(i)$ indicates how well the point X_i clustered in P_m given the partitioning $\mathcal{P}$. An overall assessment of how well the entire data set is clustered can be given by the average of silhouette of a partitioning, i.e. $sil(\mathcal{P}) = \sum_i sil(i)/n$.

The maximum average silhouette is often employed to choose the best k, or $\hat{\mathcal{K}}$, in k-means clustering method. The algorithm is executed for a set of different number of clusters $\hat{\mathcal{K}}$, and the partitioning which produces the maximum average silhouette is chosen to be the best clustering of the data set. Unlike methods such as k-means where the number of clusters should be specified before execution, we will incorporate this measure to choose a minimum subset of the MDSs of a digraph family to reveal the true clusters. Hence, the algorithm is only executed once.

6.3 Cluster Catch Digraphs

Cluster Catch Digraphs (CCDs) are unsupervised adaptations of CCCDs (Marchette, 2004). The definition of a CCD is similar to CCCDs; that is, a CCD is a digraph $D = (\mathcal{V}, \mathcal{A})$ such that $\mathcal{V} = \mathcal{X}$ and, for $u, v \in \mathcal{V}$, we have $(u, v) \in \mathcal{A}$ iff $v \in B(u, r)$. In CCCDs, however, each point $X \in \mathcal{X}$ is the center of covering ball $B(X, r(X))$ where $r(X)$ is a function on both the target and non-target classes. In CCDs, we do not have (or know the existance of) a non-target class to determine the radius. Instead, the objective is to locate, or *catch*, existing clusters around the point X, hence the name *cluster catch digraph*. In CCDs, defined by DeVinney (2003), the radius $r(X)$ is determined by maximizing a Kolmogorov-Smirnov (K-S) based statistic. We have

$$r(X) := \underset{r \in \{d(X,Z): Z \in \mathcal{X}\}}{\operatorname{argmax}} RW(X, r) - F_0(X, r). \tag{6.1}$$

Here, $RW(X, r)$ is the random walk function proportional to the percentage of points lying inside the covering ball $B(X, r)$; i.e.

$$RW(X, r) = \sum_{Z \in \mathcal{X}} I(d(X, Z) < r) \tag{6.2}$$

Here, $I(\cdot)$ is the indicator functional, and $F_0(X, r)$ is a function proportional to the null distribution that the data set tested against; that is, the distribution assuming that the points randomly fall inside the covering ball. A maximum value of $RW(X, r) - F_0(X, r)$ on the radius r indicates that the points inside $B(X, r)$ more likely constitute cluster than a random collection of points. We take $F_0(X, r) = \delta r^d$ for $\mathcal{X} \subset \mathbb{R}^d$ but other choices of the form of this function are also possible (Marchette, 2004). Here, the null hypothesis assumes that the points are randomly fell in the covering ball, then the density is proportional to the volume of the ball. However, the function $F_0(X, r)$ depends on the parameter δ which represents the intensity of the assumed (or tested against) homogeneous Poisson process. After we determine $r(X)$ for all $X \in \mathcal{X}$, we construct the CCD, D, and find the approximate MDS.

6.3.1 Clustering with CCDs

Let $\mathcal{X} = \{X_1, X_2, \cdots, X_n\}$ be a set of $\mathbb{R}^d$-valued random variables with some distribution F and the support $s(F)$, and assume X_i is drawn from a finite mixture of distributions. The objective is to find the number of these components (i.e. clusters), as well as which component X_i is more likely drawn from. We estimate the number of clusters $\mathcal{K}$, denoted as the estimand $\hat{\mathcal{K}}$, and offer algorithms to partition the data set into estimated number of disjoint sets. To do so, we use the lower complexity covers to reveal the hidden clusters provided by the covering balls associated with approximate MDSs.

Although the cover gives an estimate of some hidden class supports, each individual ball may not be equivalent to an hidden class since CCD (similar to CCCDs) are vector quantization methods, i.e. the support is modeled by a set of prototypes (Marchette, 2004). We exploit some relationship between the set of covering balls and prototypes to estimate possibly existing clusters. We assume that covering balls of the same cluster are more likely to intersect each other as, by construction, their centers would be closer to each other. Hence, MDSs, found by the Algorithm 2, are more appealing since the resulting covering balls are more likely to intersect than ones found by the Algorithm 1. We consider two covering balls intersect if and only if the neighborhoods of vertices in CCDs share at least one vertex. Let $D = (\mathcal{V}, \mathcal{A})$ be a graph such that, for $v \in \mathcal{V}$, we define the neighborhood of v as $N(v) := \{u \in \mathcal{V} : (v, u) \in \mathcal{A}\}$. Therefore, let the graph $G_{MD} = (\mathcal{V}_{MD}, \mathcal{E}_{MD})$ be defined by $\mathcal{V}_{MD} = S_{MD}$ where, for $v \in \mathcal{V}_{MD}$ and for $v, u \in \mathcal{V}_{MD}$, we have $(v, u) \in \mathcal{E}_{MD}$ if and only if $N(v) \cap N(u) \neq \emptyset$. Hence, the graph G_{MD} is referred to as an *intersection graph* (Das et al., 1989). We use $S(G_{MD})$ to denote the MDS of G_{MD}.

The more the data sets become noisy, the more we find spurious clusters. A collection of noise in the data set may be misinterpreted as a member of separate cluster, or two or more clusters may appear as a single cluster. Ben-David and Haghtalab (2014) states that a clustering algorithm may become robust to noise; that is, the noise clusters could be separated from the true clusters by increasing the presumed number of clusters before executing the algorithm. Hence, it is viewed as the algorithm includes a “garbage collector”, a term to refer to a collection of noise in the data set. We show that silhouette measure can separate noise clusters and the true clusters with a similar fashion. We assume that the more cardinality the covering have, the closer it is to the center. Starting from the two elements of $S(G_{MD})$ with most cardinality, we incrementally add points of $S(G_{MD})$ to a set S until the silhouette measure reaches to a maximum value. Here, S is the set of true cluster centers. The reason for that is, the more the silhouette measure is, the better the partitioning. If a maximum or substantially high silhouette measure is obtained, this suggest that there is no need to add more clusters, and thus, the remaining clusters are viewed potentially noise clusters.

We illustrate the cover of a data set with two clusters in Figure 6.1(a). Observe that, some covering balls associated with S_{MD} intersect each other, however there seem to be two sets of covering balls that do not intersect one another. These two sets of covering balls may represent seperate estimates of two class supports that may be the clusters we are looking for. We illustrate the covering balls of $S(G_{MD})$ in Figure 6.1(b). Although the cardinality of the set S_{MD} is seven; that is, the MDS of CCD D has seven elements, two big covering balls are those with the most cardinality. The centers of these two covering balls are elements of the MDS and they are closely located at the center of the clusters. Finally, we illustrate the covering balls of $S(G_{MD})$ and S to demonstrate the use of silhouette measure in Figure 6.2. The data sets have three normally distributed classes such that some points may have realized as noise. The silhouette measure is maximized when the three most dense covering balls added to the final set of MDSs that are the true clusters.

Some points in the data set, however, are not covered by any ball since no significant clustering have found around them, or some of them are neglected since they were the members of the noise clusters. We set the radius $r(X) = 0$ for such points, so they only cover themselves. To decide which clusters these uncovered points belong to, we find the closest covering ball. Given a convex set $\mathcal{H} \subset \Omega$, for $x \in \mathcal{H}$, we use

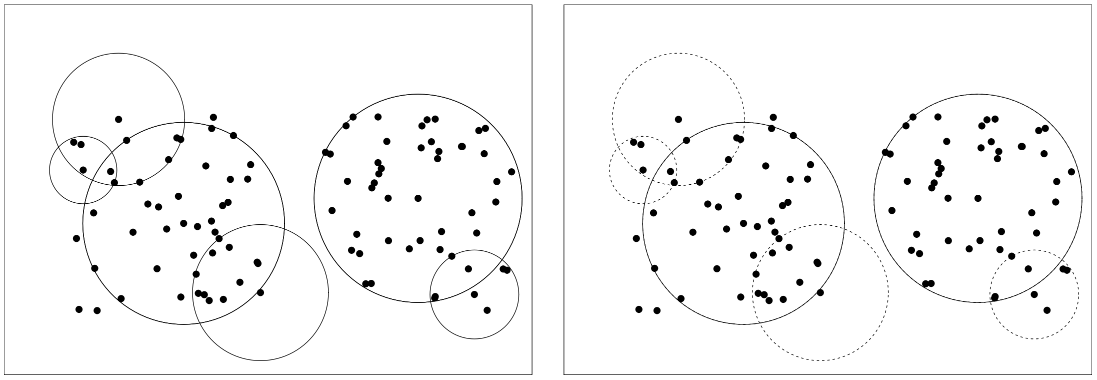


Figure 6.1: (a) Covering balls based on the points of the MDS S_{MD}. The data set is composed of two sets of the same size, $n = 50$ randomly drawn from distributions $F_0 = U([0,1]^2)$ and $F_1 = U([2.5, 3.5] \times [0,1])$. Here, the intensity parameter is $\delta = 0.2$. (b) Covering balls of $S(G_{MD})$. Balls of dominating points of $S(G_{MD})$ are given with solid lines, and balls of points of $S_{MD} \setminus S(G_{MD})$ with dashed lines.

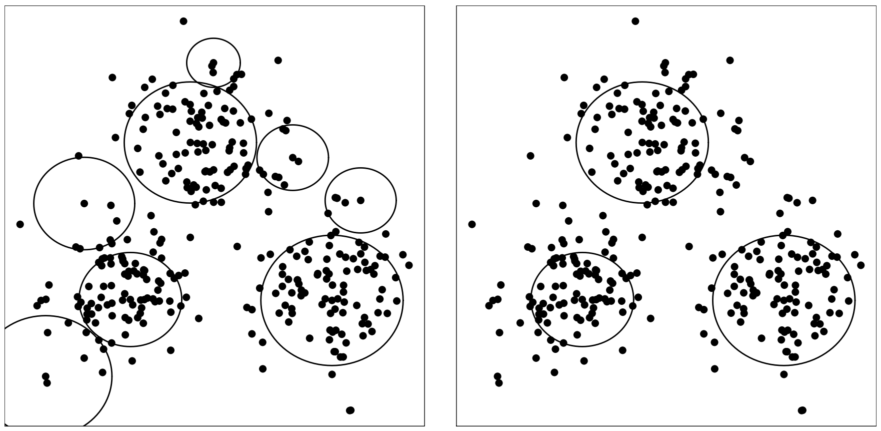
the distance function $\rho(x, z)$ of the form

$$\rho(z, \mathcal{H}) := \frac{d(z,x)}{d(y,x)}. \tag{6.3}$$

Here, y is the point of intersection of the line segment $L(x, z)$ and the boundary of the convex set $\partial(\mathcal{H})$. Some illustration on the distance $\rho(z, \mathcal{H})$ can be found in Section 5.3.1.

We give the psuedo code of the CCD algorithm with K-S based statistics in Algorithm 9. First, for all points of the data set, we choose the best radii $r(X)$ by maximizing the K-S statistics as in Equation 6.1. Later, we find the MDS, S_{MD}, which constitutes the estimates of the supports of possible hidden classes. Later, to reveal cluster centers and the number of hidden classes, we construct the intersection graph with the vertex set S_{MD}, and find its own dominating set $S(G_{MD})$. Finally, we incrementally add more points of the $S(G_{MD})$ to S until the silhouette measure is maximized. The algorithm runs in $\mathcal{O}(n^3)$ time when $d < n$.

Theorem 6.3.1.1. *Let $\mathcal{X} \subset \mathbb{R}^d$ be a data set with $n = |\mathcal{X}|$ observations. Algorithm 9 partitions the data set $\mathcal{X}$ in $\mathcal{O}(n^3 + n^2(d + \log n))$ time.*



(a) (b)

Figure 6.2: Clustering of a data set with three normally distributed classes in $\mathbb{R}^2$ (a) centers of clusters that are either the true or the noise clusters (b) the true clusters which maximize the silhouette index.

Proof: In Algorithm 9, the matrix of distances between points of training set $\mathcal{X}$ should be computed which takes $\mathcal{O}(n^2 d)$ time. For each iteration, the radius $r(x)$ that maximizes $RW(x,r) - \delta r^d$ could easily be computed by sorting the distances from each x to all other points. This sorting takes $\mathcal{O}(n \log n)$ time for each $x \in \mathcal{X}$. Also, the $r(x)$ is found on linear time. Hence, the digraph D is computed in a total of $\mathcal{O}(n^2 logn)$ time. Both Algorithm 2 and Algorithm 3 run in $\mathcal{O}(n^2)$ time in the worse case. Finally, to maximize the silhouette, for each added element of $S(G_{MD})$, we partition the data set and calculate the silhouette in $\mathcal{O}(n^2)$ time, which makes selection of silhouette take $\mathcal{O}(n^3)$ time in the worst case. ■

Here, the choice of the intensity parameter δ is of utmost importance; that is, a variety of different results can be provided by CCDs with different values of δ. Figure 6.3 illustrates the results for different values of the intensity parameter δ. Observe that the best result is obtained by $\delta = 0.2$; however, just a single cluster is found with $\delta = 0.05$ (no cluster, the data set is composed of only a single class or drawn from a single distribution), three clusters are found with $\delta = 0.4$, and three small clusters have been found for $\delta = 0.6$. Obviously, none of these δ values provide the true result other than $\delta = 0.2$. In the following section, we introduce another

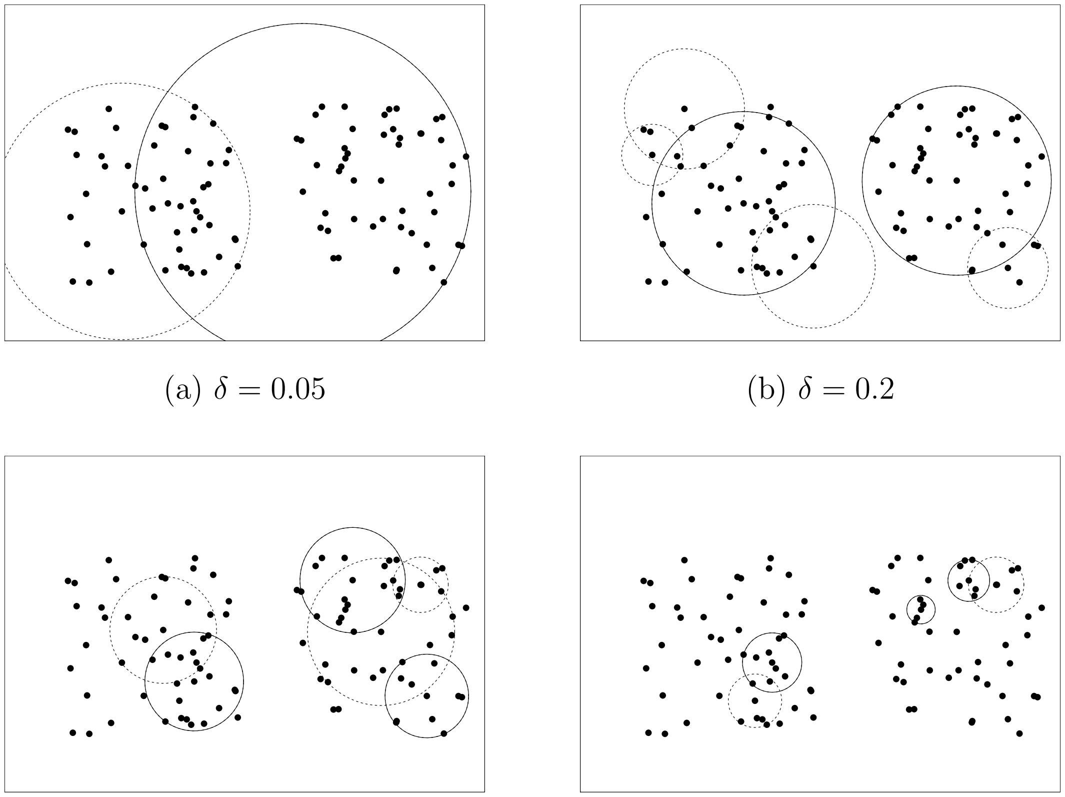
Algorithm 9 CCD clustering algorithm with K-S based statistics

```
Input: The data set $\mathcal{X}$ and the parameter $\delta \in \mathbb{R}_+$
Output: The set of centers $S$ and their corresponding radii
1: for all $x \in \mathcal{X}$ do
2:     $r(x) = \text{argmax}_{r \in \{d(x,z): z \in \mathcal{X}\}} RW(x,r) - \delta r^d$.
3: end for
4: Construct $D = (\mathcal{V}, \mathcal{A})$
5: Find $S_{MD}$ with Algorithm 2
6: Construct $G_{MD} = (\mathcal{V}_{MD}, \mathcal{E}_{MD})$
7: Find $S(G_{MD})$ with Algorithm 3 given $sc(v) = |N(v)|$
8: $S \leftarrow \emptyset$
9: for all $s \in S(G_{MD})$ do
10:     $S \leftarrow S \cup s$
11:     Let $\mathcal{P}$ be the partitioning provided by $S$
12:     if $sil(\mathcal{P})$ decrease then
13:         break
14:     end if
15: end for
```

family of CCDs that does not require any intensity parameter. Such digraphs can be used to establish parameter free clustering methods.

6.3.2 CCDs with Ripley's K function

We introduce an adaptation of CCDs employing Ripley's K function instead of K-S based statistics. In CCDs, the K-S statistics in Equation (6.1) is maximized over the radius $r(X)$ to find which radius achieves a covering ball with points inside clustered the most. We will refer to this family of digraphs as KS-CCDs throughout this chapter. K-S statistics does not test the relative distribution of data points inside covering ball, and hence may misinterpret a collection of points from two clusters



(a) $\delta = 0.05$ (b) $\delta = 0.2$

(c) $\delta = 0.4$ (d) $\delta = 0.6$

Figure 6.3: Results of Algorithm 9 with different choices of the intensity parameter δ. Here, we consider $\delta = 0.05, 0.20, 0.40, 0.60$. Covering balls of the set $S(G_{MD})$ are given with solid lines, and balls of the set $S_{MD} \setminus S(G_{MD})$ with dashed lines.

as a single cluster. On the other hand, tests based on the $\hat{K}$ function, exploits the second-order properties of a spatial distribution, and hence may potentially give a better description of clustering. We use the $\hat{K}(t)$ function to test whether points inside a ball (or a window) are from the same support or region inside has positive density.

Here, we think of each ball B as a window that we test whether points inside the ball are drawn from a homogeneous Poisson process with $\hat{\lambda} = n/\operatorname{Vol}(B)$ (i.e. complete spatial randomness), where n is the size of the data set and $\operatorname{Vol}(B)$ is the area (or volume) of the ball. Hence, we employ a spherical window that we use the translation correction to mitigate the effects of bias resulted by the edges. Let the covering ball, or the window, B has radius R, and let $SC(a, R)$ be the spherical cap with height a of the d-sphere B (a sphere of dimension d). Hence, given the distance between the pair $z, z' \in \mathbb{R}^d$ denoted by $\rho(z, z')$, let the translation correction coefficient $\vartheta(z, z')$ for Equation (2.11) be defined as:

$$\vartheta(z, z') := \frac{\mathrm{Vol}(B)}{2\,\mathrm{Vol}(SC(a, R))} = \frac{\frac{\pi^{d/2} R^d}{\Gamma(\frac{d}{2}+1)}}{2\frac{\pi^{(d-1)/2} R^d}{\Gamma(\frac{d+1}{2})} \int_0^{\arccos(\frac{\rho(z,z')}{2R})} \sin^d(t)dt}. \tag{6.4}$$

However, we achieved better performance ignoring the translation correction. The test rejects the null hypothesis, if there are significantly empty regions or there is sufficient evidence that clusters exists inside the ball. To reject the null, we do only check if the observed curve is above the upper envelope to test for possible clustering. if the curve of $\hat{K}(t)$ is below the lower curve of the envelope, the test for CSR rejected and point pattern of the data is significantly regular. However, we are not interested with regular spatial patterns at this point, since we only test if the balls are subsets of the support of hidden classes (Ripley, 1977). Moreover, we choose a t_{max}, i.e. the maximum value of distance t of $\hat{K}(t)$, as 1 which is the radius of the ball, or window, B.

The envelopes are empirically computed given a set of Monte Carlo experiments on a ball, or window, with some radius and center. However, similar to the Algorithm 9, each time the radius of a ball centered on X is increased, we have to repeat the test, and hence repeat the Monte Carlo experiment to compute envelopes. However, to decrease the computation time of the test, we rely on the invariance property of the $\hat{K}$ function under linear transformation of random sequences whose support is a ball, or the window, itself. The following theorem shows how a single set of Monte Carlo experiments could be used for multiple covering balls.

Theorem 6.3.2.1. *Let $\mathcal{Z} = \{Z_1, Z_2, \cdots, Z_n\}$ be a set of $\mathbb{R}^d$-valued random variables with support $B(\mathbf{0}_{d\times 1}, 1)$ where $B(\mathbf{0}_{d\times 1}, 1) \subset \mathbb{R}^d$ is the hyperball with radius 1 and centered on $\mathbf{0}_{d\times 1}$. Also, let $\mathcal{G}$ be a group of transformation on $\mathbb{R}^d$ such that, for all $g \in \mathcal{G}$ and $x \in \mathbb{R}^d$, $x' = g(x)$ iff $x' := ax + b$ for some $a \in \mathbb{R}$ and $b \in \mathbb{R}^d$. Therefore, we have that the tests with $\hat{K}$ function is invariant under the group $\mathcal{G}$.*

Proof: For $Z \in \mathcal{Z}$, define by $Z' = g(Z) = aZ + b$ for some $g \in \mathcal{G}$. Hence, Z' has the support $B(b, a)$. Now, let $\hat{K}(\mathcal{Z}, B, t)$ denote the $\hat{K}(t)$ associated with the set $\mathcal{Z}$ in window B, and let $\mathcal{Z}' = g(\mathcal{Z}) := \{g(Z) : Z \in \mathcal{Z}\}$. Then,

$$\hat{K}(\mathcal{Z}, B(\mathbf{0}_{d\times 1}, 1), t) = \hat{K}(\mathcal{Z}', B(b, a), at)$$

since, for all $x, y \in B(\mathbf{0}_{d\times 1}, 1)$, the statement $d(x, y) < t$ is equivalent to $d(g(x), g(y)) < at$. ■

Theorem 6.3.2.1 shows that, regardless of the center or the radius of the ball $B(X, r(X))$, $\hat{K}$ value is the same for all $X \in \mathcal{X}$ if the distance t for the test is scaled accordingly; that is, for all $B(b, a)$ for some $a \in \mathbb{R}$ and $b \in \mathbb{R}^d$ there exists an equivalent test for a ball $B(\mathbf{0}_{d\times 1}, 1)$. This fact decreases the computation time drastically such that the envelopes could be calculated. Algorithm 10 illustrates the clustering algorithm that uses CCD adaptations with Ripley's K function. We call such digraphs R-CCDs. One difference of Algorithm 9 with the KS-CCD algorithm is how the radii of the covering balls are chosen. Instead of looking at each possible ball for all $X \in \mathcal{X}$ of the data set, we incrementally check the radii until the test for CSR is rejected. Moreover, before the main loop, we simulate CSR data sets in $B(\mathbf{0}_{d\times 1}, 1)$ and calculate envelopes.

An illustration on how the method works and how the test rejects the CSR for a given points $X \in \mathcal{X}$ is given in Figure 6.4. The last panel on the far right illustrates a covering ball centered on a point from one class that encapsulates some points from the other class. As seen from the $\hat{L}(t) - t$ below, the observed curve is above the envelope for some values of t; that is, the distribution inside the covering ball significantly deviates from CSR. Hence, we stop increasing the radius, and use the previous radius as the radius of covering ball. After finding radii of all points on the condition that the supports of clusters are sufficiently far away from each other, the MDS of the intersection graph provides the clusters successfully. Although the gap between the clusters are slightly wide, the test based on the function $\hat{K}(t)$ rejects the hypothesis if the covering ball centered on some point x begins to cover regions outside of its respective cluster. In Figure 6.4, the ball gets larger until it stumbles on

Algorithm 10 CCD clustering algorithm using Ripley's K function (R-CCD).

Input: The data set $\mathcal{X}$

Output: The set of centers S and their corresponding radii

```
1: Calculate $\hat{K}(t)$ for all $N$ number of data sets simulated in $B(\mathbf{0}_{d\times 1}, 1)$
2: for all $x \in \mathcal{X}$ do
3:     $\mathbf{D}(x) := \{d(x,y) : y \in \mathcal{X} \setminus \{x\}\}$
4:     for all $r \in \mathbf{D}(x)$ in ascending order do
5:         Let $B := B(x,r)$ and $\mathcal{X}' := \mathcal{X} \cap B$
6:         $r(x) \leftarrow r$
7:         if test is rejected for $\mathcal{X}'$ then
8:             break
9:         end if
10:    end for
11: end for
12: Construct $D = (\mathcal{V}, \mathcal{A})$
13: Find $S_{MD}$ with Algorithm 2
14: Construct $G_{MD} = (\mathcal{V}_{MD}, \mathcal{E}_{MD})$
15: Find $S(G_{MD})$ with Algorithm 3 given $sc(v) = |N(v)|$
16: $S \leftarrow \emptyset$
17: for all $s \in S(G_{MD})$ do
18:     $S \leftarrow S \cup s$
19:     Let $\mathcal{P}$ be the partitioning provided by $S$
20:     if $sil(\mathcal{P})$ decrease then
21:         break
22:     end if
23: end for
```

a point from another cluster or if there are some empty or sparsely populated regions. In Figure 6.5, we illustrate some example data sets in $\mathbb{R}^2$ of the results of R-CCDs.

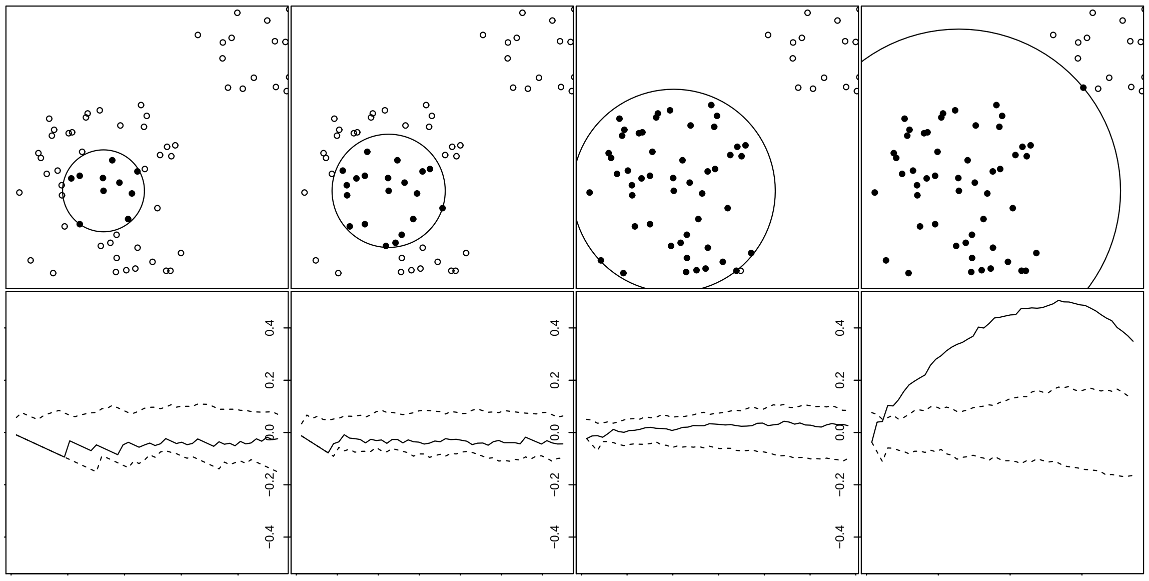


Figure 6.4: Four snapshots of the growing balls. Solid points are those inside the covering ball (top). The envelopes of four snapshots plotting $\hat{L}(t) - t$ (y-axis) against t (x-axis) where solid lines are $\hat{L}(t) - t$ curve of the data set and dashed lines are envelopes provided by the Monte Carlo Simulation (bottom).

Ripley's K function based CCDs successfully locate the true clusters of data sets with 3 or 14 clusters where each cluster is either uniformly distributed inside a unit box or normally distributed.

Algorithm 10 works well against data sets with spherical shaped clusters. MDSs are equivalent to the possible cluster centers whose balls have considerably high cardinality. On the other hand, arbitrarily shaped clusters rarely have centers. These type of data sets are often characterized not by their centers but by their shape. Hence, we have to slightly revise Algorithm 10 to locate the arbitrarily shaped clusters. Instead of finding the (approximate) MDS $S(G_{MD})$ of the intersection graph G_{MD}, we find the number of connected components of G_{MD}. The covering balls are subsets of the hidden class supports where the union estimates the clusters we are looking for.

Algorithm 10 is an parameter free version of CCD clustering algorithm given in Algorithm 9; however, R-CCDs are much slower than KS-CCDs in computation.

Theorem 6.3.2.2. *Let $\mathcal{X} \subset \mathbb{R}^d$ be a data set with $n = |\mathcal{X}|$ observations, and let N*

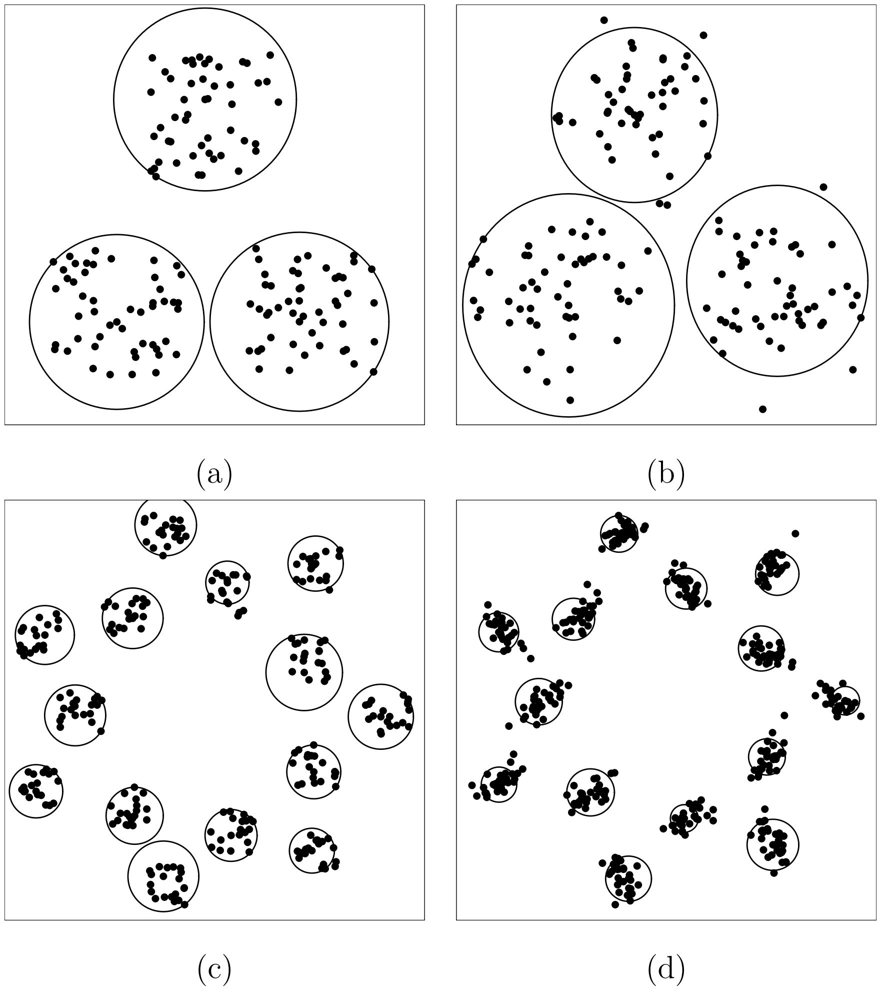


Figure 6.5: Four different settings with 3 or 14 classes, and the results of R-CCD algorithm. Each covering ball represents a cluster. (a) Three class setting with $F_1 = U([0,1]^2)$, $F_2 = U([3,4] \times [0,1])$ and $F_3 = U([0.5, 3.5] \times [3,4])$. (b) Three class setting with $F_1 = N(\mu_1, I_2)$, $F_2 = N(\mu_2, I_2)$ and $F_3 = N(\mu_3, I_2)$ where $\mu_1, \mu_2, \mu_3 \in \mathbb{R}^2$ and I_2 is the identity matrix of size 2. (c) 14 classes where each are uniformly distributed in some square regions from $\mathbb{R}^2$ (d) 14 classes where each are normally distributed in $\mathbb{R}^2$.

be the number of Monte Carlo replicants required for the confidence bands of $\hat{K}(t)$. Algorithm 10 partition the data set $\mathcal{X} \subset \mathbb{R}^d$ in $\mathcal{O}(n^4 + n^3 N + n^2 d)$ time

Proof: Following Theorem 6.3.1.1, the only difference between Algorithm 9 and Algorithm 10 is how $r(x)$ is computed for each point $x \in \mathcal{X}$. The distance matrix is found in $\mathcal{O}(n^2 d)$ time. For sorted distances from x to all other points, we calculate the $\hat{K}(t)$ value of points inside each window $B(x, r)$. This takes at worst $\mathcal{O}(n^3)$ for each x since, for each value $r \in D(x)$, $\hat{K}(t)$ is calculated in $\mathcal{O}(n^2)$ time. We calculate the upper envelope before the main loop and that takes at most $\mathcal{O}(n^3N)$ for N being the number of simulated data sets, and the $\hat{K}(t)$ is calculated for each the data set which takes $\mathcal{O}(n^3)$ time. ■

6.4 Monte Carlo Simulations and Experiments

In this section, we conduct series of Monte Carlo experiments to assess the performance of R-CCDs, and other clustering algorithms; KS-CCDs, fuzzy c-means (FCmeans), and pdfCluster (pdfC); on data sets with spherical clusters. In each trial, for each method, we record the relative frequency of finding the correct number of the clusters, and if the correct number was found, we also record the AUC measure. We use the area under curve (AUC) measure to evaluate the performance of the classifiers on the imbalanced data sets (López et al., 2013). AUC measure is often used on imbalanced real data classes (Huang and Ling, 2005). We report on the number of successes each method achieves in 100 trials, and also report the average AUC of successful trials (hence, we ignore the AUC of unsuccessful trials, which have AUC measure of 0 and thus no contribution to overall AUC). We aim to only measure the AUC of successful trails in order to assess in what degree the true clusters have been found. We investigate the performance of clustering methods on clustered data sets with fixed centers, and with random cluster centers (generated by a Strauss process), and on clustered data sets with noise. For each clustered data settings, we consider $\mathcal{K} = 2, 3, 5$ as the number of clusters, $n = 30, 50, 100, 200$ as the number of observations from each cluster, and $d = 2, 3, 5$ as the dimensionality of the data set.

R-CCDs are parameter free clustering methods, and hence do not require any parameter to be specified. On the other hand, FCmeans method require specifying the desired number of clusters $\hat{\mathcal{K}}$, and KS-CCD and pdfC methods require specifying the spatial intensity parameters δ and h, respectively. We incorporate a set of parameters for each method and report the parameters that perform the best. However, only for

Table 6.1: Centers of the clusters used in Monte Carlo simulations.

	Uniform			Normal		
	$d=2$	$d=3$	$d=5$	$d=2$	$d=3$	$d=5$
$\mathcal{K}=2$	$c_1=(0,0)$	$c_1=(0,0,0)$	$c_1=(0,0,0,0,0)$	$c_1=(0,0)$	$c_1=(0,0,0)$	$c_1=(0,0,0,0,0)$
	$c_2=(3,0)$	$c_2=(3,0,0)$	$c_2=(3,0,0,0,0)$	$c_2=(5,0)$	$c_2=(5,0,0)$	$c_2=(5,0,0,0,0)$
$\mathcal{K}=3$	$c_1=(0,0)$	$c_1=(0,0,0)$	$c_1=(0,0,0,0,0)$	$c_1=(0,0)$	$c_1=(0,0,0)$	$c_1=(0,0,0,0,0)$
	$c_2=(3,0)$	$c_2=(3,0,0)$	$c_2=(3,0,0,0,0)$	$c_2=(5,0)$	$c_2=(5,0,0)$	$c_2=(5,0,0,0,0)$
	$c_3=(1.5,2)$	$c_3=(1.5,2,2)$	$c_3=(1.5,2,2,0,0)$	$c_3=(2,4)$	$c_3=(2,3,3)$	$c_3=(2,3,3,0,0)$
$\mathcal{K}=5$	$c_1=(0,0)$	$c_1=(0,0,0)$	$c_1=(0,0,0,0,0)$	$c_1=(0,0)$	$c_1=(0,0,0)$	$c_1=(0,0,0,0,0)$
	$c_2=(3,0)$	$c_2=(3,0,0)$	$c_2=(3,0,0,0,0)$	$c_2=(5,0)$	$c_2=(5,0,0)$	$c_2=(5,0,0,0,0)$
	$c_3=(0,3)$	$c_3=(0,3,0)$	$c_3=(0,3,0,0,0)$	$c_3=(0,5)$	$c_3=(0,5,0)$	$c_3=(0,5,0,0,0)$
	$c_4=(3,3)$	$c_4=(3,3,0)$	$c_4=(3,3,0,0,0)$	$c_4=(5,5)$	$c_4=(5,5,0)$	$c_4=(5,5,0,0,0)$
	$c_5=(1.5,6)$	$c_5=(1.5,1.5,3)$	$c_5=(1.5,1.5,3,0,0)$	$c_5=(2.5,10)$	$c_5=(2.5,2.5,5)$	$c_5=(2.5,2.5,5,0,0)$

fuzzy c-means, we use the silhouette measure to choose a number of clusters that attain the best partitioning, hence we select the partitioning that achieves the maximum silhouette as the final partitioning. On the other hand, we report on the optimum parameters of KS-CCD and pdfC methods to investigate the relation between the intensity parameters and success rate of these methods. We choose $k = 2, \cdots, 10$ for fuzzy c-means, $\delta = 5, 5.05, 5.10, \cdots, 15$ for CCDs, and $h = 0.5, 0.55, 0.60, \cdots, 4$ for pdfC.

We start with simulation settings where simulated data sets have clusters with fixed centers. These centers, corresponding to each simulation setting, are given in Table 6.1. We conduct two separate experiments where in first, points of each cluster are drawn uniformly from a unit box in $\mathbb{R}^d$, and in second, points of each cluster are normally distributed in $\mathbb{R}^d$ with variance I_d which is an identity matrix of dimension $d = 2, 3, 5$. As in Table 6.1, for dimensions $d = 3, 5$, we do only change cluster centers in first two dimensions. Hence with increasing dimensionality, intra-cluster distance do not change drastically but inter-cluster distances increase, hence we expect that it would be harder to find the true clusters with increasing dimensionality.

In Table 6.2, we provide the relative frequency of success and the average AUC measures on simulated data sets of settings from Table 6.1. For uniformly distributed clusters in $d = 2, 3$, the R-CCDs achieves nearly % 90 success in finding the true number of clusters with almost 1.00 AUC. In these settings, low success rates are observed only on cases where $(\mathcal{K}, n) = (3, 30)$ and $(\mathcal{K}, n) = (5, 30)$. The low number of observations in each cluster establishes data sets with low spatial intensity. Therefore, the average inter-cluster distances are closer to intra-cluster distances that makes it harder to distinguish existing clusters, hence results in a deterioration of the performance of R-CCDs. However, especially FCmeans, all clustering methods achieve slightly more success rates than R-CCDs. For uniformly distributed clusters in $d = 2, 3$, pdfC performs relatively the worse with 52% success rate in $(\mathcal{K}, n) = (2, 30)$ but in all other cases, achieve nearly or above 90% success rate. The optimum parameters of KS-CCDs and pdfCs are nearly 5.00 and 1.00, respectively, although there is a slight decrease in both parameters with increasing number of clusters $\mathcal{K}$. For the settings with $d = 5$ however, R-CCDs perform much better than both KS-CCD and pdfCs, even though FCmeans still achieves 100% success rate in all settings including the ones with $d = 5$. Although FCmeans achieves the best performance among all methods, R-CCDs relatively perform well given that no parameters were to be introduced.

Almost in all settings, all clustering methods, especially R-CCDs, achieve nearly 1.00 AUC, but both KS-CCDs and pdfC have considerably less AUC than other methods for $d = 5$. R-CCDs and FCmeans perform relatively well although the performance of KS-CCDs and pdfC degrade with increasing dimensionality. For $d = 5$ and $\mathcal{K} = 3, 5$, because of the relationship between average inter-cluster distances and intra-cluster distances, we observe a decrease in the success rate. Other clustering methods show slightly better results than R-CCDs in $d = 2, 3$ but these methods required apriori parameters to achieve such high rates. Nevertheless, R-CCDs show relatively good performance despite the lack of any parameters. In settings with higher dimensions and higher number of clusters, the success rate of R-CCD decrease slightly but it still shows comparably great performance. When there are few number of observations and clusters being relatively close to each other, it is in fact harder to detect the clusters. The smaller the inter-cluster distances, the easier for CCDs to catch the true clusters.

Compared to all other clustering methods, R-CCD shows promising results, even though no assumptions on any parameter is made. The covering balls test the data set against CSR, it also works well with normally distributed clusters. With increasing number of observations, R-CCDs achieve nearly 80% success rate in finding the true normally distributed clusters for settings with $d = 2$. The reason for that is, once a covering ball grows closer to the center of one normally distributed cluster, it detects some locally homogeneous Poisson process, and keep growing until reaches some region of low density, i.e. a region with less intensity, which eventually make the test reject the null hypothesis. However, the greedy algorithm is able to detect these balls with high density which are more likely the centers of the normally distributed clusters. In summary, although R-CCDs test against locally CSR regions to find possible clusters, it works well for clusters with heterogeneous intensity which are unimodal (clusters with points distributed around a center).

The performance of a clustering method depends on both the intra-cluster and inter-cluster distances of several clusters in a data set. In some cases, although the data set is sampled from a mixed distribution, the samples from two or more clusters may be close to each other such that two or more clusters are estimated as a single cluster. We aim to investigate the performance of R-CCDs when clusters are forced to be distant. Therefore, we simulate data sets such that the parent distribution of the mixed distribution (the distribution of the cluster centers) is a Strauss Process. In this model, we generate cluster centers in a unit box of $\mathbb{R}^d$ that are at least $t \in \mathbb{R}_+$ apart from each other (by fixing $\beta = 0$ in the Strauss Process). We conduct two separate experiments where in first, each cluster is uniformly distributed in some square box from $\mathbb{R}^d$, where $t = (2 + \theta)r$ and $2r$ being the range of the uniform distribution in all dimensions. And in second, each cluster is normally distributed with $\Sigma = I_{d\times d}(r/5)$, where $t = (2 + \theta)r$. Here, we fix $\theta = 0.3$ and $r = 0.15$. Hence, we aim to investigate how well R-CCDs capture possible clusters compared to other methods when some

Table 6.2: Success rates and optimum parameters of all clustering methods where centers of clusters listed as in Table 6.1.

			Success Rate												param					
			$d=2$				$d=3$				$d=5$				$d=2$		$d=3$		$d=5$	
	$\mathcal{K}$	N	R-CCD	KS-CCD	FCmeans	pdfC	R-CCD	KS-CCD	FCmeans	pdfC	R-CCD	KS-CCD	FCmeans	pdfC	δ	h	δ	h	δ	h
Unif	2	30	81	100	100	75	99	96	100	52	97	70	100	45	5.20	1.30	5.05	1.00	5.00	0.50
	2	50	100	100	100	87	97	99	100	56	94	68	100	52	5.20	1.45	5.10	1.25	5.00	0.70
	2	100	98	100	100	95	92	100	100	70	84	81	100	59	5.20	1.65	5.10	1.35	5.00	0.90
	2	200	96	100	100	98	89	100	100	83	87	92	100	50	5.20	1.90	5.10	1.65	5.00	1.05
	3	30	20	61	99	90	96	91	100	94	95	30	100	20	5.15	0.95	5.05	1.10	5.35	0.80
	3	50	46	68	100	97	98	91	100	100	93	27	100	41	5.15	1.15	5.05	1.00	5.00	0.50
	3	100	99	78	100	97	98	95	100	100	40	27	100	53	5.15	1.15	5.05	1.05	5.00	0.95
	3	200	93	85	100	100	86	96	100	100	55	30	100	75	5.15	1.25	5.05	1.05	5.00	1.20
	5	30	83	99	100	97	91	81	100	90	90	20	100	1	5.05	0.85	5.00	0.90	5.30	0.50
	5	50	95	100	100	99	96	95	100	100	91	18	100	13	5.05	1.00	5.00	1.05	5.50	0.55
	5	100	81	100	100	100	92	100	100	100	57	16	100	37	5.05	0.90	5.00	0.95	5.85	0.65
	5	200	79	100	100	100	80	20	100	100	32	6	100	40	5.05	1.05	7.20	0.95	5.50	0.95
	$\mathcal{K}$	N	R-CCD	KS-CCD	FCmeans	pdfC	R-CCD	KS-CCD	FCmeans	pdfC	R-CCD	KS-CCD	FCmeans	pdfC	δ	h	δ	h	δ	h
Normal	2	30	95	97	100	96	100	88	100	63	94	26	100	34	5.05	1.10	5.00	0.95	5.00	0.50
	2	50	94	93	100	100	93	93	100	81	99	30	100	43	5.05	1.30	5.00	1.05	5.00	0.50
	2	100	88	91	100	100	85	88	100	95	73	27	100	55	5.10	1.25	5.00	1.15	5.10	0.60
	2	200	71	93	100	100	70	86	100	100	62	22	100	61	5.10	1.25	5.00	1.35	5.20	0.75
	3	30	48	79	100	100	75	56	100	98	70	18	100	9	5.05	0.85	5.00	0.95	5.00	0.60
	3	50	58	81	100	100	80	61	100	100	84	20	100	44	5.05	0.75	5.00	1.05	5.00	0.55
	3	100	68	82	100	100	71	47	100	100	57	22	100	66	5.05	0.75	5.00	0.80	5.10	0.70
	3	200	70	66	100	100	73	55	100	100	49	27	100	91	5.05	0.75	5.00	0.80	5.25	1.05
	5	30	46	75	100	99	56	19	100	99	67	2	100	1	5.00	0.75	5.20	0.75	5.00	0.70
	5	50	70	70	100	100	75	21	100	100	81	14	100	15	5.00	0.70	5.40	1.00	5.00	0.50
	5	100	73	76	100	0	57	18	100	100	54	22	100	41	5.00	0.00	5.75	0.90	5.05	0.50
	5	200	62	13	100	100	62	17	100	100	49	18	100	74	5.05	0.60	6.95	0.70	5.10	0.90

clusters are relatively close to each other.

The relative frequency of success have been given in Table 6.4. We omit reporting the AUC measures since all methods achieve almost 1.00 AUC. Compared to our previous simulation studies (see Table 6.2), some clusters are quite distant from each other. Hence, even in higher dimensions, R-CCDs achieve nearly and above 80% percent success rate in finding the true number of clusters both in uniform and normally clustered data sets. With increasing $\mathcal{K}$ however, clusters are less distant from each other, and thus, success rate degrades. Also, with decreasing n and increasing $\mathcal{K}$, the performance of R-CCDs deteriorate. FCmeans and pdfC methods achieve considerably more performance, however their performance also degrade with increasing dimensionality and number of clusters. Unlike the results in Table 6.2, FCmeans achieves nearly %60 success rate, however R-CCDs even exceed the success rates of FCmeans in some cases. Optimum parameters of both KS-CCDs and pdfCs, decrease

Table 6.3: AUC measures of all clustering methods given in Figure 6.2.

			$d = 2$				$d = 3$				$d = 5$			
	$\mathcal{K}$	N	R-CCD	KS-CCD	FCmeans	pdfC	R-CCD	KS-CCD	FCmeans	pdfC	R-CCD	KS-CCD	FCmeans	pdfC
	2	30	1.00	1.00	1.00	0.97	1.00	1.00	1.00	0.89	0.99	1.00	1.00	0.70
	2	50	0.99	1.00	1.00	0.98	1.00	1.00	1.00	0.95	0.99	1.00	1.00	0.72
	2	100	1.00	1.00	1.00	0.99	1.00	1.00	1.00	0.97	1.00	1.00	1.00	0.74
	2	200	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.98	1.00	1.00	1.00	0.84
	3	30	0.94	0.96	0.99	0.97	0.99	1.00	1.00	1.00	0.98	0.76	1.00	0.84
Uniform	3	50	0.93	0.96	0.99	0.98	1.00	1.00	1.00	1.00	0.99	1.00	1.00	0.63
	3	100	0.98	0.98	0.99	0.99	1.00	1.00	1.00	1.00	0.99	1.00	1.00	0.91
	3	200	0.99	0.98	0.99	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.98
	5	30	0.99	1.00	1.00	1.00	0.99	1.00	1.00	1.00	0.97	0.69	1.00	0.60
	5	50	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.71	1.00	0.59
	5	100	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.69	1.00	0.68
	5	200	1.00	1.00	1.00	1.00	1.00	0.67	1.00	1.00	1.00	0.61	1.00	0.00
	$\mathcal{K}$	N	R-CCD	KS-CCD	FCmeans	pdfC	R-CCD	KS-CCD	FCmeans	pdfC	R-CCD	KS-CCD	FCmeans	pdfC
	2	30	0.97	0.99	0.99	0.99	0.98	0.99	0.99	0.95	0.98	0.94	0.99	0.71
	2	50	0.98	0.99	1.00	0.99	0.98	0.99	0.99	0.97	0.98	0.97	0.99	0.73
	2	100	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.98	0.99	0.96	0.99	0.71
	2	200	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.96	0.99	0.83
	3	30	0.94	0.98	0.98	0.98	0.96	0.98	0.99	0.98	0.95	0.78	0.99	0.79
Normal	3	50	0.95	0.98	0.98	0.98	0.97	0.98	0.99	0.98	0.96	0.76	0.98	0.75
	3	100	0.97	0.98	0.98	0.98	0.98	0.98	0.99	0.98	0.98	0.77	0.99	0.89
	3	200	0.98	0.98	0.98	0.98	0.98	0.98	0.99	0.98	0.97	0.74	0.99	0.98
	5	30	0.94	0.98	0.99	0.99	0.95	0.67	0.99	0.98	0.95	0.50	0.99	0.61
	5	50	0.97	0.98	0.99	0.99	0.97	0.67	0.99	0.99	0.96	0.68	0.99	0.73
	5	100	0.98	0.99	0.99	0.00	0.98	0.68	0.99	0.99	0.97	0.62	0.99	0.70
	5	200	0.98	0.65	0.99	0.99	0.98	0.68	0.99	0.99	0.97	0.68	0.99	0.95

with number of clusters, but are usually around $\delta = 15$ and $h = 1$, respectively, for increasing number of dimensions. Although normally distributed clusters are located much harder than uniformly distributed clusters (since covering balls are tested against null hypothesis of data sets being CSR), R-CCDs locate the true clusters with considerably high success rate.

The settings, of which we illustrated the performance in Figure 6.4, were free of any noise in the data set. Even though a particular data set is noise free, with increasing dimensionality, some points in the domain could be viewed as noise due to the sparsity of the data set. Clustering in higher dimensions is often challenging, especially for density-based clustering methods. Our algorithms use the silhouette measure to clean

Table 6.4: Success rates and optimum parameters of all clustering methods where cluster centers are drawn from a Strauss process. AUC measures are all above 0,99, hence omitted.

			Success												param					
			$d=2$				$d=3$				$d=5$				$d=2$		$d=3$		$d=5$	
	$\mathcal{K}$	N	R-CCD	KS-CCD	FCmeans	pdfC	R-CCD	KS-CCD	FCmeans	pdfC	R-CCD	KS-CCD	FCmeans	pdfC	δ	h	δ	h	δ	h
Unif	2	30	95	100	100	96	99	83	100	91	99	82	100	65	11.95	1.30	14.80	1.20	13.50	0.85
	2	50	99	99	100	97	100	82	100	92	100	79	100	85	11.70	1.35	14.55	1.35	14.80	1.00
	2	100	99	100	100	99	100	68	100	98	100	79	100	96	12.55	1.75	14.25	1.50	12.75	1.25
	2	200	100	99	100	100	98	88	100	100	99	77	100	98	11.55	1.90	14.80	1.75	12.20	1.45
	3	30	85	94	91	95	95	78	85	95	92	50	88	51	8.70	0.85	15.00	0.95	14.70	0.75
	3	50	81	90	90	98	86	64	81	95	89	64	84	80	9.10	0.95	15.00	1.05	14.40	0.85
	3	100	88	93	86	99	84	75	80	98	91	60	86	97	8.40	1.10	14.90	1.10	14.85	1.10
	3	200	93	92	92	98	88	75	84	99	93	58	89	99	8.15	1.30	14.95	1.35	14.55	1.35
	5	30	44	77	90	95	85	82	84	89	98	36	90	13	5.45	0.65	14.00	0.75	14.80	0.70
	5	50	70	82	88	96	86	83	86	97	94	39	88	66	5.65	0.70	15.00	0.75	14.80	0.85
	5	100	86	87	93	99	82	88	90	100	90	31	90	95	5.70	0.75	14.80	0.80	14.90	0.85
	5	200	82	93	97	100	78	87	86	100	83	30	87	98	5.90	0.90	14.95	1.05	14.40	0.90
	$\mathcal{K}$	N	R-CCD	KS-CCD	FCmeans	pdfC	R-CCD	KS-CCD	FCmeans	pdfC	R-CCD	KS-CCD	FCmeans	pdfC	δ	h	δ	h	δ	h
Normal	2	30	96	100	100	100	98	82	100	97	97	75	100	69	12.65	1.05	14.80	0.95	14.70	0.50
	2	50	99	98	100	100	97	71	100	99	100	77	100	93	13.70	1.20	14.20	0.90	14.75	0.85
	2	100	93	98	100	100	97	77	100	99	99	74	100	99	12.15	1.45	14.25	1.05	14.15	0.95
	2	200	87	99	100	100	90	74	100	100	96	68	100	100	14.90	1.25	14.95	1.10	14.90	0.95
	3	30	75	84	81	99	73	67	79	97	79	56	73	63	9.70	0.85	14.90	0.85	14.85	0.85
	3	50	86	85	89	100	88	54	76	97	83	60	83	88	9.70	0.80	14.90	0.85	14.75	0.85
	3	100	71	95	91	100	73	58	70	100	84	42	78	99	10.05	0.90	15.00	0.90	14.15	1.20
	3	200	66	84	88	100	70	59	75	100	79	53	83	100	9.30	0.70	14.60	0.90	14.55	0.80
	5	30	48	73	92	99	70	59	76	89	80	29	70	29	6.05	0.60	14.80	0.60	14.70	0.55
	5	50	59	86	95	100	82	49	76	100	80	19	69	80	6.70	0.55	14.80	0.65	14.50	0.50
	5	100	59	79	95	100	61	43	80	100	72	18	69	98	7.35	0.60	15.00	0.75	14.40	0.80
	5	200	60	79	94	100	68	40	74	100	50	18	60	100	7.35	0.60	14.65	0.75	14.90	0.65

the set of MDSs from points which are centers of possible noise clusters. Silhouette measure is also very effective in discovering the true number of clusters when the data set is sparse. On the other hand, we simulate some data sets with clusters which are accompanied with some artificial noise. Hence, we repeat the experiments of Table 6.4 with just some background noise on mixed uniform distributions. We generate noisy points around the centers of each uniform clusters where these noise clusters are normally distributed with $\Sigma = I_{d\times d}(r/5)$, and we sample $\delta = n/5$ noisy points for each cluster.

The relative frequency of success and the average AUC measures have been given in Table 6.5. We do not report AUC measures since all methods achieve almost 1.00 AUC. Even with the existence of noise clusters, R-CCDs were able to achieve high success rates. Whether the data sets have noise or not, the performance of our

Table 6.5: Success rates and optimum parameters of all clustering methods where cluster centers are drawn from a Strauss process with some background noise around each cluster. AUC measures are all above 0,99, hence omitted.

		Success												param					
		$d=2$				$d=3$				$d=5$				$d=2$		$d=3$		$d=5$	
$\mathcal{K}$	N	R-CCD	KS-CCD	FCmeans	pdfC	R-CCD	KS-CCD	FCmeans	pdfC	R-CCD	KS-CCD	FCmeans	pdfC	δ	h	δ	h	δ	h
2	30	100	100	100	96	100	93	100	91	100	85	100	71	11.90	1.25	14.30	1.10	15.00	0.75
2	50	97	98	100	98	100	93	100	93	100	80	100	94	11.40	1.35	14.45	1.30	15.00	1.00
2	100	100	98	100	99	100	87	100	100	100	75	100	97	12.65	1.40	13.55	1.50	14.65	1.10
2	200	97	99	100	100	99	90	100	100	100	84	100	100	12.85	1.80	14.50	1.35	13.75	1.10
3	30	71	86	76	98	90	73	71	96	86	62	85	62	7.90	0.85	14.50	0.95	14.85	0.85
3	50	75	79	77	98	90	67	76	94	89	65	80	84	8.45	0.90	13.40	1.05	15.00	1.15
3	100	92	89	88	97	90	65	83	97	93	60	78	98	10.00	0.95	14.50	1.05	14.70	1.05
3	200	86	92	81	98	92	73	71	98	92	32	81	100	8.80	1.10	14.45	1.35	14.55	1.05
5	30	22	61	86	92	73	78	84	89	92	42	86	35	5.35	0.70	13.70	0.65	14.55	0.65
5	50	58	71	93	96	88	77	86	96	95	38	80	80	5.20	0.75	14.95	0.70	13.85	0.90
5	100	80	75	91	100	81	77	79	96	91	30	82	98	5.85	0.80	13.95	0.90	14.90	0.80
5	200	65	75	91	100	79	74	83	98	94	27	86	99	5.70	0.95	14.85	0.90	7.60	0.95

clustering algorithms degrade with increasing number of clusters and dimensionality. Hence, similar to other clustering algorithms, our methods are also affected by the existence of closely located clusters. Success rates of pdfC and KS-CCDs are considerably worse than R-CCDs and FCmeans, but R-CCDs have more success rate than the other methods in $d = 5$. However, R-CCDs still performs worse than others in the case $(\mathcal{K}, n, d) = (5, 30, 2)$ since the clusters are substantially close to each other. The success rate of all other clustering methods degrade with increasing number of clusters, however R-CCDs perform relatively well compared to all these methods. Also, the optimum parameters δ and h behave similar to those in results of Table 6.4.

6.5 Real Data Examples

In this section, we assess the performance of our CCD algorithms in real life data sets with arbitrarily shaped and non-arbitrarily shaped clusters (Alcalá-Fdez et al., 2011; Bache and Lichman, 2013). For data sets with considerably high variance, we apply dimension reduction to mitigate the effects of dimensionality. We use principal component analysis (PCA) to extract the principal components with high variance. We illustrate the number of clusters found and the corresponding AUC of all real data sets we consider in Table 6.6. Moreover, We illustrate some of the data sets and the clustering solutions of CCD algorithm in Figure 6.6. The ecoli2 and wine data sets are of dimensionality more than 2; however, we project these data sets to illustrate the clustering results. R-CCDs successfully locate the clusters of iris and wine data sets where in both are composed of three clusters with 0.89 and 0.86 AUC, respectively. The data set of ecoli2 inherently have two clusters but visually constitute three clusters. R-CCDs and FCmeans were able to find these three clusters, however for those optimum parameters of pdfC and KS-CCDs, the AUC is 0.66 which is low. For data sets with more clusters, R-CCDs had some errors in D31 data sets; that is, confused two clusters as one. Other than FCmeans in iris data set, all methods were able to find the true number of clusters in both iris and wine data sets. However, KS-CCDs were only able to achieve 0.56 AUC in detecting the clusters in iris data set. Moreover, the optimum values of δ for KS-CCD are drastically different for wine and iris data sets. For h value for pdfC, however, there seems to be no considerable difference. Other than FCmeans all methods were able to find 15 clusters in R15 data set. Yeast and glass data sets, in particular, constitute hard clustering problems where all methods produced different number of cluster, none of them being the true number. On the other hand, in aggregation data set, R-CCDs detect 4 clusters where in fact the true number is 4. However, the remaining three clusters are small clusters compared to the clusters detected by R-CCDs. In Figure 6.7, we illustrate the detected clusters before and after sweeped by the silhouette measure. R-CCDs were able to find the true number of clusters before applying the silhouette measure.

The data sets jain, spiral and, in particular, aggregation are data sets with arbitrarily shaped clusters that we use the Algorithm 10 of R-CCDs to do the partitioning (Chang and Yeung, 2008; Jain and Law, 2005). We compare the results of R-CCDs, KS-CCDs and DBSCAN on there data sets. In Figure 6.7, we illustrate the results of our R-CCD algorithm on these three spatial data sets, and we give the clustering results and optimum parameters of R-CCDs, KS-CCDs and DBSCAN in Table 6.7. In all data sets, unions of covering balls do not extend far from the supports of the

Table 6.6: Success rates and optimum parameters of all clustering methods for real data sets with spherical shaped clusters. PCd provides the number of principal components extracted for the associated data set. AUC measures are all above 0,99, hence ignored. (*) Without using the silhouette measure, the true number of clusters are 7 with 0.99 AUC. (**) Inherent number of spherical shaped clusters are 3 in ecoli2 data. (***) There are visually 3 possible clusters in the glass data

	Data				R-CCD		KS-CCD			FCmeans		pdfC		
	N	d	PCd	$\mathcal{K}$	$\hat{\mathcal{K}}$	AUC	$\hat{\mathcal{K}}$	AUC	δ	$\hat{\mathcal{K}}$	AUC	$\hat{\mathcal{K}}$	AUC	h
wine	178	13	4	3	3	0.86	3.00	0.95	0.020	3	0.96	3	0.89	0.75
iris	150	4	.	3	3	0.89	3.00	0.56	4.440	2	0.00	3	0.89	0.59
R15	600	2	.	15	15	0.99	15.00	0.99	20.000	14	0.00	15	0.99	0.50
D31	3100	2	.	31	30	0.00				31	0.94	31	0.99	0.50
aggregation	788	2	.	7*	4	0.00	7.00	0.99	0.007	3	0.00	7	0.99	0.61
yeast	1484	8	3	10	4	0.00	4	0.00	6.000	2	0.00	10	0.15	0.70
ecoli2	336	7	2	2**	3	0.00	2	0.66	0.400	3	0.00	2	0.69	1.70
glass	214	9	4	6***	5	0.00	4.00	0.00	0.002	2	0.00	6	0.29	0.09

data sets, even though their supports are not spherical shaped clusters. In jain data set, one cluster has less spatial intensity then the other (less average number of points in a unit area), and hence, its covering balls are much smaller compared to the other cluster. In spiral data set, however, points closer to the center of the data set is more compact than the points far from the center which produce bigger covering balls. The reason is that, within the same clusters, points have locally different intensity, but R-CCDs catch locally homogenuous sets of points, and hence CCDs are able to detect clusters with varying inter-cluster intensity. Aggregation data set has seven spherical shaped clusters but has five arbitrarily shaped clusters with two pairs of clusters weakly connected to each other. Algorithm 10 successfully locate all these clusters despite of the weak connection.

KS-CCD and DBSCAN algorithms were able to find the exact number of clusters in all data sets with arbitrarily shaped clusters. Optimum parameters are generally between [3,5] and [0.05,0.1] for KS-CCDs and DBSCAN algorithms, respectively. Although these parameters need to be initialize before the execution, and hence their

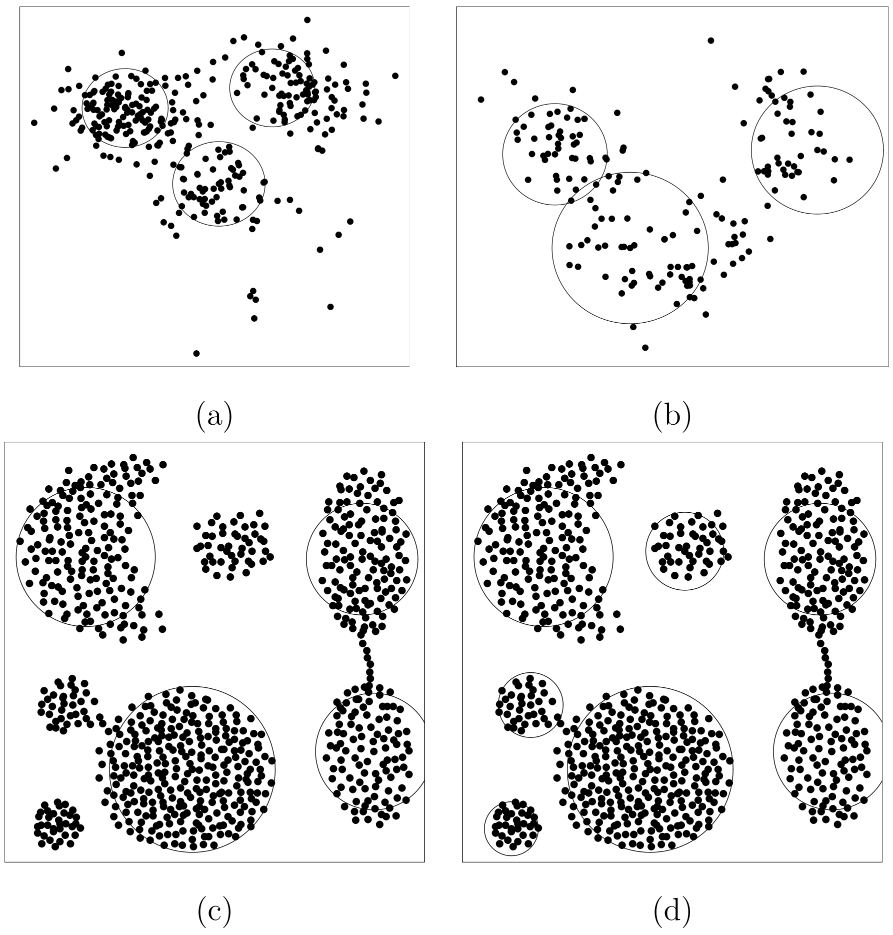


Figure 6.6: Illustrations of R-CCD results on real data sets with spherical shaped clusters. (a) ecoli2 data set has inherently 2 clusters but visually constitute 3 spherical clusters. (b) wine data set has three clusters, and R-CCDs successfully locate them. (c) The clusters of the aggregation data set with silhouette measure, (d) the clusters without silhouette measure.

selection is critical. However, R-CCDs were able to find the true number of clusters with no parameters are given.

6.6 Conclusions and Discussion

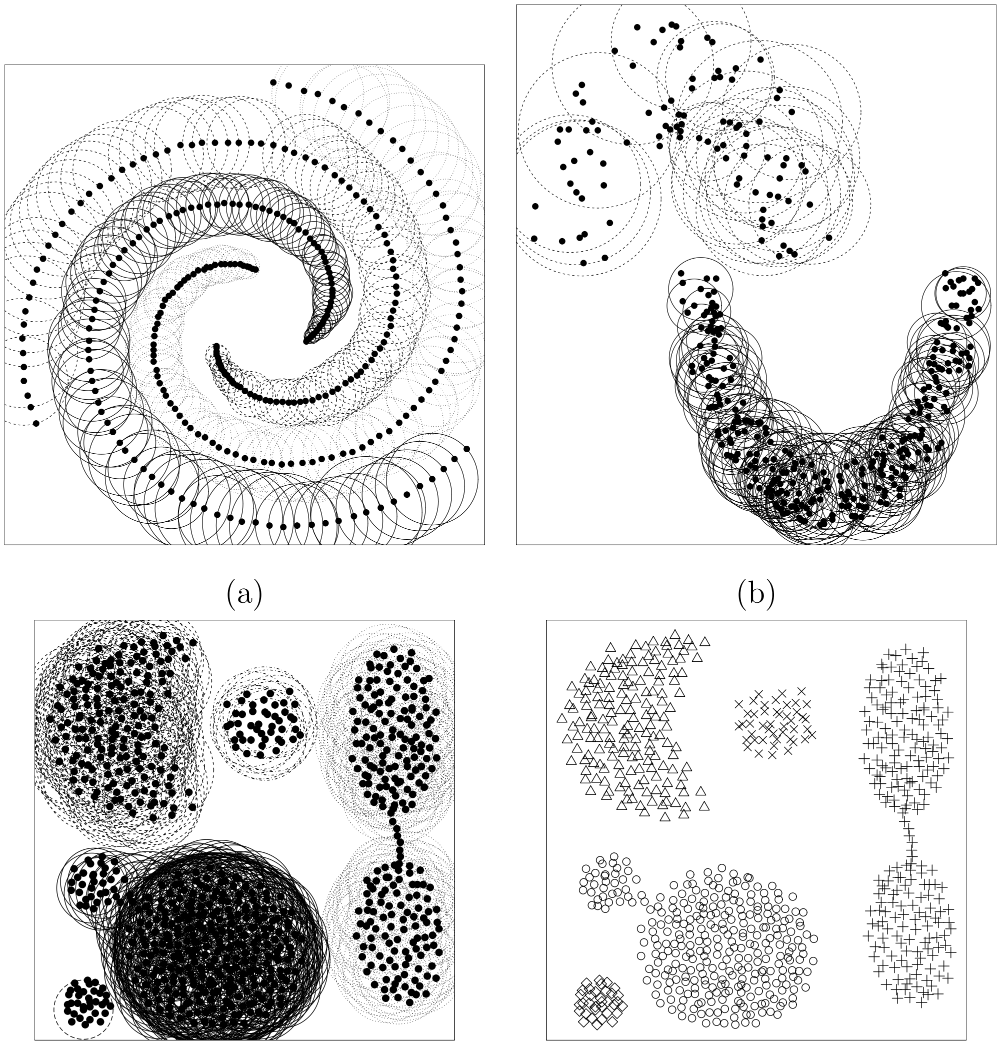
We use cluster catch digraphs (CCDs) to locate clusters and to find an optimal partitioning of data sets. CCDs are unsupervised adaptations of CCCDs, introduced to provide graph-theoretic solutions to the CCP. We use these covers to estimate the sup-

Table 6.7: Success rates and optimum parameters of all clustering methods for real data sets with arbitrarily shaped clusters. AUC measures are all above 0,99, hence ignored.

	Data			R-CCD		KS-CCD			DBSCAN		
	N	d	$\mathcal{K}$	$\hat{\mathcal{K}}$	AUC	$\hat{\mathcal{K}}$	AUC	δ	$\hat{\mathcal{K}}$	AUC	ϵ
aggregation	788	2	5	5	1.00	5	1.00	5.0	5	1.00	0.05
spiral	312	2	3	3	1.00	3	1.00	5.5	3	1.00	0.09
jain	600	2	2	2	1.00	2	1.00	3.5	2	1.00	0.09

port of the distribution of the data set, presumably a mixed distribution with a finite number of components, or clusters. We develop methods to increase the performance of recently introduced clustering methods based on CCDs, namely KS-CCDs, and use them as a framework for novel clustering algorithms. These methods are based on spatial pattern tests that incorporate Ripley's K function. We refer to these methods as R-CCDs. One important advantage of R-CCDs is that they find and estimate the number of clusters in the data set without any parameters. In the literature, density-based methods often estimate the number of clusters but these methods require prior parameters that assumes the density, or spatial intensity, around points of the data sets. DBSCAN and pdfCluster are examples of such methods. Both KS-CCDs and R-CCDs are hybrids of density- and graph-based methods, and in particular, R-CCDs estimate the spatial intensity of data sets unlike other commonly used density-based methods, including KS-CCDs.

Our estimates of the support are provided by a collection of balls that encapsulate the data set. These estimates constitute a *cover* given by the MDSs of CCDs. In KS-CCDs, the radius of each covering ball is the distance that maximizes a K-S based statistics. However, it require an intensity parameter of the spatial distribution of the data set if it were assumed to be drawn from a homogeneous Poisson process. In R-CCDs, however, the radius of each covering ball is chosen by a test based on the estimate $\hat{K}(t)$; that is, the radius is increased until the null hypothesis is rejected,



(a) (b) (c) (d)

Figure 6.7: Clustering results of Algorithm 10 on data sets with arbitrarily shaped clusters (a) *spiral* data sets with three spirally distributed clusters. (b) *jain* data set with cluster shaped as half rings (c) Covering balls that are produced. (d) Partitioning of the data.

indicating that the spatial distribution of a set of points inside the covering ball significantly deviates from complete spatial randomness (CSR). The rejection implies that a ball covers a region of the domain outside of the class support. More importantly, R-CCDs find an optimal partitioning without any parameters since $\hat{K}(t)$ is calculated by estimates of the spatial intensity parameter. We separately investigate data sets exhibiting spherical shaped and arbitrarily shaped clusters, and provide two types of algorithms for both KS-CCDs and R-CCDs, clustering these two types of data sets. We show that spherical shaped clusters are captured, or located, by MDSs of the intersection graphs, whereas we look for disconnected components of the same types of graphs to find arbitrarily shaped clusters.

By incorporating both the $\hat{K}(t)$ function and the silhouette measure, we develop clustering algorithms that are robust to noise, and work well against any data set with unimodal clusters. We demonstrate that CCD based methods successfully locate data sets with both uniformly and normally distributed clusters which may indicate that CCDs are able to detect the clusters of any data set whose clusters are spherical and have a single mode. Each covering ball is a representation of the local density; that is, the covering balls with high cardinality are more likely be the centers of normally distributed clusters. The greedy algorithm detects centers of clusters while the silhouette measure separates the low intensity regions of each individual normally distributed clusters from their centers. Moreover, we show that the CCDs are also non-sensitive to noise. CCD based clustering methods separate the true and noise clusters without assuming the number of noise clusters. We choose a minimum subset of the MDSs maximizing the silhouette measure, and we assume that the rest are noise clusters. These collections of noisy points do not substantially change the partitioning of the data set, and hence the average silhouette of the data set does not increase.

First, we conduct two extensive Monte Carlo simulations wherein simulated data sets have spherical shaped clusters. We assess the performance of both KS-CCDs and R-CCDs on these simulated data sets, and also on real data sets. We compare these methods with some partitional-based and density-based clustering methods; such as fuzzy c-means and pdfCluster. First, we incorporate fixed centers for the clusters, then, we generate cluster centers from a Strauss Process. On both simulation settings, around each cluster center, we either simulate clusters of uniformly distributed points of unit box or normally distributed points with some covariance matrix. The results show that our R-CCDs perform comparable to KS-CCDs and other methods. R-

CCDs achieve similar rates of correctly estimating the number of clusters in the data sets. R-CCDs also provide extremely high AUC if the number of clusters are correctly estimated. However, the performance of R-CCDs degrade on some cases when there are a high number of clusters, and the data set has high dimensionality. In these cases, average inter-cluster and intra-cluster distances gets closer, and hence, the clustering performance naturally decreases. Moreover, similar to other density-based methods, the performance also decreases with low number of points in each cluster. It is known that density-based methods are inefficient in locating clusters in data sets with low intensity. We also assess the performance of CCD clustering methods on real data sets with arbitrarily shaped clusters, and compare them with DBSCAN. R-CCDs successfully locate the clusters of data sets with high number of clusters, and also find arbitrarily shaped clusters with unequal spatial intensities; for example, the jain data set.

Although R-CCDs are appealing clustering algorithms, they are computationally intensive. We report on the computational complexity of both CCD methods, and show that KS-CCDs are cubic time and, in particular, R-CCDs are quartic time algorithms which makes them extremely slow compared to some other density-based methods. R-CCD based clustering algorithms are much slower in comparison to KS-CCDs, and it is due to the fact that R-CCDs are parameter free algorithms whereas KS-CCDs and other methods are not. Although the computation of envelopes before the execution of the main algorithm substantially decrease the computation time, the observed $\hat{K}(t)$ value of the data set should be computed for each point of the data set which makes the computation much slower. In that case, alternative spatial data analysis tools could be incorporated to decrease the computation time. However, our work proves the idea that parameter free CCD algorithms can be devised by estimating the spatial intensity. On the other hand, silhouette measure is sensitive to clusters with varying number of observations; that is, some clusters could be considered as noise even though they are compact and well separated from the rest of clusters, and hence a measure that robust the class imbalances could be employed instead of the silhouette measure. In our Monte Carlo experiments, the methods assuming the spatial intensity or the number of clusters may perform slightly better than R-CCDs. However, it may not always be easy to assume, or guess, such parameters.

Chapter 7

ENSEMBLE-BASED CLUSTERING WITH PROXIMITY CATCH DIGRAPHS

7.1 Introduction

Bagging, or bootstrap aggregating, is a method for strengthening the performance of learning methods. It is often the practice that a set of random samples of a data set is used to train a weak, i.e. unstable and non-optimal, learning method. Resulting set of learners constitute a meta-learner that a majority decision among the set of learners provide the final decision. A more general type of learning methods, referred to as *ensemble learning*, combine several learnings methods to create a new one which has significantly better performance than its constituents (Rokach, 2010). Bagging can also be used to aggregate the performance of clustering algorithms; that is, we can combine a set of different partitioning of a data set, i.e. *cluster ensembles*, to find an optimal partitioning of the data set (Strehl and Ghosh, 2002). We propose a bagging approach to increase the clustering performance of PCDs that are well defined only for labeled data sets. We offer algorithms to establish unsupervised adaptations of recently developed PCD families, and assess their performance in clustering.

In this chapter, we construct clustering algorithms with PE-PCDs and CS-PCDs. Clustering algorithms based on PCDs are also similar to density-based clustering methods. However, PE and CS proximity maps are ill-defined for unlabelled data sets; that is, both PE and CS maps depend on the existence of at least two classes in a data set. We choose random samples from the data set, and label these samples as members of the non-target class. We use each randomly labeled data set to construct PCDs and, later, to partition the data set. Finally, we use all these collections of partitionings to establish *cluster ensembles*. We show that, similar to CCDs, Ripley's K function can be used to establish the PE proximity regions whereas CS proximity regions do not require the Ripley's K function. We estimate the second-order moments of a set of points in a domain, and tell whether the points are clustered or fit to some sort of spatial distribution. Ripley (1977) introduced the K function to estimate the second-order moments and introduced tests for clustering of the data set. Our objective is to combine Ripley's K function with the PCDs to successfully find spherical shaped clusters.

7.2 Clustering with PCDs

Let $\mathcal{X} = \{X_1, X_2, \cdots, X_n\}$ be a set of $\mathbb{R}^d$-valued random variables with some distribution F and the support $s(F)$, and assume F is a finite mixture distributions. The objective is to find the number of these components (i.e. clusters), as well as which component X_i is more likely drawn from. We estimate the number of clusters $\mathcal{K}$, denoted as the estimand $\hat{\mathcal{K}}$, and offer algorithms to partition the data set into $\hat{\mathcal{K}}$ number of disjoint sets. To do so, we use the lower complexity covers to reveal the hidden clusters provided by the proximity regions associated with the MDSs.

We propose cluster ensembles with PCDs, whose each partitioning attained by a random set of the non-target class. Both PE-PCDs and CS-PCDs depend on the convex hull of $C_H(Z)$ for some subset $\mathcal{Z} \subset \mathcal{X}$ of the data set that we don't know if it exist. We choose a random subset $\mathcal{Z}$ to construct these PCDs. However, such an approach on clustering data sets would highly biased depending on the $\mathcal{Z}$. Hence, we repeat this for substantial amount of time, and use a collection of results to make a final decision on the partitioning of the data set. We show that, if the constituents of the ensemble are of considerably high amount, a majority of the constituents find the true number of clusters of data sets with spherical shaped clusters.

Once the data set is divided into the target class and non-target class, we construct the Delaunay tessellation of the convex hull $C_H(\mathcal{X}_{NT})$ for $\mathcal{Z} = \mathcal{X}_{NT}$. Proximity regions of each triangle is independent from the ones from other triangles. Hence, we first investigate the clustering with proximity regions in a single triangle, then move on to the algorithms for clustering the entire data set.

7.2.1 Clustering in Single Triangle Case

We use the Algorithm 4 and Algorithm 1 to find the MDSs of digraphs PE-PCDs and CS-PCDs. An advantage of PE proximity maps is that MDSs are computational tractable with respect to the size of $\mathcal{X}_{NT}$; however, they are less appealing compared to the CS proximity maps. CS maps satisfy more properties listed by Ceyhan (2010) and represent the domain influence of a point x in a triangle T much better than PE proximity maps. Given $\tau \in (0, 1]$, CS proximity maps are both central, i.e. the associated member of the dominating set is the center of its proximity region, and the proximity region is a proper subset of T. Therefore, we use Ripley's K function to determine the local extremum points of PE proximity maps, but drop their use for CS proximity maps. Hence, we show that CS-PCDs are more efficient in clustering data sets, but PE-PCDs are more accurate.

As in Chapter 2, barycentric coordinates of a set of points with respect to $T(\mathcal{Y})$ are used to characterize the set of *local extremum* points, where a subset of local extremum points constitute the minimum dominating set S_{MD}. However, we slightly change the definition of these local extremum points for clustering tasks. For $i = 1, 2, \cdots, d+1$, let $x_{[i]}$ denote the local extremum point associated with the vertex region $R_{M_C}(\mathsf{y}_i)$. We defined $x_{[i]}$ as point with the minimum barycentric coordinate with respect to y_i, i.e.

$$x_{[i]} := \underset{x \in \mathcal{Z} \cap R_M(\mathsf{y}_i)}{\operatorname{argmin}} \; w_{\mathfrak{S}}^{(i)}(x).$$

For PE proximity regions to reveal the clustering in data sets, we slightly change the definition so that $x_{[i]}$ corresponds to the minimum $w_{\mathfrak{S}}^{(i)}(x)$ with the condition that the points inside the associated proximity region, $\mathcal{N}_{PE}(x_{[i]}, r)$, does not reject the hypothesis of being CSR in $\mathcal{N}_{PE}(x_{[i]}, r)$.

Let $T_e = T_e(\mathsf{y}_1, \mathsf{y}_2, \mathsf{y}_3) \subset \mathbb{R}^2$ denote a equilateral triangle whose vertices are $y_1 = (0, 0)$, $\mathsf{y}_2 = (0, 1)$ and $\mathsf{y}_3 = (0.5, \sqrt{3})$. Now, let $\mathcal{N}(x) \subset T$ be an arbitrary triangle and let $\mathcal{G}$ be a group of affine transformations on $\mathbb{R}^2$, hence, there exist a transformation $g \in \mathcal{G}$ such that $T_e = g(\mathcal{N}(x))$. Note that $\mathcal{N}(x)$ is a simplical proximity region and constitutes a triangle in $\mathbb{R}^2$. By Theorem 2.3.0.1, the barycentric coordinates of a point z is equal to the coordinates of its image under g, i.e. $z' = g(z)$; that is, $\mathbf{w}_{\mathcal{N}(x)}(z) = \mathbf{w}_{T_e}(z')$. Therefore, for all proximity regions of arbitrary shape, we know the image of all points of $\mathcal{X}_T \cap \mathcal{N}(x)$ on T_e. We test whether points $g(\mathcal{X}_T \cap \mathcal{N}(x)) := \{g(x) : x \in \mathcal{X}_T \cap \mathcal{N}(x)\}$ in the transformed triangle T_e significantly deviate from CSR.

The digraph, induced by a single triangle, is independent from all other triangles. Hence, we investigate the clustering of points $\mathcal{X}_T \cap T$ by both PE-PCDs and CS-PCDs for an arbitrary triangle T. The results of a single triangle are given in Figure 7.1. We demonstrate the proximity regions of both PE-PCDs and CS-PCDs in three different cases. These are when $\mathcal{X}_T \cap T$ are uniformly distributed in T, when $\mathcal{X}_T \cap T$ clustered around vertices, and when $\mathcal{X}_T \cap T$ clustered around a single edge. In the first case, a single proximity region of both PE-PCD and CS-PCD catch all points in T. Such a triangle T is presumably inside the support of an hidden class. In the second case, sizes of the MDSs, i.e. the number of proximity regions, are either two or three. These triangles are most likely from corners of support of hidden classes and has less domain influence. In particular, the test rejects the hypothesis that any proximity region in one corner of the triangle is CSR, hence no local extremum point is provided on one corner of the triangle for PE-PCDs. Finally, in the third case, either 1 or 2 proximity regions are given, covering most of the dense areas those regions closer to the edge of the triangle. Observe that both proximity maps provide similar proximity regions, but CS proximity do not rely on spatial tests.

7.2.2 Random labelling and Clustering

We either use the PE-PCD or CS-PCD covers to find lower complexity covers of the data. Let $C_H(\mathcal{X})$ be the convex hull of the data set, and let $\mathcal{X}_B := \mathcal{X} \cap \partial(C_H(\mathcal{X}))$ be the boundary points of $C_H(\mathcal{X})$. We select a random sample of points from $\mathcal{X} \setminus \mathcal{X}_B$ to constitute a Delaunay tessellation of $C_H(\mathcal{X})$. Let $\mathcal{X}_S$ be this set of random samples,

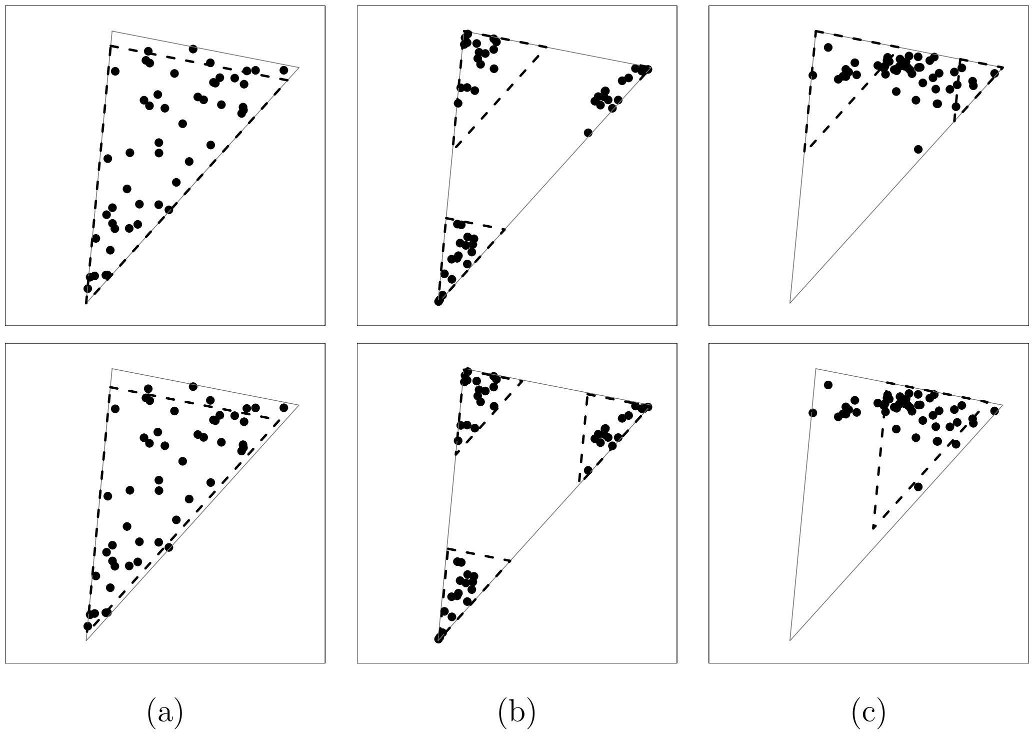


Figure 7.1: Clustering of a point set in a single triangle T. (a) Uniformly distributed points in T, (b) points clustered around vertices of T, and (c) points clustered around a single edge of T. (top) Proximity regions of PE-PCDs and (bottom) regions of CS-PCDs.

hence we denote the non-target class as $\mathcal{X}_{NT} = \mathcal{X}_S \cup \mathcal{X}_B$ and the target class as $\mathcal{X}_T = \mathcal{X} \setminus \mathcal{X}_{NT}$. Observe that $C_H(\mathcal{X}_{NT}) = C_H(\mathcal{X})$. Then, we establish either the PE-PCD or CS-PCD cover of $\mathcal{X}_T$ with respect to the Delaunay tesselation of $\mathcal{X}_{NT}$. In Figure 7.2, we illustrate the Delaunay tesselation, PE-PCD and CS-PCD covers to a data set with two uniformly distributed clusters.

7.2.3 Ensemble Clustering with PCDs

By randomly labelling points of the data set as non-target and target class, we constitute the PCD covers. However, by doing this several times, we let the ensemble of PCDs to vote for the true number of clusters. Each PCD cover partition the data set into different clusters. Hence, the majority decision of several partitionings of the data set constitute the final decision on the number of clusters. If a majority of ran-

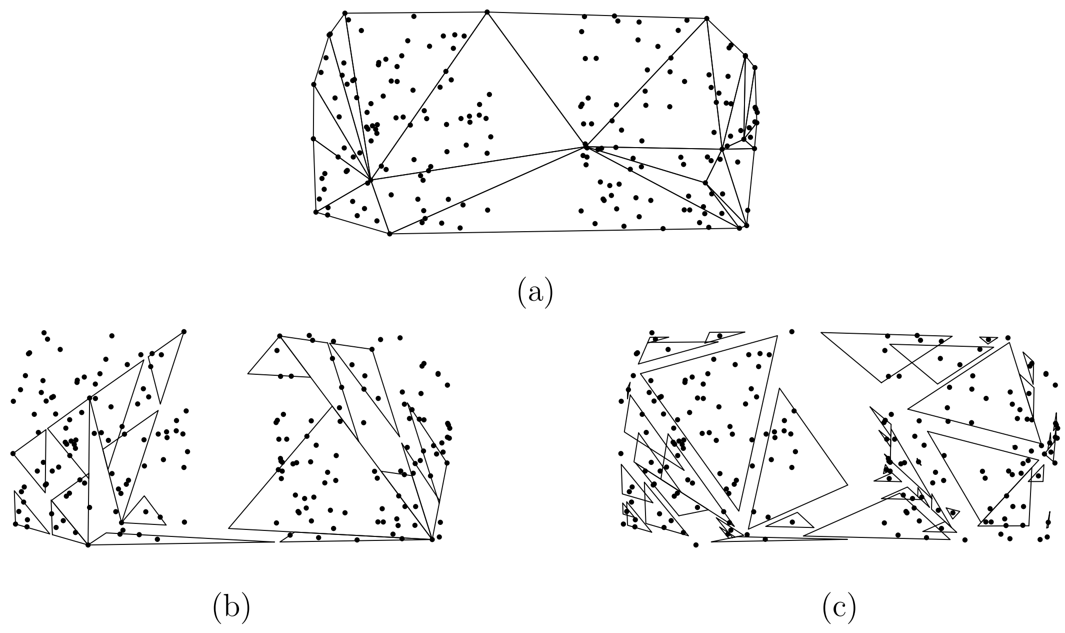


Figure 7.2: (a) The Delaunay tesselation of $\mathcal{X}_{NT} \subset \mathbb{R}^2$. The cover of (b) PE-PCD with proximity map $\mathcal{N}_{PE}(\cdot, r = 1.5)$ and (c) CS-PCD with proximity map $\mathcal{N}_{CS}(\cdot, \tau = 1)$.

dom runs suggests that partition the data set into $\hat{\mathcal{K}}$ clusters, then we conclude that there are $\hat{\mathcal{K}}$ clusters. Later, we choose the best partitioning among those conclude in $\hat{\mathcal{K}}$ cluster to choose the best clustering of the data set. We use an ensemble clustering approach to choose the clustering that resembles all the remaining clustering solutions the most.

Let N_e be the number of trials, i.e. the number of constituents, each associated with a different clustering solutions. We use the mutual information index introduced in Strehl and Ghosh (2002) to find an optimum partitioning of the data set amongst all partitionings in the ensemble. Now, $\mathcal{P}_i$ be the partitioning of the data set $\mathcal{X}$ associated with the i'th trial, and let $\mathscr{P} = \{\mathcal{P}_1, \mathcal{P}_2, \cdots, \mathcal{P}_{N_m}\}$. Thus, let (normalized) mutual information (NMI) between partitionings i and j be defined as

$$\Psi_{\mathscr{P}}(i,j) = \frac{\sum_{m_i=1}^{\hat{\mathcal{K}}_i} \sum_{m_j=1}^{\hat{\mathcal{K}}_j} n_{m_i,m_j} \log \frac{n \cdot n_{m_i,m_j}}{n_{m_i} n_{m_j}}}{\sqrt{\left(\sum_{m_i=1}^{\hat{\mathcal{K}}_i} n_{m_i} \log \frac{n_{m_i}}{n}\right)\left(\sum_{m_j=1}^{\hat{\mathcal{K}}_j} n_{m_j} \log \frac{n_{m_j}}{n}\right)}} \tag{7.1}$$

Here, $\hat{\mathcal{K}}_i$ is the estimated number of clusters in i'th trial, n_{m_i} is the number of

$\mathcal{K}=2$	$c_1=(0,0)$ $c_2=(4,0)$
$\mathcal{K}=3$	$c_1=(0,0)$ $c_2=(4,0)$ $c_3=(2,4)$
$\mathcal{K}=5$	$c_1=(0,0)$ $c_2=(3,0)$ $c_3=(0,3)$ $c_4=(3,3)$ $c_5=(1.5,6)$

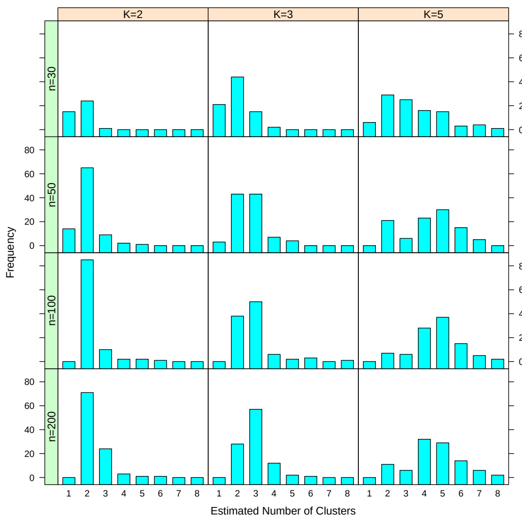
Table 7.1: Centers of the cluster used in Monte Carlo simulations for demonstrating the performance of (normalized) mutual information (NMI).

observations in the m_i'th cluster of $\mathcal{P}_i$, and n_{m_i,m_j} is the number of observations both in m_i'th and m_j'th clusters in $\mathcal{P}_i$ and $\mathcal{P}_j$, respectively. Let $\Psi_i = \sum_{j=1}^{N_m} \Psi_{\mathscr{P}}(i,j)$ be the average NMI of the partitioning $\mathcal{P}_i$. This strategy suggests that the partitioning who gives the maximum Ψ_i is the most similar to all other partitionings. We illustrate the PCD based clustering method in Algorithm 11.

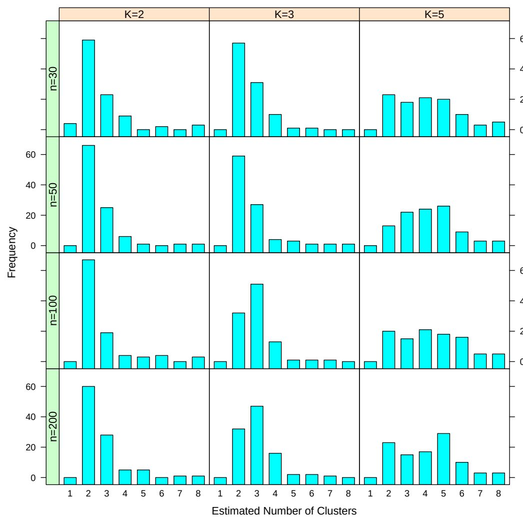
In Figure 7.3, we illustrate the number of found true clusters among N_e trials. For each clustered data settings, we consider $\mathcal{K}=2,3,5$ as the number of clusters, $n=30,50,100,200$ as the number of observations from each cluster, and $d=2$ as the dimensionality of the data set. The centers of these clusters are given in Table 7.1. Each cluster constitutes uniformly distributed points drawn from a unit box. In general, both PE-PCD and CS-PCD ensembles successfully vote for the true number of clusters. However, as the number of clusters $\mathcal{K}$ increases, it gets harder to conclude to a majority decision on the true number. Moreover, when $n=30,50$, both PCD families fail to find the true number, since the intensity of clusters are low due to the low number of observations in each cluster.

7.3 Monte Carlo Simulations and Experiments

In this section, we conduct series of Monte Carlo experiments to assess the performance of ensemble-based PCD clustering algorithms. We assess the performance of



(a)



(b)

Figure 7.3: Estimated number of clusters in $N_e = 100$ trials. (a) PE-PCD and (b) CS-PCD covers

two PCD based clustering algorithms, PE-PCDs with proximity maps $\mathcal{N}_{PE}(\cdot, r = 1.5)$ and CS-PCDs with proximity maps $\mathcal{N}_{CS}(\cdot, \tau = 1)$, and we use $N_e = 100$ number of ensembles for our clustering algorithms. In each trial, we record the relative frequency of finding the correct number of the clusters, and if the correct number was found, we also record the AUC measure. We report on the number of successes each method achieves in 100 trials, and also report the average AUC of successful trials (hence, we ignore the AUC of unsuccessful trials, which have AUC measure of 0 and thus no contribution to overall AUC). We aim to only measure the AUC of successful trails in order to assess in what degree the true clusters have been found. We investigate the performance of clustering methods on clustered data sets with fixed centers. For each clustered data settings, we consider $\mathcal{K} = 2, 3, 5$ as the number of clusters, $n = 30, 50, 100, 200$ as the number of observations from each cluster, and $d = 2$ as the dimensionality of the data set.

The cluster centers corresponding to each simulation setting are given in Table 7.2. We conduct two separate experiments where in first, points of each cluster are drawn

Algorithm 11 The PCD clustering algorithm

```
Input: The data set $\mathcal{X}$, the number of ensembling iterations $N_e$.
Output: The set of centers $S$ and their corresponding proximity regions
 1: $\mathcal{S} \leftarrow \emptyset$
 2: for all $i = 1, \cdots, N_e$ do
 3:     Let $\mathcal{X}_B = \mathcal{X} \cap \partial(C_H(\mathcal{X}))$
 4:     Let $\mathcal{X}_S$ be a random subset of $\mathcal{X} \setminus \mathcal{X}_B$
 5:     Let $\mathcal{X}_{NT} = \mathcal{X}_S \cup \mathcal{X}_B$
 6:     Let $\mathcal{X}_T = \mathcal{X} \setminus \mathcal{X}_{NT}$
 7:     Find $S_{MD}$ with Algorithm 4 for PE-PCD or with Algorithm 1 for CS-PCD
 8:     Find $S(G_{MD})$ with Algorithm 3 given $sc(v) = |N(v)|$
 9:     $S_I \leftarrow \emptyset$
10:     for all $s \in S(G_{MD})$ do
11:         $S_I \leftarrow S_I \cup \{s\}$
12:         Let $\mathcal{P}$ be the partitioning provided by $S_I$
13:         if $sil(\mathcal{P})$ decrease then
14:             break
15:         end if
16:     end for
17:     $\mathcal{S} \leftarrow \mathcal{S} \cup \{S_I\}$
18: end for
19: Set $S \in \mathcal{S}$ as the prototype set associated with $\mathcal{P}_i$ s.t. $\Psi_i = \text{argmax}_j \Psi_j$
```

uniformly from a unit box in $\mathbb{R}^2$, and in second, points of each cluster are normally distributed in $\mathbb{R}^2$ with variance I_2 which is an identity matrix of dimension $d = 2$. We have two settings in which centers of one setting are slightly further away from the centers in the other setting.

In Table 7.3, we provide the relative frequency of success and the average AUC measures of PE-PCD and CS-PCD clustering methods on simulated data sets with

Table 7.2: Centers of the cluster used in Monte Carlo simulations

	Setting 1		Setting 2	
	Uniform	Normal	Uniform	Normal
$\mathcal{K}=2$	$c_1=(0,0)$	$c_1=(0,0)$	$c_1=(0,0)$	$c_1=(0,0)$
	$c_2=(3,0)$	$c_2=(5,0)$	$c_2=(4,0)$	$c_2=(6,0)$
$\mathcal{K}=3$	$c_1=(0,0)$	$c_1=(0,0)$	$c_1=(0,0)$	$c_1=(0,0)$
	$c_2=(3,0)$	$c_2=(5,0)$	$c_2=(4,0)$	$c_2=(6,0)$
	$c_3=(1.5,2)$	$c_3=(2,4)$	$c_3=(2,4)$	$c_3=(3,6)$
$\mathcal{K}=5$	$c_1=(0,0)$	$c_1=(0,0)$	$c_1=(0,0)$	$c_1=(0,0)$
	$c_2=(3,0)$	$c_2=(5,0)$	$c_2=(4,0)$	$c_2=(6,0)$
	$c_3=(0,3)$	$c_3=(0,5)$	$c_3=(0,4)$	$c_3=(0,6)$
	$c_4=(3,3)$	$c_4=(5,5)$	$c_4=(4,4)$	$c_4=(6,6)$
	$c_5=(1.5,6)$	$c_5=(2.5,10)$	$c_5=(2,8)$	$c_5=(3,12)$

centers of clusters given in Table 7.2. For both normally and uniformly distributed clusters and for $\mathcal{K}=2$, the PE-PCD and CS-PCDs achieve extremely high performance with the case $(\mathcal{K},n)=(2,30)$ being an exception. When $n=30$, the density around each point is low, and hence clusters are hardly recognized. The low number of observations in each cluster establishes data sets with low spatial intensity. Therefore, the average inter-cluster distances are closer to intra-cluster distances which makes it harder to distinguish the existing clusters. However, the performance of both PCD methods are extremely low for $\mathcal{K}>2$, and in particular, when $\mathcal{K}=5$. Although high AUC values suggest found members of the MDS are approximately equal the cluster centers, both PE-PCD and CS-PCDs perform bad in finding the actual cluster centers. The intra-cluster distances are higher in the second simulation settings, hence we expect an increase in the clustering performance. PE-PCDs show good performance for $\mathcal{K}=3$ in Setting 2, but its performance extremely deteriorates when $\mathcal{K}=5$.

7.4 Conclusions and Discussion

We offer a bagging mechanism to boost the performance of clustering methods based on PCDs. We use PCDs to find lower complexity covers of data sets. Then, we use these covers to estimate the support of the distribution, presumably a mixed distribution with a finite number of components, or clusters. The PE-PCDs and CS-PCDs are defined for the target class (a subset of the data set) on the Delaunay tessellation of the non-target class (i.e. the class not of interest). However, in unsupervised learning schemes such as clustering, we don't know if classes exist. Hence, we choose random subsets of the data sets, and appoint them as non-target class to construct PCDs. Finally, to boost the clustering performance and to mitigate the bias resulted by randomly dividing the data set into target and non-target classes, we repeat this procedure for a substantial amount of time until a majority of constituents of a cluster ensemble vote for a final decision. We show that, if it is repeated considerably high amount of time, cluster ensembles of PCDs successfully find the true number of clusters.

We conduct two extensive Monte Carlo simulations wherein simulated data sets have spherical shaped clusters. We assess the performance of both PE-PCD and CS-PCD based clustering methods on these simulated data sets, whose centers are fixed. On both simulation settings, around each cluster center, we either simulate clusters of uniformly distributed points of unit box or normally distributed points with some covariance matrix. The results show that our PE-PCDs perform better than CS-PCDs in locating the true clusters. However, the performance of both PCD families degrade when there are high number of clusters. In these cases, average inter-cluster and intra-cluster distances gets closer, and hence the clustering performance naturally decreases. But more importantly, the performance of both PCD based clustering methods is much worse than the performance in CCD reported in Chapter 6. We use two sets of fixed centers. In second set of centers, each cluster is much further away from each other compared to the first set of centers. Hence, the performance is considerably increased.

Both PE-PCDs and CS-PCDs are ill-defined for unlabelled data sets, where we slightly boost the performance by bagging. Although PCD based clustering methods perform far worse than CCD based clustering methods, we are able to show that bagging can establish good performing clustering methods in collaboration with PCDs. However, bagging alone is not enough to compensate for the high complexity of the minimum dominating sets of PCDs. We have investigated the complexity of PE-PCDs and reported the bias resulted by the construction of PCDs with respect to the Delaunay tesselations. It has been mentioned that partitioning schemes with less complexity would mitigate the high bias of classifiers. Hence, as a future direction, our goal is to define appropriate proximity maps for new PCD families that would be appealing for clustering.

Table 7.3: Success rates and AUC of PE-PCD and CS-PCD clustering methods where centers of clusters listed as in Table 7.2. The AUC of those settings are omitted if they were to have no success in finding the true number of clusters, hence denoted as "-".

			PE-PCD				CS-PCD			
			Setting 1		Setting 2		Setting 1		Setting 2	
	$\mathcal{K}$	n	Success	AUC	Success	AUC	Success	AUC	Success	AUC
Uniform	2	30	32	0,957	32	0,967	100	0,992	100	0,997
	2	50	100	0,990	100	0,998	100	0,997	100	0,999
	2	100	100	0,999	100	1,000	100	0,997	100	0,999
	2	200	100	0,998	100	1,000	100	0,999	100	1,000
	3	30	0	-	0	-	8	0,992	13	0,993
	3	50	6	0,989	35	0,994	3	0,987	22	0,997
	3	100	48	0,983	93	0,996	16	0,981	29	0,997
	3	200	74	0,986	90	0,998	22	0,984	39	0,995
	5	30	0	-	7	0,991	0	-	8	0,955
	5	50	0	-	76	0,973	0	-	12	0,965
	5	100	0	-	74	0,972	0	-	14	0,958
	5	200	1	0,938	44	0,968	2	0,945	18	0,966
	$\mathcal{K}$	n	Success	AUC	Success	AUC	Success	AUC	Success	AUC
Normal	2	30	24	0,969	38	0,972	100	0,986	100	0,995
	2	50	100	0,989	100	0,997	100	0.989	100	0,997
	2	100	100	0.991	100	0.997	100	0.991	100	0.997
	2	200	100	0.992	100	0.998	100	0.992	100	0.997
	3	30	0	-	0	-	4	0.975	19	0.995
	3	50	10	0.973	27	0.993	17	0.978	34	0.991
	3	100	50	0.977	84	0.989	24	0.975	39	0.992
	3	200	39	0.974	87	0.989	21	0.976	38	0.987
	5	30	0	-	0	-	0	-	3	0.951
	5	50	0	-	24	0.985	1	0.900	5	0.963
	5	100	0	-	36	0.959	0	-	3	0.934
	5	200	0	-	13	0.951	0	-	1	0.963

Part IV

Conclusions

Chapter 8

SUMMARY AND DISCUSSION

In this thesis, we approach many popular challenges in machine learning literature with recently developed geometric digraph (or directed graph) families. Priebe et al. (2001) characterized the class cover problem with class cover catch digraphs (CCCDs) that pioneered an intriguing line of research that could be applicable to many aspects of the statistical learning. Later, Ceyhan (2005) generalized CCCDs by defining proximity catch digraphs (PCDs), introducing a new framework for arbitrarily defined proximity regions around the nodes of vertex-random directed graphs. One advantage of PCDs is that a proximity map can be defined to suit the needs of any learning practice. In this work, we offer tools and algorithms to apply PCDs in classification and clustering tasks.

We specifically investigated two PCD families associated with simplicial proximity maps (those maps that constitute simplices in the domain), namely proportional-edge and central-similarity proximity maps, that are defined for the class-labeled data sets with two classes. These are the target class (i.e. the class of interest) and the non-target class (i.e. the class of non-interest). A Delaunay tessellation in $\mathbb{R}^d$ is a collection of non-intersecting convex d-polytopes such that their union covers a region. Although this tessellation is likely to generate acute simplices in the domain, it does not partition (or applicable to) outside of the convex hull of the non-target class. An important contribution of this thesis is the extension of intervals in $\mathbb{R}$ with infinite boundaries to the higher dimensions, namely outer simplices, by using a similar framework defined by Deng and Zhu (1999). Each facet of the Delaunay tesselation is associated with an outer simplex. Such a region can be viewed as an infinite "drinking glass" with the facet being the bottom while top of the glass reaching infinity, similar to intervals in $\mathbb{R}$

with infinite endpoints. By introducing new proximity maps for the outer simplices, we successfully applied PCDs to classification whose proximity regions and minimum dominating sets have appealing properties in building discriminant regions.

We also characterized vertex and face regions of PCDs, which are auxiliary tools for defining the associated proximity regions, based on barycentric coordinates as this coordinate system is more convenient for computation in higher dimensions. The coordinate system is an important building block for PCDs; that is, we characterize the position of target class points inside simplices, define convex distance functions, and offer clustering algorithms with the invariant property of the barycentric coordinates under affine transformations.

Class imbalance problem is observed in many real life practices, especially in areas such as medicine and fraud detection. In a class-labeled data set, when the majority class (the class with an abundant number of observation) dwarfs the minority class (the class with few number of observations), a bias occurs towards the majority class, hindering the classification performance of many known learning methods, such as k nearest neighbor (k-NN) and support vector machines (SVM) classifiers. In Part II, we show that CCCDs and, in particular, PCDs are robust to the class imbalance problem. The minimum dominating sets (MDS) of these digraph families are equivalent to the prototype sets which are subsets of the data sets. Hence, we balance the number of observations from both classes by pruning the majority class. Two distinct families of CCCDs, namely pure CCCDs (P-CCCDs) and random walk CCCDs (RW-CCCDs) are robust to the imbalances between classes. However, MDSs of CCCDs (found by greedy algorithms) are computationally intractable. We use the property of MDSs of PCDs with proportional-edge proximity maps (PE-PCDs) being computationally tractable to build consistent classifiers based on PE-PCDs that are also robust to the class imbalances. We focus on both the class imbalance problem and the class overlapping problem. We found that the effects of class imbalance problem are more severe when it co-occurs with the class overlapping problem. We also define *local class imbalance* which usually occurs in the overlapping regions of both class supports.

In the literature, it is usually the practice to characterize the class imbalance with the ratio of number of observations from both classes. However, we investigate a case where local class imbalance exist despite of this ratio shows that there exist no imbalance. We show that both CCCDs and PCDs are robust to the local class imbalances.

Finally in Part III, we focus on the clustering of unlabeled data sets with PCDs. We investigate an unsupervised adaptation of the CCCDs, called cluster catch digraphs (CCDs). Spherical proximity regions of CCCDs are defined by the distance to the closest non-target class point. However, in a clustering task, we have to rely on spatial clustering tests to determine the radius of a covering ball (or a spherical proximity region) since classes do not exist. CCDs employ Kolmogorov-Smirnov based statistics to establish covers of the data set which also makes them members of density-based clustering methods. However, these statistics require a parameter that is proportional to the spatial intensity of a homogeneous Poisson process. We use a spatial data analysis test based on Ripley's K function that estimates the local spatial intensity. Hence, we build novel CCD algorithms, also called R-CCDs, that partition data sets with no assumptions on the value of any parameter. We assess the performance of R-CCDs against Kolmogorov-Smirnov based CCDs, i.e. KS-CCDs, a kernel-based clustering algorithm, pdfCluster, a fuzzy clustering alternative of the k-means algorithm, i.e. fuzzy c-means, and a density-based clustering methods, DBSCAN. R-CCDs successfully locate the true clusters of data sets with either spherical or arbitrarily shaped clusters, are robust to noise, and show comparable performance to those others methods we have listed, despite not requiring any parameter.

We also offer a bagging mechanism to boost the performance of PCD clustering methods. Both PE-PCDs and CS-PCDs are defined for the target class (a subset of the data set) on the Delaunay tessellation of the non-target class (i.e. the class not of interest). However, we don't know if classes exist, hence we choose random subsets of the data sets, and appoint them as non-target class to construct PCDs. Finally, to boost the clustering performance and to mitigate the bias resulted by randomly dividing the data set into target and non-target classes, we repeat this procedure for a substantial amount of time until a majority of constituents of a cluster ensemble vote for a final decision. We show that if this procedure is repeated sufficiently many times, PCD based clustering methods locate the actual clusters. On the other hand, the clustering performance drastically decrease with increasing dimensionality and with increasing number of clusters.

In this thesis, we showed that PCDs can be used to build efficient and consistent classifiers that also exhibit appealing properties like being robust to class imbalances or possessing computational tractable prototype sets. However, two PCD families with simplicial proximity maps, namely proportional-edge and central-similarity proximity maps, are constructed with respect to the non-target class whose complexity increases exponentially with dimensionality of the data set. Indeed, this leads to an overfitting of the data set in classification tasks and it also drastically decreases the clustering performance. However, we believe an alternative to the Delaunay tessellation, say for example a rectangular partitioning scheme, that produces less partitioning than a Delaunay tessellation would be more appealing for PCD covers. Such schemes could also have computationally tractable MDSs. Such classifiers and clustering methods are topics of ongoing research.

There is also the case that CCD and PCD based clustering algorithms are computationally intensive. Especially, Ripley's K function based CCDs, i.e. R-CCDs, are quartic time algorithms which makes them extremely slow compared to some other density-based methods. The observed $\hat{K}(t)$ value of the data set should be computed for each point of the data set which makes the computation much slower. In that case, alternative spatial data analysis tools could be incorporated to decrease the computation time.

BIBLIOGRAPHY

Akbani, R., Kwek, S., and Japkowicz, N. (2004). Applying support vector machines to imbalanced datasets. In *Proceedings of 15th European Conference on Machine Learning*, pages 39–50, Pisa, Italy.

Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., and Garc´a, S. (2011). Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Multiple-Valued Logic and Soft Computing*, 17(2-3):255–287.

Alpaydın, E. (1999). Combined 5× 2 cv F test for comparing supervised classification learning algorithms. *Neural Computation*, 11(8):1885–1892.

Angiulli, F. (2012). Prototype-based domain description for one-class classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(6):1131–1144.

Ankerst, M., Breunig, M. M., Kriegel, H.-P., and Sander, J. (1999). Optics: Ordering points to identify the clustering structure. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, pages 49–60, New York, NY, USA.

Arora, S. and Lund, C. (1996). *Approximation Algorithms for NP-Hard Problems*, chapter Hardness of Approximations. PWS Publishing, Boston, MA, USA.

Bache, K. and Lichman, M. (2013). UCI machine learning repository.

Baddeley, A., Rubak, E., and Turner, R. (2015). *Spatial Point Patterns: Methodology and Applications with R.* Chapman and Hall/CRC Press, London.

Ball, G. and Hall, D. (1965). Isodata: A novel method of data analysis and pattern classification. Technical report, Stanford Research Institute, Menlo Park.

Barandela, R., Sánchez, J. S., Garcıa, V., and Rangel, E. (2003). Strategies for learning in class imbalance problems. *Pattern Recognition*, 36(3):849–851.

Batista, G. E., Prati, R. C., and Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter*, 6(1):20–29.

Batista, G. E., Prati, R. C., and Monard, M. C. (2005). Balancing strategies and class overlapping. In *Proceedings of 6th International Symposium on Intelligent Data Analysis: Advances in Intelligent Data Analysis VI*, pages 24–35, Madrid, Spain.

Ben-David, S. and Haghtalab, N. (2014). Clustering in the presence of background noise. In Xing, E. P. and Jebara, T., editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 280–288, Bejing, China. PMLR.

Bereg, S., Cabello, S., Díaz-Báñez, J. M., Pérez-Lantero, P., Seara, C., and Ventura, I. (2012). The class cover problem with boxes. *Computational Geometry*, 45(7):294–304.

Bien, J. and Tibshirani, R. (2011). Prototype selection for interpretable classification. *The Annals of Applied Statistics*, 5(4):2403–2424.

Cannon, A. H. and Cowen, L. J. (2004). Approximation algorithms for the class cover problem. *Annals of Mathematics and Artificial Intelligence*, 40(3):215–223.

Ceyhan, E. (2005). *An investigation of proximity catch digraphs in Delaunay tessellations*. PhD thesis, Johns Hopkins University, Baltimore, MD, USA.

Ceyhan, E. (2010). Extension of one-dimensional proximity regions to higher dimensions. *Computational Geometry*, 43(9):721–748.

Ceyhan, E. and Priebe, C. E. (2005). The use of domination number of a random proximity catch digraph for testing spatial patterns of segregation and association. *Statistics & Probability Letters*, 73(1):37–50.

Ceyhan, E., Priebe, C. E., and Marchette, D. J. (2007). A new family of random graphs for testing spatial segregation. *Canadian Journal of Statistics*, 35(1):27–50.

Ceyhan, E., Priebe, C. E., and Wierman, J. C. (2006). Relative density of the random r-factor proximity catch digraph for testing spatial patterns of segregation and association. *Computational Statistics & Data Analysis*, 50(8):1925 – 1964.

Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Chang, H. and Yeung, D.-Y. (2008). Robust path-based spectral clustering. *Pattern Recognition*, 41(1):191–203.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16(1):321–357.

Chawla, N. V., Japkowicz, N., and Kotcz, A. (2004). Editorial: special issue on learning from imbalanced data sets. *ACM SIGKDD Explorations Newsletter*, 6(1):1–6.

Chvatal, V. (1979). A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235.

Cieslak, D. A. and Chawla, N. V. (2008). Learning decision trees for unbalanced data. In *Proceedings of the ECML PKDD 2008 Machine Learning and Knowledge Discovery in Databases: European Conference*, pages 241–256, Antwerp, Belgium.

Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27.

Das, S., Sen, M., Roy, A., and West, D. B. (1989). Interval digraphs: An analogue of interval graphs. *Journal of Graph Theory*, 13(2):189–202.

Deng, X. and Zhu, B. (1999). A randomized algorithm for the Voronoi diagram of line segments on coarse-grained multiprocessors. *Algorithmica*, 24(3-4):270–286.

Denil, M. and Trappenberg, T. (2010). Overlap versus imbalance. In *Advances in Artificial Intelligence*, pages 220–231. Springer.

DeVinney, J., Priebe, C., Marchette, D., and Socolinsky, D. (2002). Random walks and catch digraphs in classification. In *Proceedings of the 34th Symposium on the Interface, Volume 34: Computing Science and Statistics*, Montreal, Quebec, Canada.

DeVinney, J. G. (2003). *The class cover problem and its application in pattern recognition.* PhD thesis, Johns Hopkins University, Baltimore, MD, USA.

Devroye, L., Gyorfi, L., and Lugosi, G. (1996). *A Probabilistic Theory of Pattern Recognition.* Springer Verlag, New York.

Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923.

Domingos, P. (1999). Metacost: A general method for making classifiers cost-sensitive. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, pages 155–164, New York, NY, USA. ACM.

Drummond, C., Holte, R. C., et al. (2003). C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In *Workshop on Learning from Imbalanced Datasets (II)*, volume 11, Washington DC, USA.

Dunn, J. C. (1973). A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3(3):32–57.

Elkan, C. (2001). The foundations of cost-sensitive learning. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 973–978, Melbourne, Australia.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining*, pages 226–231, Portland, Oregon.

Eveland, C. K., Socolinsky, D. A., Priebe, C. E., and Marchette, D. J. (2005). A hierarchical methodology for class detection problems with skewed priors. *Journal of Classification*, 22(1):17–48.

Evelyn Fix, J. L. H. (1989). Discriminatory analysis. nonparametric discrimination: Consistency properties. *International Statistical Review / Revue Internationale de Statistique*, 57(3):238–247.

Fernández-Delgado, M., Cernadas, E., Barro, S., and Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems. *Journal of Machine Learning Research*, 15(1):3133–3181.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139.

Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., and Herrera, F. (2012). A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 42(4):463–484.

Gan, G., Ma, C., and Wu, J. (2007). *Data clustering : theory, algorithms, and applications.* ASA-SIAM series on statistics and applied probability. Philadelphia, Pa. SIAM, Society for Industrial and Applied Mathematics Alexandria, Va. American Statistical Association.

Gao, B. J., Ester, M., Xiong, H., Cai, J. Y., and Schulte, O. (2013). The minimum consistent subset cover problem: A minimization view of data mining. *IEEE Transactions on Knowledge and Data Engineering*, 25(3):690–703.

García, V., Mollineda, R. A., and Sánchez, J. S. (2008). On the k-nn performance in a challenging scenario of imbalance and overlapping. *Pattern Analysis and Applications*, 11(3-4):269–280.

Hammer, P., Liu, Y., Simeone, B., and Szedmk, S. (2004). Saturated systems of homogeneous boxes and the logical analysis of numerical data. *Discrete Applied Mathematics*, 144(12):103 – 109.

Han, H., Wang, W.-Y., and Mao, B.-H. (2005). Borderline-smote: A new oversampling method in imbalanced data sets learning. In *Proceedings of International Conference on Intelligence Computing, ICIC*, pages 878–887. Springer.

Hand, D. J. and Vinciotti, V. (2003). Choosing k for two-class nearest neighbour classifiers with unbalanced classes. *Pattern Recognition Letters*, 24(9):1555–1562.

He, H. and Garcia, E. A. (2009). Learning from imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on*, 21(9):1263–1284.

Huang, J. and Ling, C. X. (2005). Using AUC and accuracy in evaluating learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 17(3):299–310.

Jain, A. and Law, M. (2005). Data clustering: A users dilemma. *Pattern Recognition and Machine Intelligence*, pages 1–10.

Japkowicz, N. and Stephen, S. (2002). The class imbalance problem: A systematic study. *Intelligent Data Analysis*, 6(5):429–449.

Jaromczyk, J. W. and Toussaint, G. T. (1992). Relative neighborhood graphs and their relatives. *Proceedings of the IEEE*, 80(9):1502–1517.

Jarvis, R. A. and Patrick, E. A. (1973). Clustering using a similarity measure based on shared near neighbors. *IEEE Transactions on Computers*, 22(11):1025–1034.

Joachims, T. (1999). Making large-scale SVM learning practical. In Schölkopf, B., Burges, C., and Smola, A., editors, *Advances in Kernel Methods - Support Vector Learning*, pages 169–184. MIT Press, Cambridge, MA.

Julesz, B. (1975). Experiments in the visual perception of texture. *Scientific American*, 232:34–43.

Juszczak, P., Tax, D., and Duin, R. (2002). Feature scaling in support vector data description. In *Proceedings of 8th Annual Conference of the Advanced School for Computing and Imaging*, pages 95–102, Delft, Netherlands.

Karr, A. F. (1992). *Probability.* Springer-Verlag, New York, NY, USA, 1st edition.

Kotsiantis, S., Kanellopoulos, D., Pintelas, P., et al. (2006). Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering*, 30(1):25–36.

Kuhn, M. and Johnson, K. (2013). *Applied Predictive Modeling.* Springer, New York, USA.

Ling, C. X., Yang, Q., Wang, J., and Zhang, S. (2004). Decision trees with minimal costs. In *Proceedings of the 21th International Conference on Machine Learning*, page 69, Banff, Alberta, Canada.

Liu, X.-Y., Wu, J., and Zhou, Z.-H. (2009). Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 39(2):539–550.

Longadge, R. and Dongre, S. (2013). Class imbalance problem in data mining: Review. *International Journal of Computer Science and Network*, 2(1):83–87.

López, V., Fernández, A., García, S., Palade, V., and Herrera, F. (2013). An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences*, 250:113–141.

Mani, I. and Zhang, I. (2003). Knn approach to unbalanced data distributions: A case study involving information extraction. In *Proceedings of ICML'2003 Workshop on Learning from Imbalanced Datasets II*, Washington, DC, USA.

Marchette, D. (2010). Class cover catch digraphs. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(2):171–177.

Marchette, D. J. (2004). *Random Graphs for Statistical Pattern Recognition.* John Wiley and Sons, Inc., Hoboken, New Jersey, USA.

Marchette, D. J. (2013). *cccd: Class Cover Catch Digraphs.* R package version 1.04.

Mazurowski, M. A., Habas, P. A., Zurada, J. M., Lo, J. Y., Baker, J. A., and Tourassi, G. D. (2008). Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance. *Neural Networks*, 21(2):427–436.

Mehta, M., Rissanen, J., and Agrawal, R. (1995). Mdl-based decision tree pruning. In *Knowledge Discovery and Data Mining*, pages 216–221.

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., and Leisch, F. (2014). *e1071: Misc Functions of the Department of Statistics (e1071), TU Wien.* R package version 1.6-4.

Parekh, A. K. (1991). Analysis of a greedy heuristic for finding small dominating sets in graphs. *Information Processing Letters*, 39:237–240.

Pękalska, E., Duin, R. P., and Paclík, P. (2006). Prototype selection for dissimilarity-based classifiers. *Pattern Recognition*, 39(2):189 – 208.

Phua, C., Alahakoon, D., and Lee, V. (2004). Minority report in fraud detection: Classification of skewed data. *ACM SIGKDD Explorations Newsletter*, 6(1):50–59.

Prati, R. C., Batista, G. E., and Monard, M. C. (2004). Class imbalances versus class overlapping: An analysis of a learning system behavior. In *Proceedings of 3rd Mexican International Conference on Artificial Intelligence*, pages 312–321, Mexico City, Mexico.

Priebe, C. E., DeVinney, J. G., and Marchette, D. J. (2001). On the distribution of the domination number for random class cover catch digraphs. *Statistics & Probability Letters*, 55(3):239–246.

Priebe, C. E., Marchette, D. J., DeVinney, J., and Socolinsky, D. (2003a). Classification using class cover catch digraphs. *Journal of Classification*, 20(1):3–23.

Priebe, C. E., Solka, J. L., Marchette, D. J., and Clark, B. T. (2003b). Class cover catch digraphs for latent class discovery in gene expression monitoring by dna microarrays. *Computational Statistics & Data Analysis*, 43(4):621–632.

R Core Team (2015). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Raskutti, B. and Kowalczyk, A. (2004). Extreme re-balancing for SVMs: a case study. *ACM SIGKDD Explorations Newsletter*, 6(1):60–69.

Ripley, B. D. (1976). The second-order analysis of stationary point processes. *Journal of applied probability*, 13(02):255–266.

Ripley, B. D. (1977). Modelling spatial patterns. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 172–212.

Ripley, B. D. (1979). Tests of randomness' for spatial point patterns. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 368–374.

Ripley, B. D. (2005). *Spatial statistics*, volume 575. John Wiley & Sons.

Rissanen, J. (1989). *Stochastic Complexity in Statistical Inquiry Theory*. World Scientific Publishing Co., Inc., River Edge, NJ, USA.

Rokach, L. (2010). Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1):1–39.

Sajana, T., Rani, C. S., and Narayana, K. (2016). A survey on clustering techniques for big data mining. *Indian Journal of Science and Technology*, 9(3).

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

Seidel, R. (1995). The upper bound theorem for polytopes: an easy proof of its asymptotic version. *Computational Geometry*, 5(2):115 – 116.

Serafini, P. (2014). Classifying negative and positive points by optimal box clustering. *Discrete Applied Mathematics*, 165:270 – 282.

Socolinsky, D. A., Neuheisel, J. D., Priebe, C. E., De Vinney, J., and Marchette, D. (2003). Fast face detection with a boosted CCCD classifier. In *Proceedings of the 35th Symposium on the Interface, Volume 34: Computing Science and Statistics*, Salt Lake City, Utah, USA.

Strehl, A. and Ghosh, J. (2002). Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec):583–617.

Streib, K. and Davis, J. W. (2011). Using ripley's k-function to improve graph-based clustering techniques. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2305–2312. IEEE.

Takigawa, I., Kudo, M., and Nakamura, A. (2009). Convex sets as prototypes for classifying patterns. *Engineering Applications of Artificial Intelligence*, 22(1):101–108.

Tang, Y., Zhang, Y.-Q., Chawla, N. V., and Krasser, S. (2009). SVMs modeling for highly imbalanced classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 39(1):281–288.

Tax, D. (2014). Ddtools, the data description toolbox for MATLAB. version 2.1.1.

Tax, D. M. and Duin, R. P. (2004). Support vector data description. *Machine Learning*, 54(1):45–66.

Thai-Nghe, N., Busche, A., and Schmidt-Thieme, L. (2009). Improving academic performance prediction by dealing with class imbalance. In *Proceedings of 19th Internation Conferenence on Intelligent Systems Design and Applications*, pages 878–883. IEEE.

Toussaint, G. T. (1980). The relative neighbourhood graph of a finite planar set. *Pattern recognition*, 12(4):261–268.

Toussaint, G. T. (2002). Proximity graphs for nearest neighbor decision rules: Recent progress. In *Proceedings of the 34th Symposium on the Interface*, volume 34, Montreal, Quebec, Canada.

Ungar, A. A. (2010). *Barycentric Calculus in Euclidean and Hyperbolic Geometry: A Comparative Introduction*. World Scientific Publishing Co. Pte. Ltd., Singapore.

Vazirani, V. V. (2001). *Approximation Algorithms*. Springer-Verlag New York, Inc., New York, NY, USA.

Wang, W., Xu, Z., Lu, W., and Zhang, X. (2003). Determination of the spread parameter in the Gaussian kernel for classification and regression. *Neurocomputing*, 55(34):643 – 663.

Warren, J. (1996). Barycentric coordinates for convex polytopes. *Advances in Computational Mathematics*, 6(1):97–108.

Watson, D. F. (1981). Computing the n-dimensional delaunay tessellation with application to voronoi polytopes. *The Computer Journal*, 24(2):167–172.

West, D. B. (2000). *Introduction to Graph Theory*. Prentice Hall, New Jersey, USA, 2 edition.

Wilson, D. L. (1972). Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-2(3):408–421.

Woźniak, M., Graña, M., and Corchado, E. (2014). A survey of multiple classifier systems as hybrid systems. *Information Fusion*, 16:3 – 17.

Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Philip, S. Y., et al. (2008). Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37.

Xiong, H., Wu, J., and Liu, L. (2010). Classification with class overlapping: A systematic study. In *Proceedings of the 1st International Conference on e-Business Intelligence*, Shanghai, China.

Xu, X., Ester, M., Kriegel, H. P., and Sander, J. (1998). A distribution-based clustering algorithm for mining in large spatial databases. In *Proceedings of the 14th International Conference on Data Engineering*, pages 324–331, Orlando, Florida, USA.

Zadrozny, B., Langford, J., and Abe, N. (2003). Cost-sensitive learning by cost-proportionate example weighting. In *Proceedings of 3rd IEEE International Conference on Data Mining*, pages 435–442, Melbourne, Florida, USA.