OPTIMIZATION BASED PREDICTIVE METHODS FOR LARGE SCALE DATA

PhD Dissertation

Emre ÇİMEN

Eskişehir, 2018

OPTIMIZATION BASED PREDICTIVE METHODS FOR LARGE SCALE DATA

Emre ÇİMEN

DOCTOR OF PHILOSOPHY DISSERTATION

Graduate School of Sciences
Department of Industrial Engineering
Supervisor: Assoc. Prof. Dr. Gürkan ÖZTÜRK

Eskişehir
Eskişehir Technical University
Graduate School of Sciences
November, 2018

This dissertation is supported in part by the Scientific Research Projects commission of Eskişehir Technical University under the doctor of philosophy dissertation grant 1603F122 and by the Scientific Research Projects commission of Anadolu University under the general purpose grant 1506F499.

FINAL APPROVAL FOR THESIS

This thesis titled "Optimization Based Predictive Methods for Large Scale Data" has been prepared and submitted by Emre ÇİMEN in partial fullfillment of the requirements in "Eskişehir Technical University Directive on Graduate Education and Examination" for the Degree of Doctor of Philosophy (PhD) in Industrial Engineering Department has been examined and approved on 02/11/2018.

Committee Members		**Title Name Surname**	**Signature**
Member (Supervisor)	:	Assoc. Prof. Dr. Gürkan ÖZTÜRK	....
Member	:	Prof. Dr. Ayhan DEMİRİZ	....
Member	:	Prof. Dr. Hakan ÇEVİKALP	....
Member	:	Prof. Dr. Nihal ERGİNEL	....
Member	:	Prof. Dr. Refail KASIMBEYLİ	....

Prof. Dr. Ersin YÜCEL
Director of Graduate School of Sciences

ABSTRACT

OPTIMIZATION BASED PREDICTIVE METHODS
FOR LARGE SCALE DATA

Emre ÇİMEN

Department of Industrial Engineering

Eskişehir Technical University, Graduate School of Sciences, November, 2018

Supervisor: Assoc. Prof. Dr. Gürkan ÖZTÜRK

The prediction has historically been a topic which is of great importance and will not lose interest in the future. The prediction was made with primitive methods in the past. However, with the introduction of large scale data to our lives, primitive methods have left its place to the machine learning algorithms. Prediction methods with machine learning are split into two sub-problems as classification and regression. In this thesis, three novel machine learning methods have been developed which target different problems that can work with large scale data. The proposed methods are mainly based on mathematical programming and optimization. The first method is "Incremental Conic Functions (ICF) Algorithm for Large Scale Classification Problems" which applies an efficient data reduction method to the data. Furthermore, it does not require to solve a linear programming (LP) problem in some cases. The second method is "One-Class Polyhedral Conic Functions (O-PCF) Algorithm for One-Class Classification." This method can classify data points and detect outliers when the data is only available from one class. The last method is developed for "clusterwise linear regression" when the data size is large. These methods are tested on real-life datasets and compared with the well-known methods in the literature. It is possible to apply these three methods to real-life problems because of the short training and test times.

Keywords: Prediction, Machine learning, Optimization, Classification, Regression.

ÖZET

BÜYÜK ÖLÇEKLİ VERİ İÇİN ENİYİLEME TEMELLİ TAHMİNLEYİCİ YÖNTEMLER

Emre ÇİMEN

Endüstri Mühendisliği Anabilim Dalı

Eskişehir Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Kasım, 2018

Danışman: Doç. Dr. Gürkan ÖZTÜRK

Tahminleme geçmişten beri büyük öneme sahip olan ve gelecekte de güncelliğini yitirmeyecek bir konudur. Geçmişte ilkel yöntemlerle yapılan tahminleme günümüzde büyük verinin hayatımıza girmesiyle yerini büyük boyutlu verileri kullanabilen makine öğrenmesi algoritmalarına bırakmıştır. Makine öğrenmesi ile tahminleme yöntemleri, sınıflandırma ve regresyon problemleri olarak ikiye ayrılır. Bu tez çalışmasında, büyük boyutlu veriler ile çalışabilecek farklı problemleri hedef alan üç makine öğrenmesi yöntemi geliştirilmiştir. Geliştirilen yöntemler matematiksel programlama ve eniyileme temellidir. İlk geliştirilen yöntem, veriye etkin eleme yöntemleri uygulayan "Büyük Boyutlu Sınıflandırma Problemleri çin Arttırımlı Konik Fonksiyonlar (AKF)" algoritmasıdır. Ayrıca bu yöntem bazı durumlarda doğrusal programlama (DP) problemi çözmeyi gerektirmez. İkinci yöntem, "Tek Sınıf Sınıflandırma için Tek Sınıf Çokyüzlü Konik Fonksiyonlar (T-ÇKF)" algoritmasıdır. Bu algoritma sadece bir sınıfa ait veri olduğunda bile sınıflandırma yapabilir ve aykırı noktaları belirleyebilir. Son yöntem, veri boyutu büyük olduğunda "kümeleme temelli doğrusal regresyon" problemi için geliştirilmiştir. Tüm yöntemler gerçek hayat veri kümeleri üzerinde test edilmiş ve yazındaki iyi bilinen yöntemler ile karşılaştırılmıştır. Geliştirilen yöntemlerin eğitim ve test zamanları kısa olduğu için, bu yöntemleri bir çok gerçek hayat problemine uygulamak mümkündür.

Anahtar Kelimeler: Tehminleme, Makine öğrenmesi, Eniyileme, Sınıflandırma, Regresyon.

ACKNOWLEDGMENT

First of all, I would like to express my special thanks to my supervisor Assoc. Prof. Dr. Gürkan ÖZTÜRK for supporting and guiding me during my PhD. His openness to new ideas and enthusiasm has kept my motivation at the highest level. Since my undergraduate degree, he does his best to help me to become a better academician. I would also like to thank my dissertation comitee members Prof. Dr. Ayhan DEMİRİZ and Prof. Dr. Hakan ÇEVİKALP. This dissertation is presented thanks to their partipication, feedback and support.

I owe my deep gratitude to Prof. Dr. Adil BAGİROV. He is always a role model to me. Apart from his great academic guidance, he has accepted me as a family member during my visit to Federation University Australia for a semester. We have worked together on large-scale clusterwise linear regression problems in this dissertation.

I would also like to extend my thanks to Prof. Dr. Ömer Nezih GEREK, not only for his contribution to the ICF algorithm in this dissertation but also for his support to my personal and academical development for long years.

Finally, I must express my very profound gratitude to my parents Nermin and Mecit who strive to raise me through their life, and my beloved wife Simge for all of her love and being my other half. I felt their presence even when they were thousands of miles away from me.

Emre ÇİMEN

02/11/2018

STATEMENT OF COMPLIANCE WITH ETHICAL PRINCIPLES AND RULES

....

Emre ÇİMEN

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF NOTATIONS AND ACRONYMS

NOTATIONS

x	: scalar
$\mathbf{x}$	: vector
$\mathbf{X}$	: matrix, data set
$\langle .,. \rangle$	: inner product
$\lvert\cdot\rvert$ or $\psi\{\cdot\}$	: cardinality of a set
$\lVert\cdot\rVert$	: associated norm

ACRONYMS

ANN	: Artificial Neural Networks
CART	: Classification And Regression Trees
CF	: Conic Functions
CLR	: Clusterwise Linear Regression
EM	: Expectation Maximization
ICF	: Incremental Conic Functions
LP	: Linear Programming
MAE	: Mean Absolute Error
NN	: Nearest Neighbors
O-SVM	: One-Class Support Vector Machines
O-PCF	: One-Class Polyhedral Conic Functions
PCF	: Polyhedral Conic Functions
QP	: Quadratic Programming
RBF	: Radial Basis Function
RF	: Random Forest
RMSE	: Root Mean Squared Error
SVDD	: Support Vector Domain Description
SVM	: Support Vector Machines

1. INTRODUCTION

It is very important to predict a future event or future value to make better decisions. The prediction should be accurate, timely and reliable to develop right strategies based on the predictions. Since the beginning of human history, people have tried to predict the future based on their observations. For example, accurate predictions relating to weather events, agriculture/harvest, population and army have provided great advantages to civilizations. The importance of prediction is exponentially increasing day by day and spreading more rapidly to all areas of life. Furthermore, nowadays the problem of decision making has become much more complex.

With the invention of the computer, data structures and sizes have changed. The field of data science/data analysis/data mining has seen rapid strides over the past two decades and has been studied extensively [1]. It would not be wrong to say that the most important components of the data science are mathematics, statistics and computer science. This area of research has enabled the development of machine learning algorithms that give computers the ability to predict. These research areas are very new to the history of mankind from a wide perspective. No matter how disturbing it is, there is a possibility that the machines will learn independently from the people, decide and even build their own civilizations.

Apple's Siri and Face ID, Google's Duplex and Lens, self driving cars of Volkswagen, Uber and Tesla, robots of Boston Dynamics, Amazon's Go and Prime Air services can be shown among the products which developed using the cutting edge technologies in the field of data science and machine learning.

In [2], machine learning is defined as a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty. The usability, scalability and computational implementation of machine learning algorithms are very important for computer scientists [1]. With concepts like the internet of things (IoT) and Industry 4.0, there are countless data generating sensors in our daily life. More than that, according to IDC Digital Universe Study [3], the data volumes are increasing and more data has been created in a past two years than in the entire previous history. More interestingly, there are as many bits of information in the digital universe as stars in our physical universe. Although computer technology has been evolving rapidly, conventional machine learning algorithms do not meet the needs of users when the size of the data is large/big. It is important for algorithms to solve problems at reasonable times with acceptable accuracies.

There is no single unified and clear definition of big data. It has various definitions. According to a generally accepted definition, in order to define a dataset as big data, it should satisfy some of these properties (5Vs of big data): volume, velocity, variety, veracity and value. Large datasets are created in big data problems. In this dissertation, we have developed methods that can work effectively on large datasets that can not be effectively solved by the existing methods. These proposed effective algorithms can be used with parallel programming and distributed computing to solve big data problems. Therefore, the concept of "large data" is used, avoiding the use of the concept of "big data".

Solution approaches of machine learning problems are generally based on optimization. Although there are a variety of approaches, it is usually attempted to minimize an error function which is related to the emprical error. In this dissertation, three new supervised methods have been developed for classification and regression problems where the data size is large. All the proposed methods are mainly based on mathematical programming and optimization. Another common point of the proposed methods is to use clustering in order to take its advantage on unevenly distributed data. Thus, the proposed methods can adapt to different local distributions. Local learning methods try to locally adjust its parameters to the training set in each area of the input space [4]. Adopting local distributions is important because real-life problems generally consist of various distributions. Local learning methods have been applied to various machine learning problems including feature selection, clustering and classification [5–7].

In Chapter 2., the brief descriptions of data analysis and machine learning algorithms that mentioned in the next sections are given. Different problem types are discussed. Preliminaries about some existing methods are also provided.

In Chapter 3., *Incremental Conic Functions Algorithm for Large Scale Classification Problems* is provided [8, 9]. In this chapter, in order to cope with classification problems involving large datasets, we propose a new mathematical programming based method by extending the clustering based polyhedral conic functions approach. Despite the high classification efficiency of polyhedral conic functions, the realization previously required a nested implementation of k-means and conic function generation, which has a computational load related to the number of data points. In the proposed algorithm, an efficient data reduction method is employed to the k-means phase prior to the conic function generation step. The new method not only improves the computational efficiency of the successful conic function classifier, but also helps to avoid model over-fitting by giving fewer (but more representative) conic functions.

In Chapter 4., *One-Class Polyhedral Conic Functions (O-PCF) Algorithm for One-Class Classification* is presented. One-class classification, or outlier detection, is of great importance when data can be properly obtained from only one target class. The problem has many applications in various areas when the outlier class, defined as the complementary set to the target class, is absent. In this chapter, we develop a novel one-class classification algorithm called the one-class polyhedral conic functions (O-PCF) algorithm. In this algorithm, the decision boundary for the target class is defined by PCFs' level sets. The level set of a PCF is a convex polyhedron; thus, only convex decision boundaries can be obtained with one PCF. However, the target class may have a non-convex structure. Thus, the O-PCF algorithm divides the target class into k clusters and obtains a PCF for each cluster. O-PCF constructs the final classifier as the minimum of k PCFs to generate non-convex separating surfaces. The performance of the O-PCF algorithm is presented in comparison with other methods in the literature. The test results lead us to conclude that the O-PCF algorithm outperforms the other methods in many cases.

In Chapter 5., the study *A Fast Algorithm For Clusterwise Linear Regression In Very Large Datasets* is given. This chapter presents a new algorithm for solving clusterwise linear regression (CLR) problems in very large datasets. This algorithm is designed using the nonsmooth optimization model of the CLR problem and an incremental approach. The incremental approach is used to formulate an auxiliary CLR problem to find starting points. Results from the previous iterations of the incremental algorithm are used to develop a procedure to reduce the number of candidate starting points. This procedure consists of two main steps. In the first step dense areas in the dataset are identified and data points close enough to each other are removed from the list of possible candidates. In the second step, we remove points whose regression errors are smaller than some given tolerance. Using results from the previous iteration of the incremental algorithm, we remove data points with small regression errors for solving the auxiliary CLR problem. Finally, all points with small regression errors are not considered for minimizing the overall fit function with a large number of linear functions. Using the results of numerical experiments with very large datasets, we demonstrate that the use of these four procedures allows one to significantly reduce CPU time without any significant loss of accuracy. The proposed algorithm is tested using, in particular, very large real world datasets and compared with the mainstream regression algorithms using their prediction ability.

In Chapter 6., the dissertation is concluded and the future line of related research is given.

2. REVIEW ON DATA ANALYSIS AND MACHINE LEARNING

Data analysis is a term that is responsible for developing models, explanations and proposing hypotheses. It uses analytical methods and involves a wide range of processes such as data cleaning, extracting, visualization and modeling. Data mining is a subset of data analysis which tries to discover hidden information in a raw data. Data mining is a pipeline which includes the data collection, preprocessing (cleaning, feature extraction, feature selection etc.) and learning phase. Machine learning predicts values by using the learned models. It is very closely related to the data mining and statistics fields. But it differs slightly in terms of its terminology and emphasis [2].

If one says that the roots of the machine learning depend on least squares method [10] and the Bayes' Theorem [11] in the 17^{th} century, it would be true. After two centuries, Fisher's discriminant analysis [12] broke fresh ground for classification area. Discoveries of Markov Chains [13], perceptron [14] and artificial neural networks, nearest neighbors method [15], random forest algorithm [16], Support Vector Machines (SVMs) [17] and deep neural networks are the milestones for machine learning.

Machine learning methods can be divided into two main types:

- Predictive Methods
- Descriptive Methods

Predictive and descriptive methods are also known as supervised and unsupervised learning, respectively. This dissertation is focused on predictive methods. Although these methods are generally thought as two main types of machine learning, semi-supervised learning and reinforcement learning are defined as other types. Additionally, prescriptive methods which are a relatively new area in data analysis in comparison to predictive and descriptive methods should be mentioned. Prescriptive methods deal with effects of future decisions and update the decisions in advance. In this way, future outcomes are taken into consideration. Prescriptive methods has various applications. Decisions related to inventory, production, supply chain design are some examples that can be optimized by using prescriptive methods.

There are various methods to solve machine learning problems. It is not possible to cover all methods in detail. Therefore, only a brief description of the machine learning methods which are used in the following sections, are given.

2.1. Predictive Methods

The aim of such a method is to learn a model which maps its inputs $\mathbf{x}$ to output y. The set used for learning the model is named training set and it consists of $\mathbf{x}$ and y pairs [2]. Hence, it can be said that the training is learning a model by a machine under supervision of a set (training set) of which outputs are known.

$$\mathbf{X} = \{(\mathbf{x}_i, y_i)\}, \qquad i = 1, \ldots, p \tag{2.1}$$

The set in Equation 2.1 has p observations and $\mathbf{x}_i \in R^n$. The n values in the vector $\mathbf{x}$ are called features or attributes. With respect to various problem types, $\mathbf{x}$ could correspond to an image, a signal, a text or a DNA series, etc.

The response value be either categorical or real-valued. If y_i is a categorical value from a finite set, the problem is defined as classification. On the other hand, if y_i is a real-value, the problem is defined as regression. In classification, the response values could be spam/not spam for a spam filter, the name of the person in a face image, arrhythmic/normal heartbeat for arrhythmia detection, etc. On the other hand, one can predict the price of a house, future sales of a factory or the volume of a molecule by a regression model.

2.1.1. Support vector machines

SVMs are introduced by Vapnik [17] for binary classification problems and this paper [17] has been a pioneer in many types of research in this area. Then, the original binary classification is extended to multi-class classification and regression problems. In the simplest version linear SVM, in order to separate classes, decision boundaries between two classes are defined with a hyperplane. This hyperplane is obtained by solving an optimization problem.

SVMs are based on a very intuitive expectation. The optimization problem in SVM tries to find a hyperplane by maximizing the separation margin. Thus, this hyperplane is named as maximum margin hyperplane. Support vectors can be defined as the data points which would change the maximum margin hyperplane if removed.

In order to explain the SVM concept, generally, it is started with explaining the linearly separable case. In **Figure 2.1.**, one can see a linear separable case. Support vectors which define the maximum margin hyperplane between the two classes are marked with squares. The linear separating hyperplane is formulated as in Equation 2.2:

$$\langle \mathbf{w}, \mathbf{x} \rangle + b = 0 \tag{2.2}$$

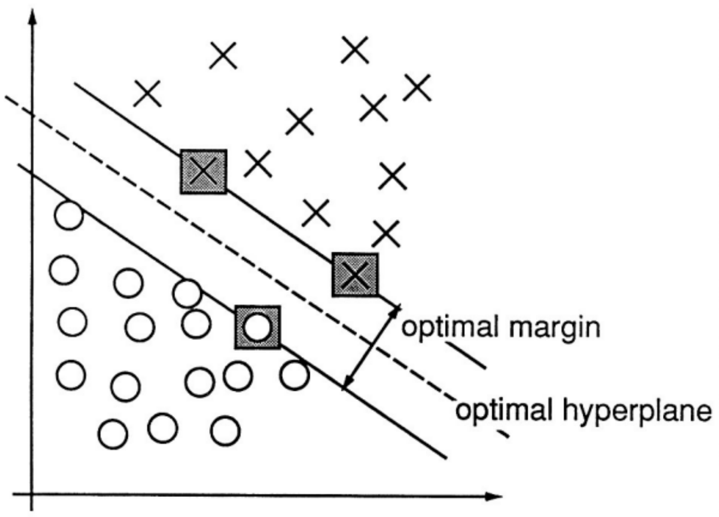

Figure 2.1. *A linear separable problem in 2-d space [17]*

Here, $\mathbf{w}, \mathbf{x} \in R^n$ and $\mathbf{w}$ is the vector which corresponds to the normal direction of the separating hyperplane. $\mathbf{x}$ is the data point and b is the bias which determines the distance of the hyperplane from the origin.

Assume that class labels $y_i \in \{+1, -1\}$, where $i = 1, \ldots, p$ in **Figure 2.1.** Because the example is linearly separable, all data points $\mathbf{x}_i$ with $y_i = +1$ should lie on the positive side of the hyperplane. On the contrary, all data points $\mathbf{x}_i$ with $y_i = -1$ should lie on the negative side of the hyperplane. Please see Equation 2.3 and Equation 2.4.

$$\langle \mathbf{w}, \mathbf{x}_i \rangle + b \geq 0, \qquad \forall i \in \{y_i = +1 |\, 1, \dots, p\} \tag{2.3}$$

$$\langle \mathbf{w}, \mathbf{x}_i \rangle + b < 0, \qquad \forall i \in \{y_i = -1 |\, 1, \dots\, p\} \tag{2.4}$$

Linear separability is the very rare case. Therefore, the linear SVM model is re-formulated as in Equation 2.5. This formulation is called soft margin SVM.

$$\min \quad \frac{1}{2} \|\mathbf{w}\| + C \sum_{i=1}^{p} \xi_i \tag{2.5}$$

subject to

$$\langle \mathbf{w}, \mathbf{x}_i \rangle + b \geq +1 - \xi_i, \qquad \forall i \in \{y_i = +1 |\, 1, \dots, p\} \tag{2.6}$$

$$\langle \mathbf{w}, \mathbf{x}_i \rangle + b < -1 + \xi_i, \qquad \forall i \in \{y_i = -1 |\, 1, \dots, p\} \tag{2.7}$$

$$\xi_i \geq 0, \qquad \forall i \in \{1, \dots, p\} \tag{2.8}$$

ξ_i is the violation of each margin constraint by $\mathbf{x}_i$. C is a user-defined parameter which penalizes the violations. Large values of the C minimizes the training errors.

In many cases, the linear separation boundary does not satisfy the needs of the user. For this reason, the need for obtaining non-linear separation boundaries has arisen. A non-linear decision boundary can be seen from **Figure 2.2.** In order to cope with this issue kernel functions are used. A kernel function is defined to be a real valued function of two arguments, $K(\mathbf{x}_i, \mathbf{x}_j) \in R$ [2]. $K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ where $\phi(.)$ maps the input vector into a higher dimensional space.

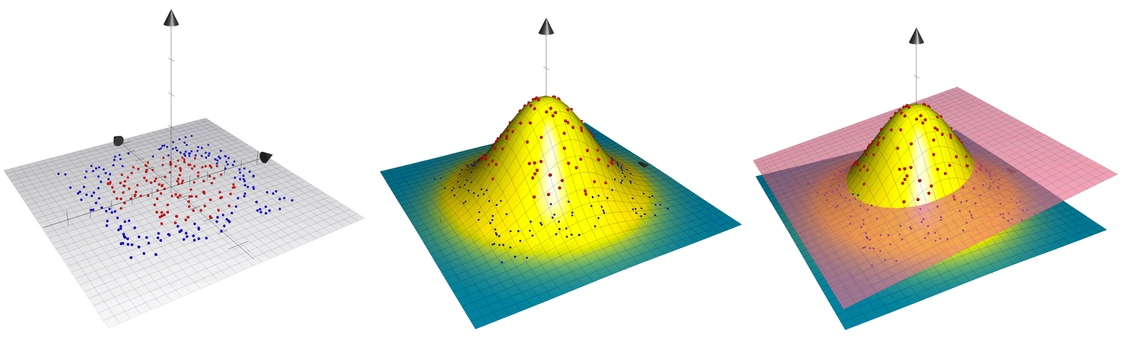

Figure 2.2. *Getting a non-linear decision boundary*

The kernel functions that are used commonly:

- Gaussian radial basis kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2})$

- Polynomial kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i.\mathbf{x}_j + c)^h$
- Sigmoid kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = tanh(\kappa \mathbf{x}_i.\mathbf{x}_j - \delta)$

It is very important to choose the right parameters of the kernel functions. This selection has a great impact on the success of the model. In non-linear case, the primal optimization problem becomes as in Equation 2.9.

$$\min \quad \frac{1}{2} \|\mathbf{w}\| + C \sum_{i=1}^{p} \xi_i \tag{2.9}$$

subject to

$$\langle \mathbf{w}, \phi(\mathbf{x}_i) \rangle + b \geq +1 - \xi_i, \qquad \forall i \in \{y_i = +1 | \, 1, \ldots, p\} \tag{2.10}$$

$$\langle \mathbf{w}, \phi(\mathbf{x}_i) \rangle + b < -1 + \xi_i, \qquad \forall i \in \{y_i = -1 | \, 1, \ldots, p\} \tag{2.11}$$

$$\xi_i \geq 0, \qquad \forall i \in \{1, \ldots, p\} \tag{2.12}$$

However, generally the dual form of the SVM problem is solved, due to the possible high dimensionalty of the vector $\mathbf{w}$. Dual form of the SVM formulation is given as follows [18]:

$$\min_{\alpha} \quad \frac{1}{2} \alpha^T \mathbf{Q} \alpha - \langle \mathbf{e}^T, \alpha \rangle \tag{2.13}$$

subject to

$$\langle \mathbf{y}^T, \alpha \rangle = 0 \tag{2.14}$$

$$0 \leq \alpha_i \leq C, \qquad \forall i \in \{1, \ldots, p\} \tag{2.15}$$

where $\mathbf{e} = [1, \ldots, 1]^T$ is the vector of ones, $\mathbf{Q}$ is an p by p positive semidefinite matrix, $\mathbf{Q}_{ij} \equiv y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$.

The classification formulation is extended by countless researches. However, some studies that define the various types of SVM have made a great impact. These studies include Bennett and Demiriz's semi-supervised SVM [19], Schölkopf's one-class SVM [20] and ν-SV regression [21], Tax and Duin's Support Vector Domain Description (SVDD) [22].

2.1.2. Nearest neighbors

The Nearest Neighbors (NN) algorithm was introduced by Cover and Hart in 1967 [15]. They showed the single NN rule has been shown to be admissibleamong the class of k-NN rules for the n-sample case for any n. It is simple and useful for both classification and regression. It is used widely as a benchmark for other well-known algorithms.

k-NN is an instance-based/lazy algorithm which means the algorithm does not priorly learn a model before testing. Therefore, this method has two disadvantages. First, in order to run the algorithm, the training set must held in memory (memory cost). This creates problems in large datasets. The second disadvantage is that the test time may be long because the algorithm does not learn a model in advance (computational cost). Test time is very important in some real-world applications such as health care, image processing or gesture recognition. The k-NN algorithm is non-parametric. This means that the algorithm does not make any assumptions about the distribution of the data.

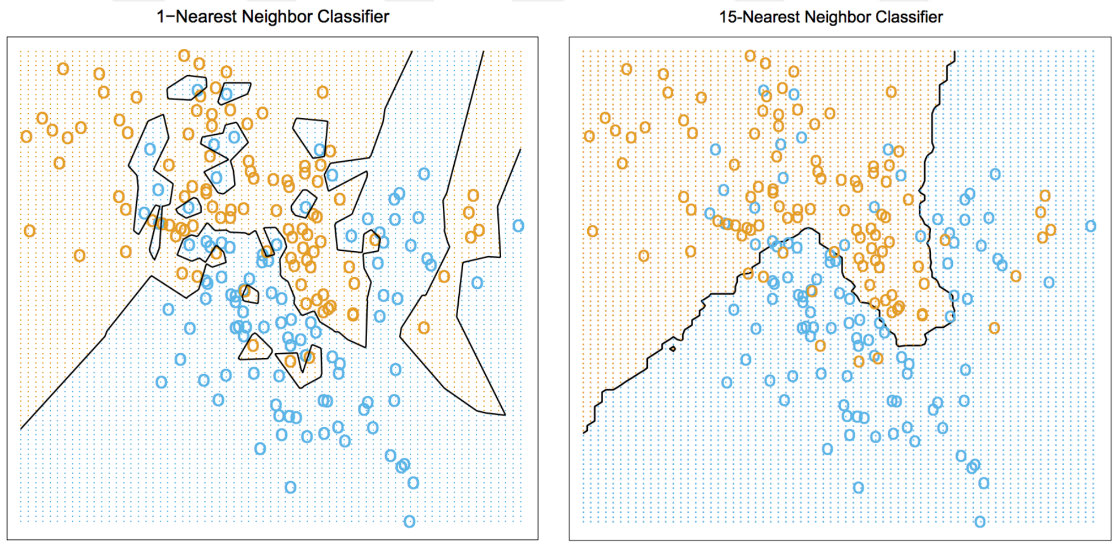

Figure 2.3. *The effect of the k on decision boundaries [23]*

In the test case, the distance between the test data point **x** and all data points in the training dataset is calculated. Then k points that are nearest to **x** is selected from the training data points. Finally, the class label is predicting by using class probabilities. The voting procedure or weighting depending on distance can be used in order to calculate probabilities.

The selection of the k affects the classification/regression accuracy. While a small k generates flexible decision boundaries, a large k generates smoother decision boundaries. Test time will be longer when the k is large. On the other hand, the algorithm will be more dependent on outliers when the k is small. Please see the **Figure 2.3.**

2.1.3. Classification and regression trees

Classification and regression trees (CART) are introduced by Breinman et al. in 1984 [24]. This model recursively partitions the input space and defines a different model for each local part of the input space. In other words, the method splits the feature space into rectangle regions, and then fit a simple model (like a mean of the region or a linear model) in each one. The major competitor of the CART is C4.5 [25] which is also a tree-based algorithm [23].

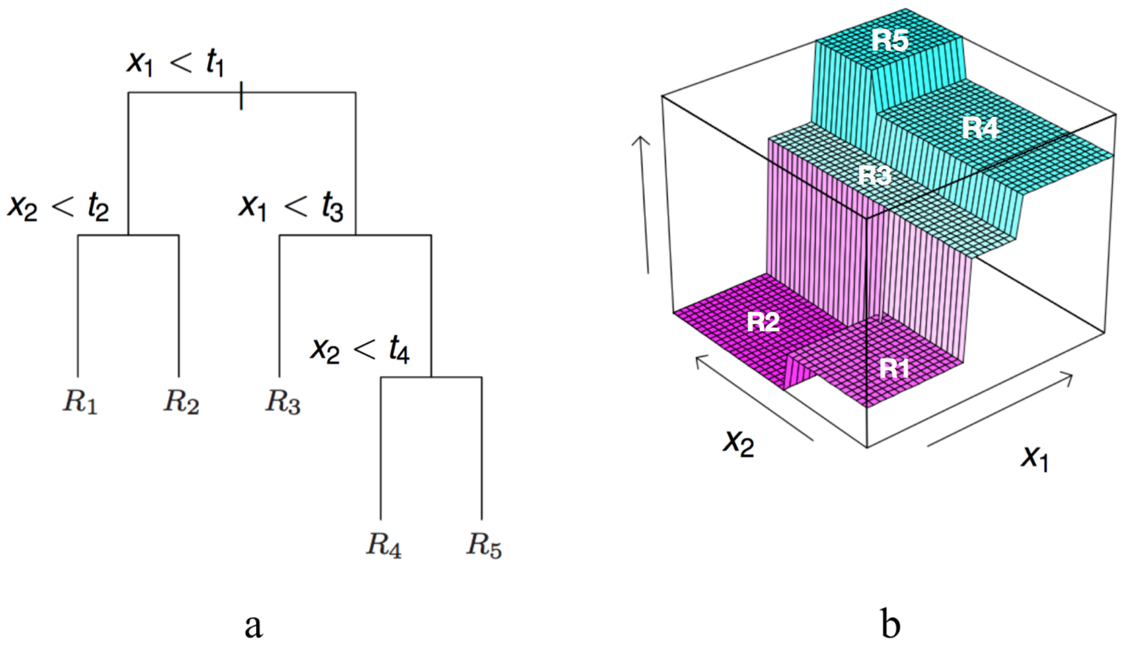

Figure 2.4. *(a)The tree corresponding to the partition in 2-d space (b) Plot of the prediction surface [23]*

In order to explain the model, a regression problem in *2-d* space is considered in **Figure 2.4.** with two features x_1 and x_2. One can see that the input space is split into 5 regions $R_1, R_2, \ldots, R_5$. If the input data point remains in a region R_m, the corresponding regression model is used to predict the output. Note that the splits are axis parallel.

The model can be written as in Equation 2.16:

$$f(\mathbf{x}) = E[y|x] = \sum_{m=1}^{M} w_m(\mathbf{x} \in R_m) = \sum_{m=1}^{M} w_m \phi(\mathbf{x}; \mathbf{v}_m) \tag{2.16}$$

where R_m is the m'th region, w_m is the mean response of the region, and $\mathbf{v}_m$ encodes the choice of the variable to split on, and the threshold value, on the path from the root to the m'th leaf [2].

2.1.4. Random forest

Random forest method developed by Breinman in 2001 [16]. It is an ensemble learning algorithm and used for classification and regression tasks. The algorithm creates multiple decision trees and combines them in order to get a robust final classifier. The algorithm is a generalization of the bagging method that is applied to decision trees. The main idea is to reduce the correlation between the different ensemble components by using randomized decision trees [1]. Random forest algorithm has been used by many researchers and applied various applications including Microsoft's Kinect for body pose estimation [2, 26].

If the model is trained with M different trees on different randomly selected subsets of the data, the ensemble is calculated as in Equation 2.17 [2]:

$$f(\mathbf{x}) = \sum_{m=1}^{M} \frac{1}{M} f_m(\mathbf{x}) \tag{2.17}$$

where f_m is the prediction of the m'th tree and $\mathbf{x}$ is the test data point.

An illustrative example of a random forest can be seen from **Figure 2.5.** In this play/do not play classification example, trees A and C votes for yes, and tree B votes for no. The dominant vote yes, therefore the child can play [27]. It should be noted that all the features are chosen randomly to create a decision tree.

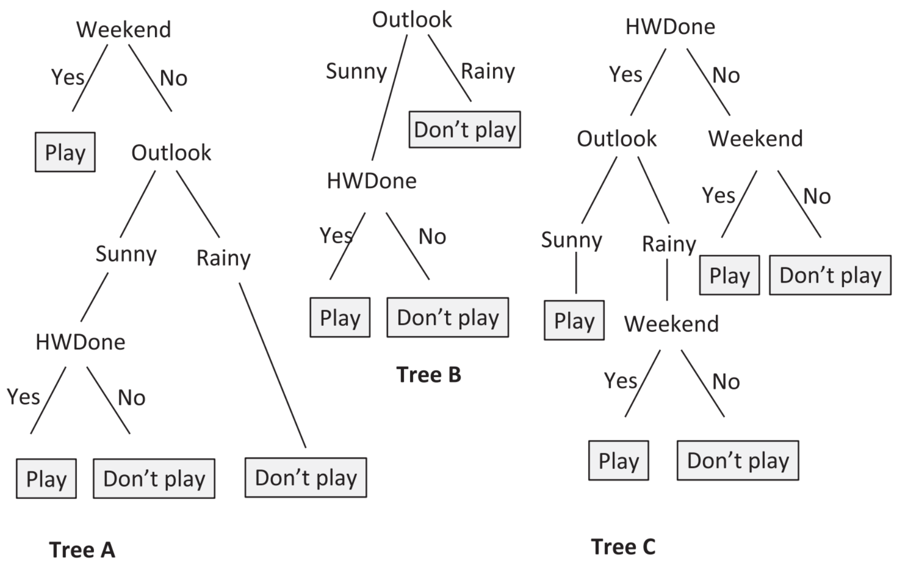

Figure 2.5. *A random forest with three trees [27]*

2.1.5. Artificial neural networks

The human nervous system consists of billions of cells named neurons. These specialized cells create a complex network structure. The neurons transmit messages to the following neuron through the links called synapses. Each neuron contributes to the decision to be made and training continues throughout life.

Artificial neural networks are one of the well-known machine learning models to solve classification and regression problems. These models are inspired by the biological nervous system. The neurons are the computation units which get the input data from other neurons, make computations, and send them to other neurons [1]. Basically, the computation function is a weighted sum of different inputs. The model is learned by updating the weights and the necessary data, external stimulus, is provided by the training set.

A wide variety of network structures exist. The simplest type is single layer perceptrons [14] that finds a separating hyperplane in the training data. The perceptron consists of two layers of nodes: input nodes and an output node. However, complex multilayer networks can be constructed in order to get robust classifiers/regressors. Layers between input and output layers are called hidden layers. The schema of a single layer and a multilayer neural networks can be seen in **Figure 2.6.**

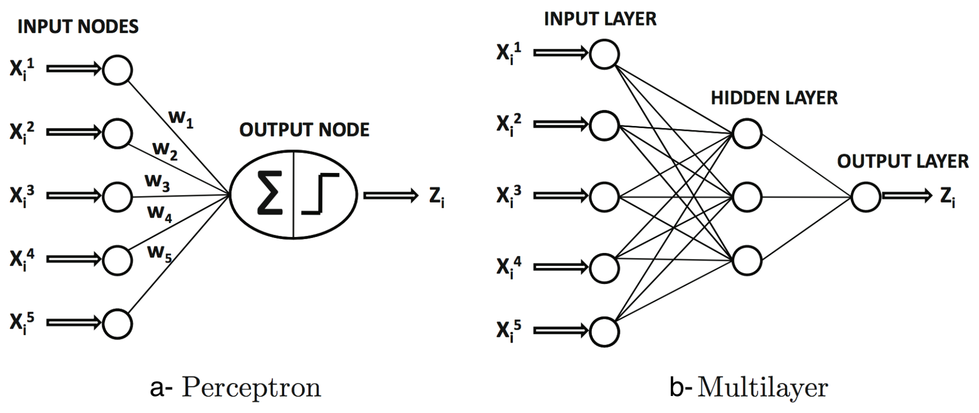

Figure 2.6. *(a)Single layer (b) Multilayer neural networks [1]*

In order to explain the output function of a perceptron, the decision function of a binary classification problem is considered in Equation 2.18:

$$z = sign\{\sum_{j=1}^{n} w_j x^j + b\} \tag{2.18}$$

where $\mathbf{w} = (w_1, \ldots, w_n)$ is the weight vector, $\mathbf{x} = (x^1, \ldots, x^n)$ is the input vector, b is the bias and $z \in \{-1, +1\}$ is the output classification label.

The following simplest perceptron case example is given to explain how weights are updated in Equation 2.19:

$$\mathbf{w}^{t+1} = \mathbf{w}^t + \eta(y - z)\mathbf{x} \tag{2.19}$$

where $\mathbf{w}^t$ is the weight vector in the t'th iteration, η is the learning rate, y is the actual output and z is the predicted **output**. The method starts with a random $\mathbf{w}^0$. Then, the training continues until $\mathbf{w}$ converges. In the multilayer case, the ground-truth outputs of the hidden layers are necessary. However, this information can not be known in the training. In order to cope with this problem, Rumelhart, Hinton and Williams presented the backpropagation algorithm in 1986 [28].

2.1.6. Polyhedral conic functions

In this subsection, we briefly describe the notions of PCFs and polyhedral conic separation. A more detailed description can be found in [29].

Let **A** and **B** be given disjoint sets in R^n containing m and p points, respectively:

$$\mathbf{A} = \{\mathbf{a}^1, \ldots, \mathbf{a}^m\}, \quad \mathbf{a}^i \in R^n, \quad i = 1, \ldots, m, \tag{2.20}$$

$$\mathbf{B} = \{\mathbf{b}^1, \ldots, \mathbf{b}^p\}, \quad \mathbf{b}^j \in R^n, \quad j = 1, \ldots, p. \tag{2.21}$$

PCFs construct a separation function for the sets **A** and **B** as following.

Definition 2.1. A function $g : R^n \rightarrow R$ is called *conic function* if its graph is a cone and all its level sets satisfy [29]:

$$S(\alpha) = \{\mathbf{x} \in R^n : g(\mathbf{x}) \leq \alpha\}, \tag{2.22}$$

for $\alpha \in R$, yielding, so called, convex sets.

Figure 2.7. shows a graph of a PCF and its level set. As shown in **Figure 2.7.**, the graph of a PCF is a cone, and its level set is a convex polyhedron.

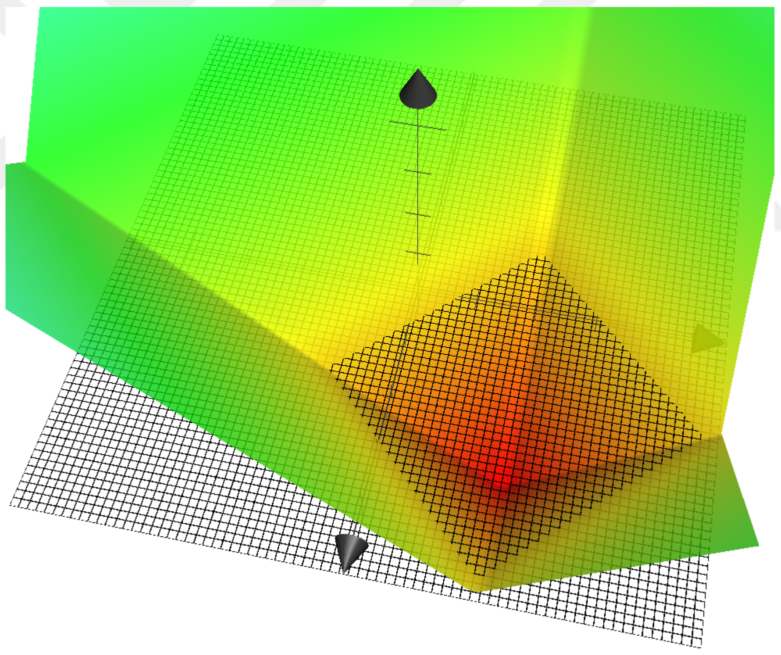

Figure 2.7. *Graph and level set of a PCF in the 2-dimensional feature space*

Given $\mathbf{w}, \mathbf{c} \in R^n$, $\xi, \gamma \in R$, the general form of conic function $g_{(\mathbf{w},\xi,\gamma,\mathbf{c})}$: $R^n \rightarrow R$ is defined as follows:

$$g_{(\mathbf{w},\xi,\gamma,\mathbf{c})}(\mathbf{x}) = \langle \mathbf{w}, (\mathbf{x} - \mathbf{c}) \rangle + \xi \left\| \mathbf{x} - \mathbf{c} \right\|_p - \gamma, \tag{2.23}$$

where $\|\mathbf{x}\|_p$ is an l_p-norm of the vector $\mathbf{x} \in R^n$ and $g(\mathbf{x})$ defines a conic function that can be used for constructing discriminating regions of two arbitrary sets: **A**

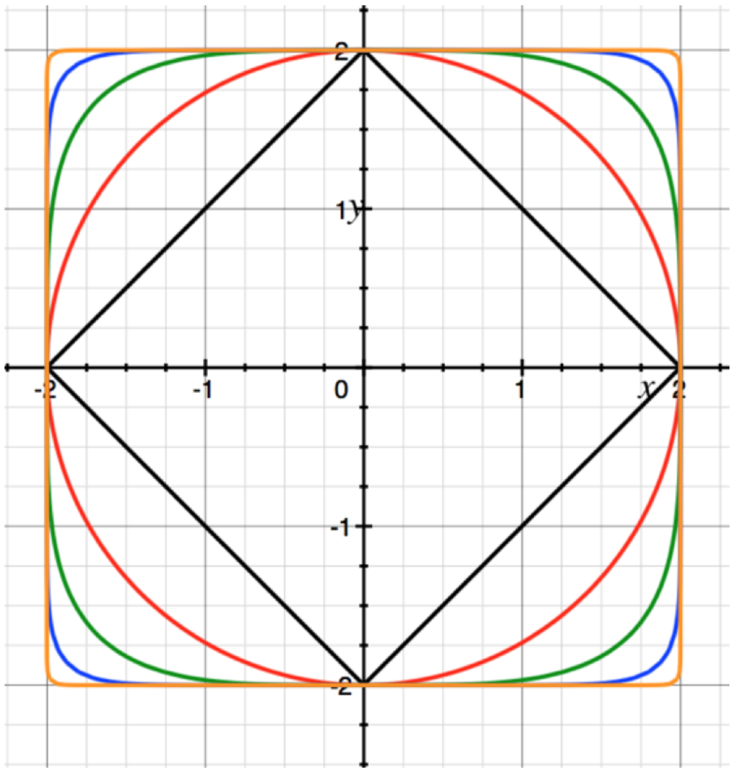

Figure 2.8. *Level sets of conic function for* $p = 1, 2, 4, 10$ *and 50, and* $\boldsymbol{w}^{\mathrm{T}} = [1, 1]$, $\boldsymbol{c}^{\mathrm{T}} = [0, 0]$, $\gamma = 2$

and **B**. Here, it must be noted that **x** is a point in the set and **c** is the Euclidian center (or selected from R^n with some approaches) (i.e., not the l_p center) of the set, that was already known before the solution of the LP. This center also corresponds to the lowest (vertex) point of the cone. The degree (p) of l_{p} norm can be varied to obtain a rich class of convex sets, ranging from 1 to ∞. In **Figure 2.8.**, level sets (horizontal intersection with the plane $z = 0$) of various cones are illustrated for $p = 1$, 2, 4, 10 and 50 and $\mathbf{w}^{\mathrm{T}} = [1, 1]$, $\mathbf{c}^{\mathrm{T}} = [0, 0]$, $\gamma = 2$:

Throughout this work, l_1-norm is used to define a particular conic shape, which is the "polyhedral conic shape":

$$g_{(\mathbf{w},\xi,\gamma,\mathbf{c})}(\mathbf{x}) = \langle \mathbf{w}, (\mathbf{x} - \mathbf{c}) \rangle + \xi \left\| \mathbf{x} - \mathbf{c} \right\|_1 - \gamma. \tag{2.24}$$

Using multiple different polyhedral conic functions, it becomes possible to separate and classify arbitrarily distributed (non-convex) sets **A** and **B**. PCF algorithms [29], k-means based PCF algorithm [30] and incremental PCF algorithm [31] have different center selection and updating strategies. In [29], the center points are randomly selected from the set **A** and classification was achieved by sequentially eliminating correctly classified points. In [30], the centers of PCFs are found by solving k-means algorithm and then a PCF is obtained by solving LP for each cluster. In [29] and [30] an eventual classifier is constructed as the point-wise minimum of all PCFs, requiring several PCF constructions. In [31], the classifier is obtained by minimizing a single error function in an incremental manner. In [32], a method is proposed to optimally estimate the vertex point of a PCF.

2.2. Descriptive Methods

In unsupervised learning, only the features are observed and there are no measurements of the outcome. The aim is to describe how the data are organized or clustered. The algorithms in this type are sometimes called knowledge discovery and/or descriptive models. Clustering algorithms and association analysis algorithms are included in this group.

Clustering algorithms split the data into groups containing similar data points. Various similarity measures can be defined. However, one of the commonly used one is the sum of the distances of the data points to corresponding cluster centers. Clustering is in used various areas such as astronomy, e-commerce, biology, customer segmentation and social network analysis in real-life.

The clustering problem is NP-hard. Therefore, efficient heuristic algorithms, which converge to local optimum solutions, are used generally. There are three distinct types of the clustering algorithms: combinatorial algorithms, mixture modeling, and mode seeking. Combinatorial algorithms work on data without using any probability model. Mixture models are described by probability density functions (pdf). The algorithm fits the model to the data by maximum likelihood or corresponding Bayesian approaches. Mode seekers attempt to estimate distinct modes of the pdf. [23]. k-Means, mean shift, density-based spatial clustering, expectation-maximization (EM), hierarchical clustering and self-organizing map are the well-known clustering algorithms. Since the following sections refer to k-Means and EM, details about these algorithms are provided in this section.

Association analysis of which purpose is to find joint values of the variables that appear most frequently is among descriptive methods. Association analysis is a popular tool for commercial data [23]. Market basket analysis is one of the most frequent applications of this problem. Including the Apriori algorithm, there are various algorithms for association analysis.

2.2.1. k-Means algorithm

k-Means is one of the famous and one of the simplest clustering algorithm. MacQueen introduced the k-Means in 1967 [33]. The k refers to the number of clusters and it must be defined priorly. The algorithm starts with k random centers from the dataset and then updates the centers to minimize the sum of the distances from each data point to the center that is closest to it. k-Means algorithm is given in Algorithm 2.1 where $\mathbf{x}_i$ is the data point, c_j is the j'th cluster center and z_i is the cluster label of i'th data point.

Algorithm 2.1. *The k-Means Algorithm:*

Step 1: Select k center points randomly from the dataset.
repeat
Step 2: Assign the data points to closest cluster center. z_i is the cluster label.

$$z_i = argmin_j \left\|\mathbf{x}_i - \mathbf{c}_j\right\|_2 \tag{2.25}$$

Step 3: Update each cluster center by calculating the mean of the all data points assigned to it:

$$\mathbf{c}_j = \frac{1}{N_j} \sum_{i:z_i=j} \mathbf{x}_i, \qquad \forall j \in \{1, \ldots, k\} \tag{2.26}$$

until converged

As stated the k-Means algorithm starts with random centers, which causes to obtain different local optimal solutions in each run. One way to cope with this problem is using multi-start method which finds the solution multiple times and chooses the best solution. Especially in large datasets, it is very hard to obtain global or near-global solutions with the k-Means algorithm. Nevertheless, the k-Means is still the most used clustering algorithm.

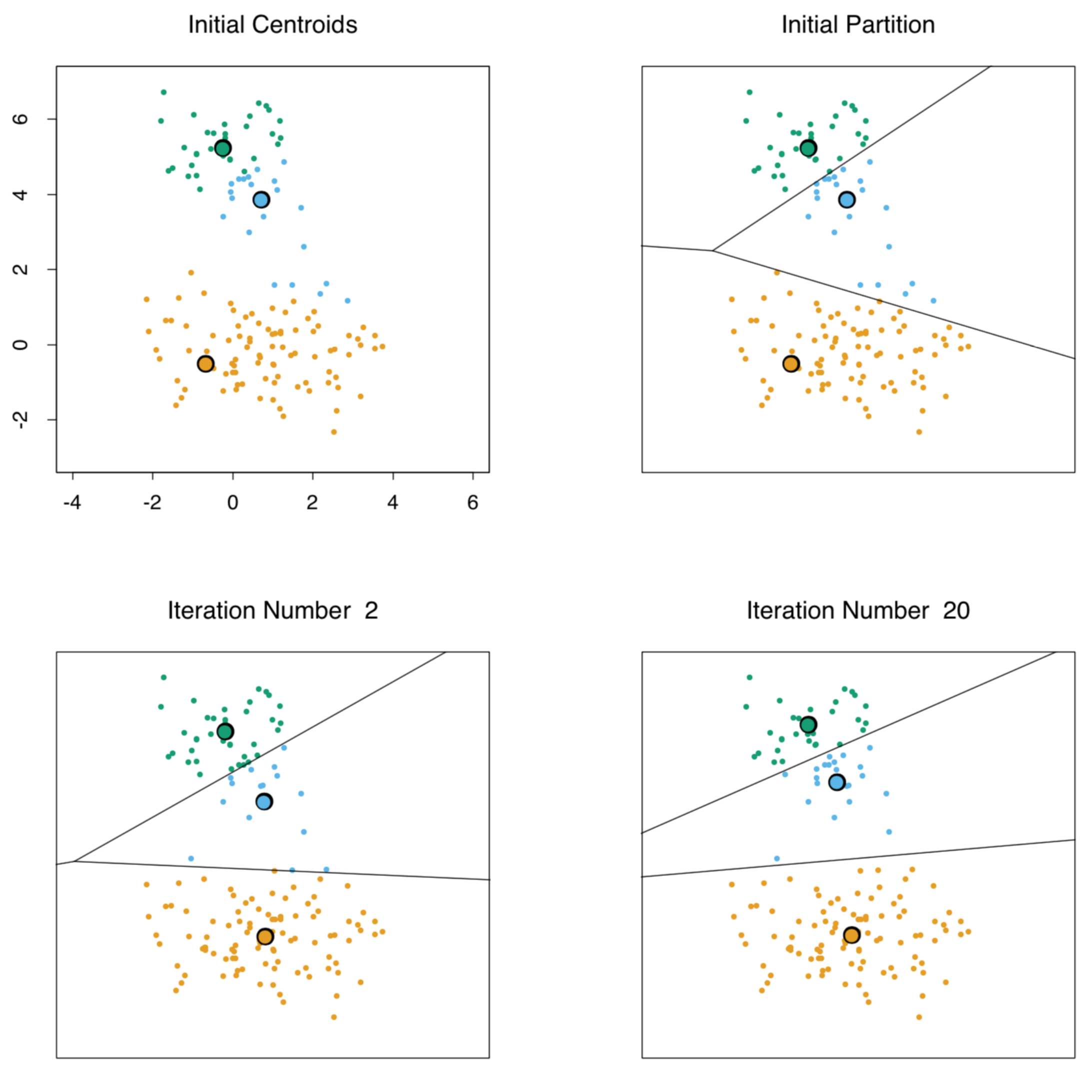

Figure 2.9. *Simulation of the k-Means clustering algorithm [23]*

2.2.2. Expectation maximization algorithm

Expectation Maximization (EM) algorithm was introduced by Dempster et al. in 1977 [34]. EM is an iterative algorithm which fits a mixture of Gaussians to the data.

There are two main steps in the EM: expectation and maximization. The expectation step assigns membership scores for each data point based on its relative density under each mixture component, and the maximization step updates the component density parameters based on the current membership scores [23]. The EM algorithm starts with a random solution and then converges to a locally optimal solution.

"The EM algorithm models the data by specifying a joint distribution $p(\mathbf{x}_i, z_i) = p(\mathbf{x}_i|z_i)p(z_i)$. $z_i \sim Multinomial(\phi)$ where $\phi_j \geq 0$, $\sum_{j=1}^{k} \phi_j = 1$, $\phi_j = p(z_i = j)$ and $\mathbf{x}_i|z_i = j \sim N(\mu_j, \Sigma_j)$." [35]. z_i's can take on k numbers.

The EM algorithm is given in Algorithm 2.2.

Algorithm 2.2. *The EM Algorithm:*

```
repeat
    Step 1: Expectation. Assume that the training set is ${\mathbf{x}_1, \ldots, \mathbf{x}_m}$. $z_i$'s are latent random variables. For each $i$, $j$ set:
        $w_j^i := p(z_i = j|x_i; \phi, \mu, \Sigma)$    (2.27)
    Step 2: Maximization:
        $\phi_j := \frac{1}{m}\sum_{i=1}^{m} w_j^i$    (2.28)
        $\mu_j := \frac{\sum_{i=1}^{m} w_j^i \mathbf{x}_i}{\sum_{i=1}^{m} w_j^i}$    (2.29)
        $\Sigma_j := \frac{\sum_{i=1}^{m} w_j^i (\mathbf{x}_i - \mu_j)(\mathbf{x}_i - \mu_j)^T}{\sum_{i=1}^{m} w_j^i}$    (2.30)
until converged
```

Unlike k-Means, EM finds the cluster membership probabilities of each data point. This type of clustering is named soft clustering.

3. INCREMENTAL CONIC FUNCTIONS ALGORITHM FOR LARGE SCALE CLASSIFICATION PROBLEMS

The ever-increasing internet bandwidth together with mass storage has long required mining, clustering and classification [36], for applications ranging from finger print or iris recognition based security technologies to detecting spam e-mails, to gesture or face recognition. These technologies improve innovative aspects of commercial products or improve the convenience of the provided service. Consequently, researchers have been interested in classification problems for years. The early research on classification focused mostly on feature extraction and classifier optimization by means of finding best separating functions/surfaces. Unfortunately, the ever-increasing data amount enforced researchers to adopt different strategies to handle the new problem of big data. According to a not-so-new IDC Digital Universe Study [3] collected data doubles in every two years. Nowadays, it is argued that the doubling period is also shrinking. Because of this, developed algorithms must be appropriate to work with large datasets for both in training step (classifier construction) and test step (application). Any improvement to reduce computation times by multiple-folds would be welcome.

Naturally, a necessary property of classification algorithms is its classification accuracy. In that aspect, the developed algorithms are expected to perform well with respect to standard classifiers such as Bayesian classifiers [37], artificial neural networks [38], decision trees [39, 40], and support vector machines [17].

Starting from 1960's there has been an interest to classification algorithms based on mathematical programming. A literature example is by Bennett and Mangasarian, where a robust approach for linear separation was developed [41]. In another work, Astorino and Gaudioso used more than one hyper planes to separate two sets; which were found with mathematical programming [42]. Max–Min separation is another related successful approach developed by Bagirov [43]. Similarly, Uney and Turkay developed an Integer Programming algorithm to classify more than one classes with hyper boxes [44]. One of the most famous and commonly used classification algorithms is Support Vector Machines (SVMs), which is based on quadratic programming methods [17]. A survey based on SVMs and its latest improvements can be found in [45]. In [46] non-smoothness in classification, in [47] non-linear programming in classification, in [48] margin maximization based on polyhedral separability, and in [49] ellipsoidal separation for classification problems were investigated. Finally, classification with truncated l_1 distance kernel was introduced in [50]. This presented work is also considered within the class of classification algorithms that use LP.

A critical idea that is utilized in this paper is to construct and use Polyhedral Conic Functions (PCFs), which were first proposed for classification by Gasimov and Ozturk [29] and the classification method was also named PCF algorithm [36]. The authors then combined the PCF approach with Bagirov's Max–Min separation algorithm [51]. Later, clustering based PCF algorithm [30] was developed and successfully applied to real life problems such as arrhythmia classification [52] and gesture recognition [53]. An incremental piecewise linear classification algorithm based on polyhedral conic separation was also introduced in [31].

In this section, the highly accurate clustering based PCF [30] is considered as a starting point, and the part of the creation of the LP problem, to construct the classifier, is smartly modified to improve the computational efficiency, which is expected to enable processing large datasets. The clustering based PCF is a novel method which outperforms several state-of-the-art classifiers, including SVM [30]. However, in its original version, a linear constraint exists in the LP model for each additional data point. Consequently, when data size is very large, the model had too many constraints to be handled. In this section, unnecessary data points (hence constraints) after the clustering stage are eliminated. The new strategy is observed to keep the high accuracy of PCFs on datasets from UCI Machine Learning Repository while reducing computation times and avoiding over-fitting with fewer conic functions.

The rest of the section is organized as follows. In Subsection 3.1., the proposed algorithm is explained with algorithmic layouts and example illustrations. In Subsection 3.2., experimental results are given. Finally, conclusions about proposed method are provided in Subsection 3.3.

3.1. Incremental Conic Functions (ICF) Algorithm

3.1.1. Review of k-means based polyhedral conic functions (PCFs)

In the k-means based Polyhedral Conic Function approach [30], the classifiers are constructed with the simultaneous use of the polyhedral conic separation approach [29] and the k-means clustering technique. This algorithm applies k-means algorithm to find centers of PCFs. In order to construct the classifier for a specific class, the class is first divided into sub-clusters via k-means algorithm. Then, an LP is solved for each cluster in order to obtain a PCF that separates the cluster from the other classes. Thus, k PCFs are obtained after this operation. The classifier of the selected class is obtained as a point-wise minimum of k separate PCFs. These steps are repeated for each class. If the number of classes is η, the total number of constructed PCFs is $\eta \times k$. In the test phase, a data point is applied to each of these $\eta \times k$ PCFs, and the class η_i is chosen for the PCF function which yields the minimum function value. In the original form of the polyhedral conic separation method [29], the number of PCFs to be tested was huge as compared to the limited ($\eta \times k$) number of PCFs. Therefore, the use of clustering algorithm allows to significantly decrease the number of centers and consequently the number of PCFs which makes the algorithm reasonable for real life applications and helps to avoid over-fitting problem.

3.1.2. Proposed algorithm: incremental conic functions (ICF)

In this subsection, in order to eliminate drawbacks of k-means based PCF we propose a new algorithm called Incremental Conic Functions (ICFs). A drawback of the above k-means based PCF is the need for a-priori determination of "number of clusters" in the beginning of the algorithm. Another drawback is the need for knowledge for the size of the LP model because the LP model in k-means based PCF directly uses all data points as constraints for LP. In k-means based PCF, the number of LP solving steps is, therefore, (number of classes) $\times$ (k). This repetitive structure is time consuming for large datasets.

The ICF algorithm resolves the problem of "choice of k" because, unlike the previous version in [30], the parameter, k, is determined incrementally within the algorithm, itself. Overall, it is argued that the use of clustering algorithms allows us to significantly decrease the number of vertices, which consequently reduces the number of PCFs to avoid over-fitting problem.

The proposed algorithm also aims to eliminate redundant constraints. In this elimination process, first of all, radii of data points in set **A** calculated and used to find, so called, "*purity*" of the set **A**. If the purity of set **A** is less than a pre-set threshold (Algorithm 3.2, Step 2), data points in set **A** are partitioned into two clusters with k-means algorithm ($k = 2$). Then radii of these clusters are calculated and used to find "*purity*" of the clusters (Algorithm 3.2 , Step 2). Following this, the average purity is calculated. If this average is greater than a pre-set threshold, the clustering operations stop. If not, the cluster count (k) is increased by one and the calculations are repeated. This incremental process can be stopped if the purity conditions are met at a certain k value, or if the clusters become too small (with respect to a threshold). Here, the purity term naturally provides an indication of how well the combination of PCF regions (clusters) cover the data belonging to the class of interest. If majority of the data within a cluster belongs to the same class, then the cluster is considered to be pure (induced as the opposite of impurity [54]). The mathematical definition is as provided in Equation 3.18.

Once the clusters are set (together with their corresponding purities), a pre-check for all clusters with a desirable level of purity greatly reduces the computation time. If a cluster is *very* pure, then, there is no need to solve an LP for PCF construction. The separating cone can be analytically obtained (in a single step, given in Algorithm 3.1, Step-3) with the available radius with no further PCF steps. This situation constitutes a significant achievement of the proposed ICF algorithm.

If a cluster is not pure enough (as opposed to the case in the upper paragraph), then the LP model must be solved. However, even at this stage, a significant computational improvement (through smart data elimination) is proposed herein. First, the largest radius inside a cluster is calculated and it is scaled with a parameter (Algorithm 3.2, Step-3) to get the threshold for data elimination. Then all the data points of which distance is less than that threshold can be totally omitted from the LP solver for that cluster. This leaves the LP algorithm to handle only few repre-

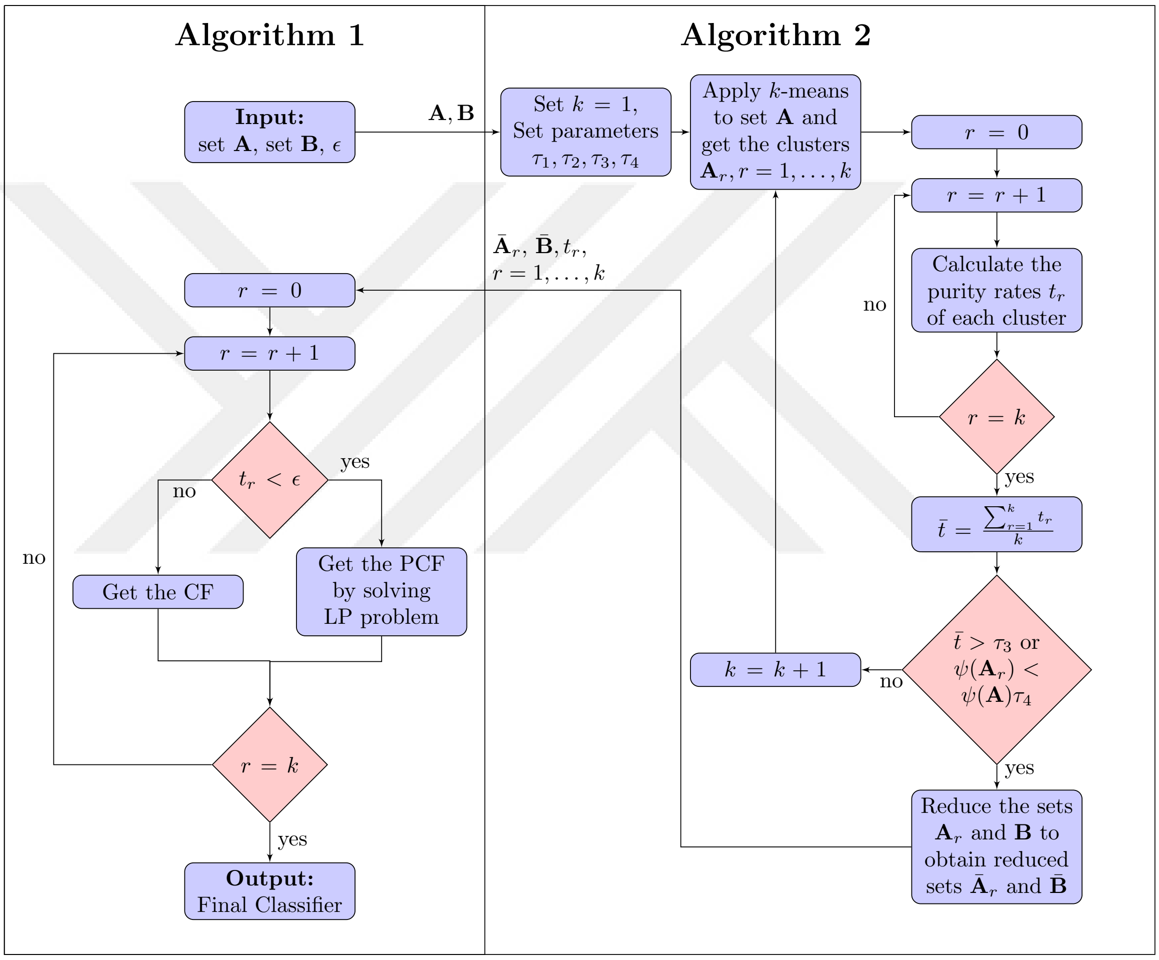
Algorithm 1
Algorithm 2
Input: set $\mathbf{A}$, set $\mathbf{B}$, ϵ
$\mathbf{A}, \mathbf{B}$
Set $k = 1$, Set parameters $\tau_1, \tau_2, \tau_3, \tau_4$
Apply k-means to set $\mathbf{A}$ and get the clusters $\mathbf{A}_r, r = 1, \ldots, k$
$r = 0$
$r = r + 1$
Calculate the purity rates t_r of each cluster
no
$r = k$
yes
$\bar{t} = \frac{\sum_{r=1}^{k} t_r}{k}$
$\bar{t} > \tau_3$ or $\psi(\mathbf{A}_r) < \psi(\mathbf{A})\tau_4$
no
$k = k + 1$
yes
Reduce the sets $\mathbf{A}_r$ and $\mathbf{B}$ to obtain reduced sets $\bar{\mathbf{A}}_r$ and $\bar{\mathbf{B}}$
$\bar{\mathbf{A}}_r, \bar{\mathbf{B}}, t_r, r = 1, \ldots, k$
$r = 0$
$r = r + 1$
$t_r < \epsilon$
yes
no
Get the CF
Get the PCF by solving LP problem
no
$r = k$
yes
Output: Final Classifier

Figure 3.1. *The flowchart of the ICF Algorithm*

sentative data samples along the *shell* of the cluster, while eliminating all obvious central portions (which do not really contribute to the PCF shape at the end of LP). The central data elimination step can be visualized in **Figure 3.2.**, where **Figure 3.2.**(b) contains all data points for a cluster of the yellow-labeled class, and **Figure 3.2.**(e) shows the elimination of central (completely pure) portion of the cluster for faster LP solving. It must be noted that the constructed PCF (at the end of an LP) for the yellow cluster in **Figure 3.2.**(d) and the resulting PCF for the yellow cluster in **Figure 3.2.**(e) will be covering the corresponding classes equally well for both cases.

For multi-class problems, classifiers are obtained for each class by using one-against-all approach. In that approach, a classifier is obtained for the points belong to a class (considered as set **A**) and all the points belonging to other classes considered as set **B**. Then, further classifier generation steps are performed for each class within the previous set **B**.

The overall ICF construction for generating multiple conic functions through reduced LPs can be explained as nested calling of two sub-routines, it is called as *Algorithm 3.1* and *Algorithm 3.2*. The algorithm nesting is briefly explained in **Figure 3.1.** and detailed algorithm steps are presented as pseudo-codes below.

Algorithm 3.1.

$$\mathbf{A} = \{\mathbf{a}^1, \ldots, \mathbf{a}^m\}, \quad \mathbf{a}^i \in R^n,\ i \in I = \{1, \ldots, m\} \tag{3.1}$$

$$\mathbf{B} = \{\mathbf{b}^1, \ldots, \mathbf{b}^p\}, \quad \mathbf{b}^j \in R^n,\ j \in J = \{1, \ldots, p\} \tag{3.2}$$

Define: $\psi\{\cdot\}$ operator as the number of data points in the corresponding set (number of rows in the matrix).

Step 0: Choose $\epsilon \in [0,\ 1]$

Step 1: Apply Algorithm *3.2* using set **A** and set **B**. Get the reduced sets as $\overline{\mathbf{A}}_r$ and $\overline{\mathbf{B}}$, purity rates of $\overline{\mathbf{A}}_r$ as t_r and cluster centroids for each cluster as $\mathbf{c}_r$.

Step 2: Finding g_r: For each $r \in \{1, \ldots, k\}$ repeat this step.

If $t_r < \epsilon$:

Determine g_r with parameters $(\mathbf{w}^r, \xi_r, \gamma_r, \mathbf{c}_r)$ by solving the corresponding LP problem, LP_r (described below). Here, g_r corresponds to the polyhedral conic function for $\overline{\mathbf{A}}_r$. While solving LP_r, use reduced sets $\overline{\mathbf{A}}_r$ and $\overline{\mathbf{B}}$, centers $\mathbf{c}_r \in R^n$, $r = 1, \ldots, k$.

The LP_r is defined as the following optimization:

$$\min \frac{1}{\psi\{\overline{\mathbf{A}}_r\}} \sum_{i \in \overline{I}_r} y_i + \frac{1}{\psi\{\overline{\mathbf{B}}\}} \sum_{j \in \overline{J}} z_j \tag{3.3}$$

subject to

$$\langle \mathbf{w}^r(\mathbf{a}_i - \mathbf{c}_r)\rangle + \xi_r \left\|\mathbf{a}_i - \mathbf{c}_r\right\|_1 - \gamma_r + 1 \leq y_i, \quad \forall i \in \overline{I}_r \tag{3.4}$$

$$-\langle \mathbf{w}^r, (\mathbf{b}_j - \mathbf{c}_r)\rangle - \xi_r \left\|\mathbf{b}_j - \mathbf{c}_r\right\|_1 + \gamma_r + 1 \leq z_j, \quad \forall j \in \overline{J} \tag{3.5}$$

$$y_i > 0, \quad z_j > 0 \tag{3.6}$$

else: Calculate $radius_a$ and $radius_b$ as follows:

$$radius_a = \max_{i \in \overline{I}_r} \left\|\mathbf{a}_i - \mathbf{c}_r\right\|_2 \tag{3.7}$$

$$radius_b = \min_{j \in \overline{J}} \left\|\mathbf{b}_j - \mathbf{c}_r\right\|_2 \tag{3.8}$$

$$\gamma_r = \frac{radius_a + radius_b}{2} \tag{3.9}$$

to obtain g_r function;

$$g_r(\mathbf{x}) = \left\|\mathbf{x} - \mathbf{c}_r\right\|_2 - \gamma_r \tag{3.10}$$

Please note that Equation 3.10 is the simplified form of the conic functions given in Equation 2.23. Here; $p = 2$ denotes that the norm used is l_2-norm; $\mathbf{w} = \mathbf{0} \in R^n$ and $\xi = 1$.

Step 3: Determine the final classifier function as the minimum function surface of the found g_rs:

$$G(\mathbf{x}) = \min_{r=1,\ldots,k} g_r(\mathbf{x}), \tag{3.11}$$

where "min" operator renders a combination surface by range-wise combining the *smallest*-valued conic functions (g_rs).

The following set of pseudo-codes explain the nestedly-called Algorithm 3.2, which processes $\mathbf{A}$ and $\mathbf{B}$ sets to yield:

- the value for number of clusters: k,
- reduced subsets of $\mathbf{A}$ as $\overline{\mathbf{A}}_r$,

- reduced set $\mathbf{B}$ with eliminated points as $\overline{\mathbf{B}}$,
- subset purities as t_r, and
- cluster centers as $\mathbf{c}_r$;

and significantly reduce the computation time.

Algorithm 3.2.

Step 0: Set k =1 and choose $\tau_1, \tau_2 \in [0, \infty]$, $\tau_3, \tau_4 \in [0, 1]$.
Step 1: Apply the k-means Algorithm to $\mathbf{A}$. Get clusters $\mathbf{A}_r$, and their centers, $\mathbf{c}_r$, $r \in \{1, \ldots, k\}$
Step 2: Determine the purity rate of each cluster $\mathbf{A}_r$ as t_r, through the following calculations:

$$I_r = \{i \in I : \mathbf{a}_i \in \mathbf{A}_r\} \tag{3.12}$$

$$J = \{j \in J : \mathbf{b}_j \in \mathbf{B}\} \tag{3.13}$$

$$radius_a = \frac{\sum_{i \in I_r} \|\mathbf{a}_i - \mathbf{c}_r\|_2}{\psi\{\mathbf{A}_r\}} \tag{3.14}$$

$$radius_b = \frac{\sum_{j \in J} \|\mathbf{b}_j - \mathbf{c}_r\|_2}{\psi\{\mathbf{B}\}} \tag{3.15}$$

$$l = \max_{i \in I_r} \|\mathbf{a}_i - \mathbf{c}_r\|_2 \tag{3.16}$$

$$\tilde{\boldsymbol{B}} = \{\mathbf{b} \in \mathbf{B} : \|\mathbf{b} - \mathbf{c}_r\|_2 < l\} \tag{3.17}$$

$$t_r = \frac{\psi\{\mathbf{A}_r\}}{\psi\{\mathbf{A}_r\} + \psi\{\tilde{\mathbf{B}}\}} \tag{3.18}$$

where t_r is the purity of the cluster, r. Let's define an average purity of these r clusters as $\bar{t} = \frac{1}{k}\sum_{i=1}^{k} t_i$. If $\bar{t} > \tau_3$ or $\psi\{\mathbf{A}_r\} < \psi\{\mathbf{A}\} \times \tau_4$, then go to *Step 3*. Otherwise increment $k := k + 1$ and go to *Step 1*.

Step 3: Determine the sets $\overline{\mathbf{A}}_r$ and $\overline{\mathbf{B}}$ as follows and pass the all calculation results to Algorithm 3.1.

$$\overline{\mathbf{A}}_r = \{\|\mathbf{a}_i - \mathbf{c}_r\|_2 > radius_a \times \tau_1\}, \quad \forall i \in I_r \tag{3.19}$$

$$\overline{\mathbf{B}} = \{\|\mathbf{b}_j - \mathbf{c}_r\|_2 < radius_b \times \tau_2\}, \quad \forall j \in J \tag{3.20}$$

$$\overline{I}_r = \{i \in I : \mathbf{a}_i \in \overline{\mathbf{A}}_r\} \tag{3.21}$$

$$\overline{J} = \{j \in J : \mathbf{b}_j \in \overline{\mathbf{B}}\} \tag{3.22}$$

With the above algorithm, all data points that have limited or no contribution for k-means based PCF algorithm are eliminated. Consequently, several redundant constraints of LP model are eliminated.

Both Algorithm 3.1 and Algorithm 3.2 can be considered as the contribution of this paper. Starting from **Figure 3.2.**, an illustrative example with three classes, shown in green, yellow and purple is presented. The one-against-all step starts by choosing the yellow-labeled class as class-A and combining the other two classes to another class, labeled in black (**Figure 3.2.**(b)). Assume that $k = 1$ does not satisfy the stopping criteria in Algorithm 3.2, to separate all of yellow points from all of black points, the algorithm (Step 1 of Algorithm 3.2) makes $k = 2$ in the next step, and considers the PCF count to become 2 (illustrated with dark- and light-yellow in **Figure 3.2.**(c)). Then, the central (convex-part) data points from the light-yellow point cloud is eliminated for speed improvement before obtaining a PCF (g_r-1) for this light-yellow set (**Figure 3.2.**(e)). It must be noted that these last steps must be repeated for the dark-yellow subset and its corresponding PCF (g_r-2) must be merged with the previous reduced PCF (g_r-1) to come up with the overall PCF of the yellow class (g).

Similar to the steps acting upon the yellow class, the combination of individual PCFs (or cones) over the green class is illustrated. The iterative k-means clustering step provides that the total number of clusters for the green class is 3. According to the illustrative example, the two clusters in the farther side are found to be pure enough. Therefore, the corresponding conic functions are directly (algebraically) found (via Algorithm 3.1, Step-2). The algorithm yields two simple cones for the first part of the disconnected green set (**Figure 3.3.**). The cone expressions are quickly determined as simple analytical expressions.

For the third cluster of the green class, the purity level is determined to be less than the threshold. This necessitates polyhedral conic function construction by solving LP. After running the necessary algorithms in the LP steps, the eventual PCF level-set is obtained. **Figure 3.4.** shows the overall conic functions (from **Figure 3.3.**) and the new PCF. Notice that the sides of the classifier corresponding to the third green-labeled cluster do not form a regular cone, but a PCF. By

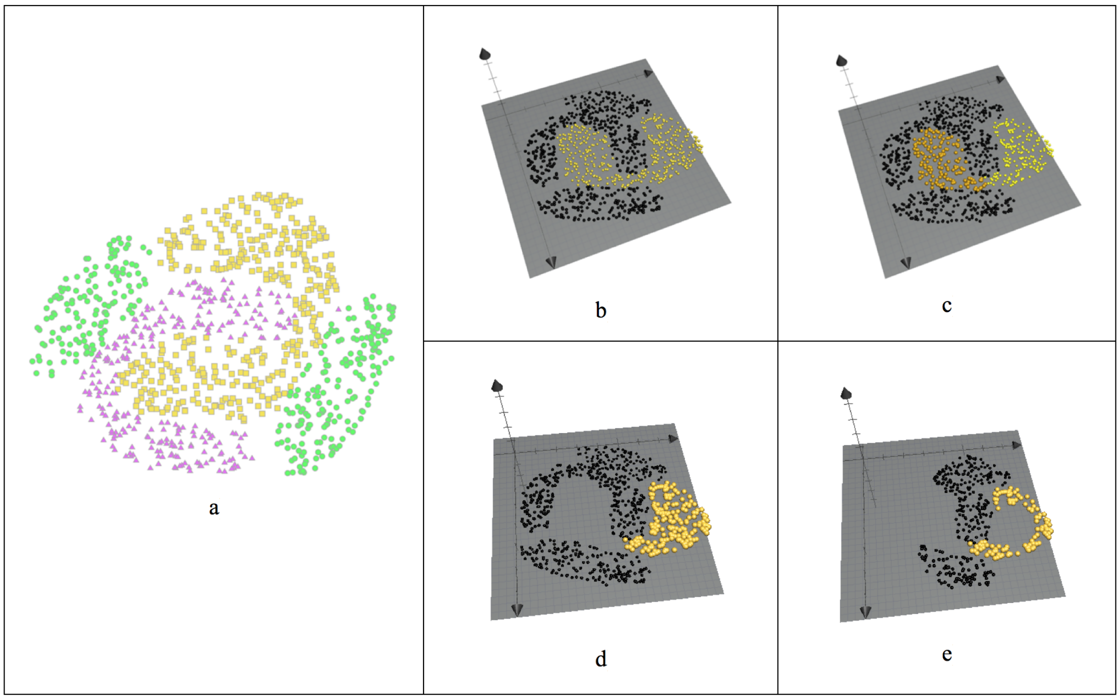

Figure 3.2. *An illustration of splitting and data reduction (Algorithm 3.2) (a) Three classes with green, purple and yellow labels (b) Selection of the yellow class as class A and the rest as set B (black labeled) (c) Splitting class A into two clusters if $k = 2$ (light and darker yellow: A_1 and A_2) (d) Considering the light yellow part as the selected sub-cluster: A_1 (e) Reducing the sub cluster A_1 by eliminating central data points and obtaining $\overline{A}_1$ while also obtaining the reduced set $\overline{B}$ The clusters are illustrated as the horizontal plane in 3D for compatibility with* ***Figure 3.2.*** *and onwards, where the vertical axes correspond to a dimension to represent the cone or PCF value*

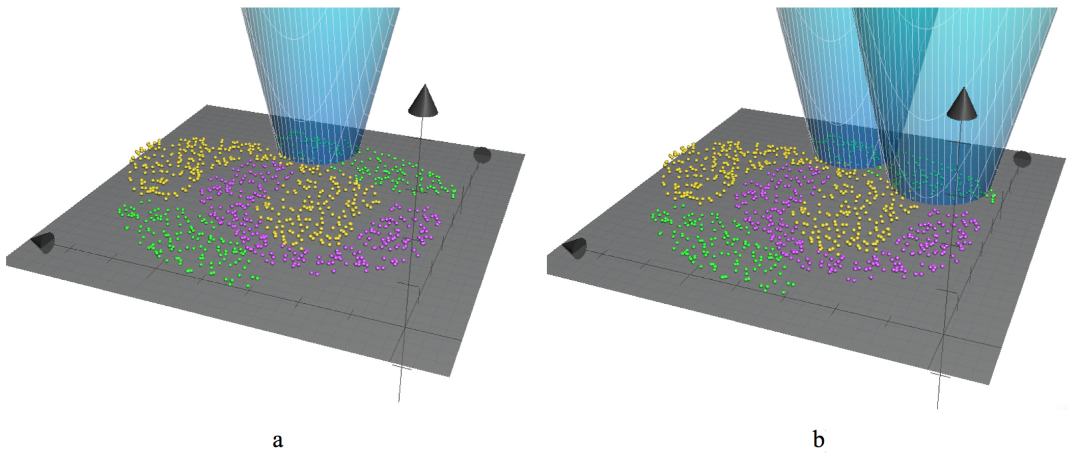

Figure 3.3. *(a) Solution of a conic function that best separates a sub-cluster of green-labeled set (b) Combination of two such conic functions, separately optimized for different sub-clusters of green-labeled set*

combining this PCF with the previous two cone functions, the final classifier is achieved. **Figure 3.5.** shows the final classifier as the point-wise minimum surface of three functions. For clarity, top and bottom views are presented in **Figure 3.4.** and **Figure 3.5.**

3.2. Computational Tests

In this section, training and test accuracy rates of the ICF are presented and the ICF is compared with k-means based PCF algorithm. Due to elimination of certain data points, such comparison is necessary for motivating the proposed method. The algorithms are implemented using Python 2.7. Linear programming model is solved with the Gurobi package [55]. Tests are carried out on a computer with Intel(R) Core(TM) i7 CPU running at 2.5 GHz and 16 GB RAM.

By definition $\tau_1, \tau_2 \in [0, \infty]$, $\tau_3, \tau_4 \in [0, 1]$. We choose $\tau_1 = 0.9$, $\tau_2 = 1.1$, $\tau_3 = 0.95$ and $\tau_4 = 0.05$, which were heuristically selected, considering a compromise between speed versus accuracy. The data points for classification problems are pre-divided to training and test sets as described in their showcase. Properties of the datasets are given in **Table 3.1.**. The selection of ϵ, τ_1, τ_2, τ_3 and τ_4 parameters must be performed according to the following facts: The ϵ parameter is a

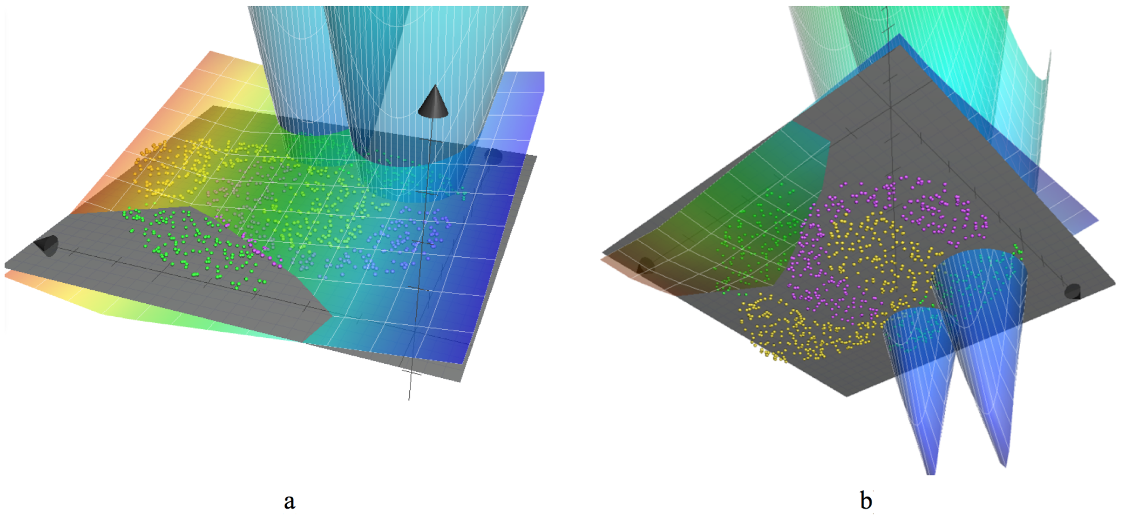

Figure 3.4. *Three conic functions (rendered as colored surfaces) obtained by ICF algorithm are illustrated by top and bottom views (a)–(b)*

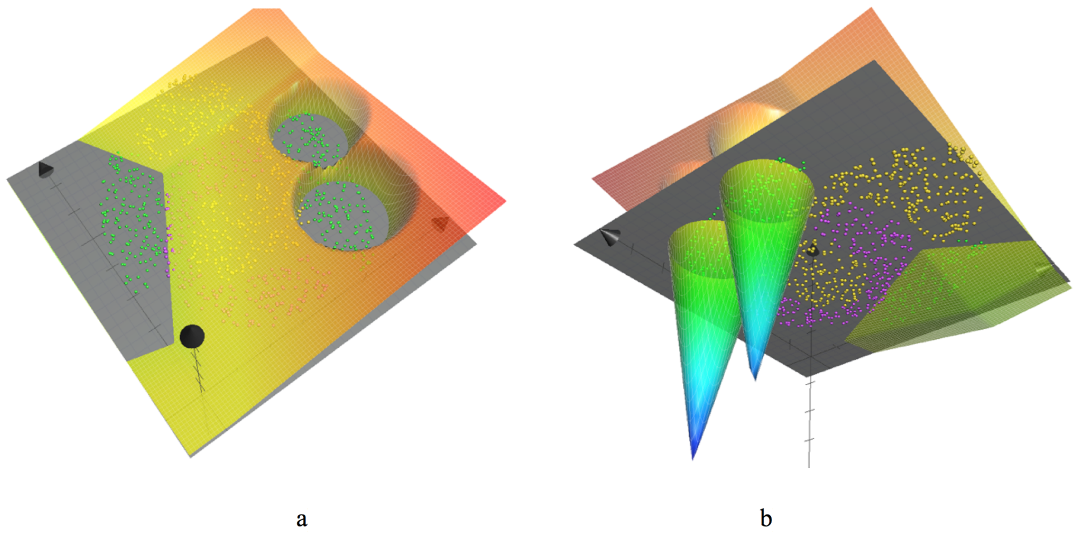

Figure 3.5. *Top and bottom views (a)–(b) of the final classifier obtained as point-wise minimum of the three conic functions in* ***Figure 3.4.***

Table 3.1. *Datasets and their corresponding properties*

Dataset	# of training data points	# of test data points	# of features	# of classes
Abalone [56]	3133	1044	8	3
Page blocks [57]	4000	1473	10	5
Satellite [58]	2000	4435	36	6
Shuttle [59]	43500	14500	9	7
Cover type [60]	15120	565892	54	7

Table 3.2. *Train and test set accuracy rates, together with combined run-times*

Dataset	Proposed Algorithm-ICF			k-means based PCF			
	Train	Test	Time (s)	k	Train	Test	Time (s)
Abalone	48.80	47.70	16.02	5	48.50	46.7	40.90
Page blocks	89.10	86.08	27.30	3	93.75	81.13	98.00
Satellite	87.40	86.10	181.50	6	87.60	86.10	372.20
Shuttle	93.48	93.47	2671.20	3	96.42	96.50	13038.50
Cover type	61.43	53.60	1242.33	4	67.78	52.45	2972.08

threshold to decide whether an LP is necessary, or not, in the ICF algorithm. High values for this parameter increases the chance of calling the LP for PCF construction, while its low values favor for algebraic cone construction (corresponding to a faster, but possibly lower resolution result). The parameter τ_1 defines the ratio of how much of redundant data are to be eliminated from set **A**. As τ_1 increases, more data elimination occurs. Similarly, τ_2 defines the data elimination ratio for set **B**. If more data is eliminated from sets, the operations become faster while reducing the training accuracy. For example, although choice of τ_1 and τ_2 as infinity is theoretically allowed, it causes a complete loss of accuracy. Therefore, a practicing engineer/scientist should naturally avoid choices of ill conditioned values for these parameters. The other two parameters, τ_3 and τ_4, define thresholds to increase/decrease number of sub-sets. Details of the sample implementations can be reached from [8, 9].

The retained accuracy (as opposed to the original – and accurate – k-means based PCF) while the reduced computational time results clearly indicate that the proposed algorithmic improvements pay. The "k" values in the first column of the

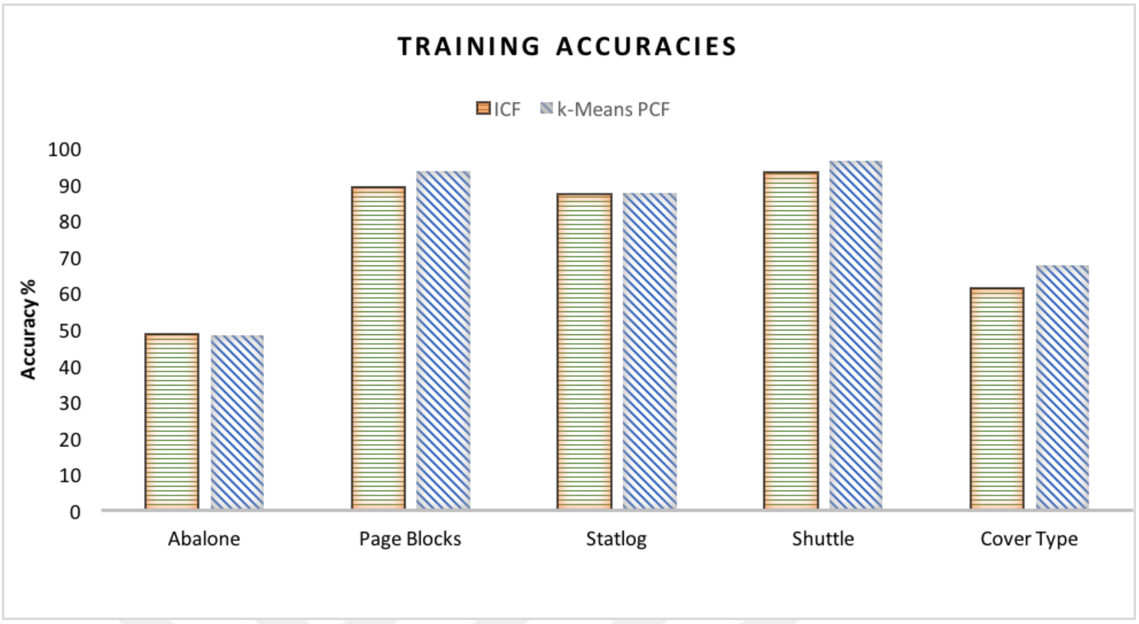

Figure 3.6. *Training Accuracies, bars with horizontal lines for the proposed, bars with diagonal lines for the previous k-means PCF*

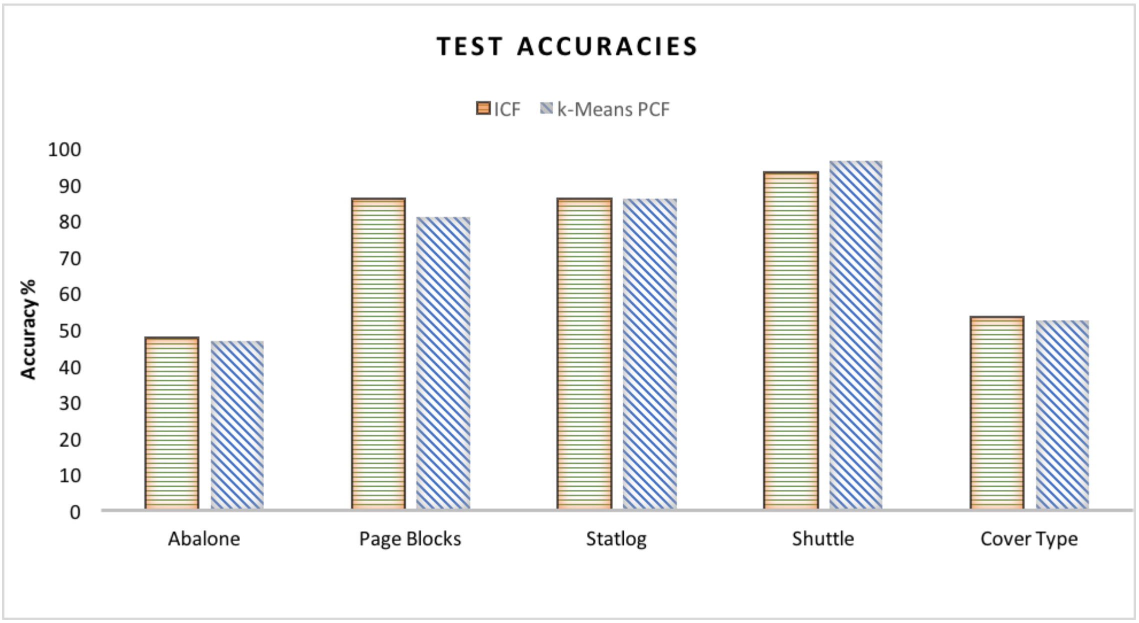

Figure 3.7. *Test Accuracies, bars with horizontal lines for the proposed, bars with diagonal lines for the previous k-means PCF*

right part (k-means based PCF) are, in fact, determined *using* the proposed incremental algorithm (for fair comparison). Therefore, it must also be noted that the original k-means PCF algorithm does not have a methodological approach for determining the k value to begin with. Hence our incremental algorithm also provides another advantage of determining the k value. Finally, it must be noted that the theoretical computational complexity of the overall ICF algorithm necessarily depends on the complexity of the LP in Algorithm 3.1, Step 2 (if part). Therefore, the less the algorithm hits this part, the faster the program runs. The contribution of this work is, therefore, by reducing the number of times this *time-consuming* LP runs.

In **Figure 3.6.** and**Figure 3.7.**, training and test accuracy rates for five datasets are given in chart form for comparing the performances of the two methods. One can see that the data elimination does not adversely affect classification accuracy. In fact, by eliminating the over-fitting problem, test accuracies occasionally increase. From **Table 3.3.**, the improvement in actual computation times is clear. In order to average-out the effects of training sample selection in the performance, the experiments are repeated 10 times by randomly selecting the training and testing samples from each dataset, including the recommended training-test separation by the authors of these datasets. The results presented in **Table 3.2.** are the averages of these 10 runs. In this table, it can be observed that the test accuracies seem not to be adversely affected by the proposed incremental steps which causes significant time improvement. In fact, in three of the five sets, the test accuracies increase and in one case it does not change. Since the given run times correspond to the total durations from test and training phases, and since the test run times were set to correspond to the same amount of comparisons (in the new ICF and the previous k-means based PCF), it can be concluded that the run time reduction is mostly due to the savings from the execution times during the training phase. Clearly, by fine tuning according to the structure of the datasets, several state of the art methods can give better results in these available datasets. For instance, Dozono and Nakakuni reports that a test accuracy of around 60% could be achieved for the Abalone dataset using FP-SOM [61]. On the other hand, this work is not intended to produce the highest attainable test accuracies; it only

Table 3.3. *Time improvements for different datasets*

Dataset	**Abalone**	**Page blocks**	**Satellite**	**Shuttle**	**Cover type**
Improvement	60.83%	72.14%	51.24%	79.51%	58.19%

proposes an improvement to the already available PCF approaches.

The time improvements achieved with the ICF algorithm are presented as percentages in **Table 3.3.** Percentages are calculated as:

$$improvement = \frac{time_{k\text{-}means\ PCF} - time_{ICF}}{time_{k\text{-}means\ PCF}} \tag{3.23}$$

3.3. Conclusions On The ICF

In this section, a new optimization based classification method using k-means clustering algorithm with conic functions is proposed. The conic functions already constitute a successful classifier for several real-life problems. On the other hand, they require repeated solutions of LPs, rendering the algorithm questionable for large datasets. By systematically looking for occasions of LP-avoidance, and further eliminating non-representative data points (for the construction of separating surfaces), the total LP run-time is significantly reduced. The LP-avoidance occurs whenever a cluster within a class has high purity. Then, a conic function can be analytically obtained without even solving the LP sub problem. When there is no possibility of constructing an analytical cone representation, the LP becomes necessary. But, even in that case, our new algorithm further eliminates the central pure portion of clusters (which do not contribute to the formation of PCF shapes) and significantly reduces the number of data points to enable LP to be solved faster. These two time-saving aspects experimentally made the method to run in about 64% less time than the previous implementation of the k-means based PCF algorithm (on typical PC configurations). Besides, the proposed method doesn't require the a-priori knowledge of number of clusters (k) in the clustering stage, and iteratively converges to the correct selection of k. The experimental results (Subsection 3.2.) are in accordance with the expected improvements.

With the above observations, the proposed method is expected to contribute well to the novel k-means based PCF classifier, which is argued to be a plausible alternative for the big data community [30]. The k-means PCF is a relatively new method which outperforms the celebrated SVM in several applications [30]. However, its extensive usage of LP was a drawback of the algorithm for larger datasets. Therefore, the presented time improvement is expected to attract interest in various classification problems. It remains an interesting problem to pursue an approach to solve the particular LP without requiring simplex algorithm during PCF classifier construction to further improve computational speeds.

4. O-PCF ALGORITHM FOR ONE-CLASS CLASSIFICATION

Supervised data classification is an important task in data mining and machine learning. The aim of supervised data classification is to label test data by constructing a function using a classification algorithm based on training data. There are many classifiers based on statistical methods and optimization approaches. Among these classifiers, optimization-based classifiers have been shown to be particularly effective.

The one-class classification problem differs in one essential aspect from the conventional classification problem. In one-class classification, it is assumed that information on only one of the classes, the target class, is available. This means that only example objects from the target class can be used and that no information about the other class, namely, the class of outlier objects, is present. The decision boundary between the two classes must be estimated from data corresponding to only the target (normal, genuine) class. The task is to define a decision boundary around the target class such as many as possible of the target objects are accepted, while the chance of accepting outlier objects is minimized [62]. The term one-class classification originates from Moya [63]. This type of classification is also known as concept learning [64], outlier/novelty detection [65, 66], anomaly detection [67, 68] or single-class classification [69]. Although the classifiers are typically trained only on the target class, there are some classification algorithms that use the poorly sampled outlier class (the complementary set to the target class) or unlabeled data in addition to the target class [70]. The problem of building text classifiers using positive and unlabeled examples is studied and the biased SVM algorithm is presented in [71]. In [72], Elkan and Noto studied identifying protein records problem and used only positive and unlabeled data in order to construct the classifier. One-class classification algorithms offer solutions to important problems. For instance, for very rare diseases, it can be difficult to obtain patient data; similarly, acquiring fault information that must be obtained to enable the advance detection of machine failures can be difficult. Similar difficulties also arise in object recognition, document classification, spam detection, speaker classification, etc.

Two fundamental studies on one-class classification are “Support Vector Domain Description" (SVDD) [22] and "Estimating the Support of a High-Dimensional Distribution" (O-SVM) [20], which represent two different approaches. In [22], Tax and Duin attempted to separate the target class from all other possible data objects by constructing a hypersphere with the minimum radius around the target class. In [20], Schölkopf et al. used a hyperplane that was maximally distant from the origin to separate the target class. An algorithm called Mapping Convergence was proposed in [70]. It computes the boundary between the target class and the negative data by incrementally labeling unlabeled data. Manevitz and Yousef proposed different versions of a support vector machines (SVM) method for one-class classification in the context of information retrieval [73]. They reported the accuracy of the algorithm on the Reuters dataset. In [74], Zhu et al. presented a weighted one-class support vector machine to eliminate the sensitivity of one-class SVM classification to noise. In this algorithm, the weights are determined using a k-nearest neighbors approach, and lower weights are assigned to noise. They tested their algorithm on real datasets from the UCI Machine Learning Dataset Repository [75] and web problems. Hao incorporated the concept of fuzzy set theory into one-class SVM classification in [76]. In [68], Erfani et al. addressed the curse of the dimensionality in anomaly detection. They presented a hybrid model in which deep belief networks are trained to extract features and a one-class SVM is trained from the features learned by the deep belief networks. Campbell and Bennett proposed a simpler kernel method for support estimation based on linear programming (LP) for novelty detection [66]. They presented test results obtained on medical and fault detection datasets. Khan and Madden presented a detailed survey in [77].

In this section, a novel one-class classification algorithm, namely, one-class polyhedral conic functions (O-PCF) is developed. PCFs are the basis of many multi-class classification algorithms [30, 31, 51] and PCF-based classifiers have been applied to many real-life problems, e.g., data reduction [51], arrhythmia classification [52], visual object detection [78] and gesture recognition [53]. An investigation of the use of PCF-based classifiers for one-class classification and the corresponding novel O-PCF algorithm based on mathematical programming model are our main contributions.

The remainder of this section is organized as follows. The proposed one-class

PCF (O-PCF) algorithm is introduced in Subsection 4.1. The results of experiments on real datasets are presented in Subsection 4.2. Finally, conclusions about the O-PCF are provided in Subsection 4.3.

4.1. Proposed Algorithm: One-Class PCF (O-PCF)

In this section of the dissertation, a novel algorithm for the one-class classification problem using PCFs is developed. Unlike in the binary classification task, in one-class classification, there are no available data from the outlier class. Therefore, we have access only to data objects from the target class, set $\mathbf{A}$. Hence, when constructing a classifier, only the target class can be used for training. When this is the case, existing methods based on PCFs can not solve the classification problem.

The O-PCF algorithm requires an LP model to be solved in the training phase for the purpose of finding the best separating function with the parameters $\mathbf{w}$, ξ and γ. If the target class $\mathbf{A}$ is defined as in Equation 4.1, the LP model given in Equation 4.2 is solved in order to obtain a PCF that separates the target class $\mathbf{A}$ from the outliers.

$$\mathbf{A} = \{\mathbf{a}^1, \dots, \mathbf{a}^m\}, \quad \mathbf{a}^i \in R^n, \quad i = 1, \dots, m. \tag{4.1}$$

$$\begin{aligned}
\min \quad & \sum_{i \in I} -(\langle \mathbf{w}, (\mathbf{a}^i - \mathbf{c}) \rangle + \xi \left\| \mathbf{a}^i - \mathbf{c} \right\|_1 - \gamma) + \lambda \sum_{i \in I} z_i \\
\text{s.t.} \quad & \\
& \langle \mathbf{w}, (\mathbf{a}^i - \mathbf{c}) \rangle + \xi \left\| \mathbf{a}^i - \mathbf{c} \right\|_1 - \gamma \le z_i \qquad \forall i \in I \\
& \xi, \gamma \ge 1 \\
& z_i \ge 0 \qquad \forall i \in I
\end{aligned} \tag{4.2}$$

where $\mathbf{a}, \mathbf{w}, \mathbf{c} \in R^n$ and $\xi, \gamma, z, \lambda \in R$. The $\mathbf{c}$ is the centroid of the target set $\mathbf{A}$ and is calculated in advance before the solution of the LP (Please note that the model in Equation 4.2 obtains a PCF for the class $\mathbf{A}$. On the other hand, the O-PCF algorithm obtains k PCFs for the class $\mathbf{A}$ and the centers of these k PCFs are get from the k-means algorithm.). $\mathbf{w}$, ξ and γ is obtained from the solution of the LP.

In this LP model, the variable z_i represents the classification error of data object $\mathbf{a}^i \in \mathbf{A}$. It takes a non-negative value, as shown in Equation 4.4. The left side of the first constraint in the Equation 4.2 is the $g(\mathbf{a}^i)$ which is defined in Equation 4.3. Because any $\mathbf{a}^i$ is from the target class, it is expected that $g(\mathbf{a}^i) \leq 0$. That means, $\mathbf{a}^i$ is classified correctly in the training. In this case, z_i takes 0 value because the objective function tries to minimize the sum of all z_i. z_i takes a positive value as misclassification error when $g(\mathbf{a}^i) > 0$ which means $\mathbf{a}^i$ couldn't be classified correctly with the related PCF (If the classification algorithm generates more than one PCF, it may be separated by an another PCF using different center).

$$g_{(\mathbf{w},\xi,\gamma,\mathbf{c})}(\mathbf{a}^i) = \langle \mathbf{w}, (\mathbf{a}^i - \mathbf{c}) \rangle + \xi \left\| \mathbf{a}^i - \mathbf{c} \right\|_1 - \gamma \tag{4.3}$$

$$z_i = \begin{cases} g(\mathbf{a}^i) & \text{if} \quad g(\mathbf{a}^i) > 0 \\ 0 & \text{if} \quad g(\mathbf{a}^i) \leq 0 \end{cases}, \qquad \forall i \in I \tag{4.4}$$

LP model in Equation 4.2 incorporates two criteria. The first part of the objective function, $\sum_{i \in I} -(\langle \mathbf{w}, (\mathbf{a}^i - \mathbf{c}) \rangle + \xi \left\| \mathbf{a}^i - \mathbf{c} \right\|_1 - \gamma)$, seeks to minimize the size of the decision boundaries, and the second part, $\lambda \sum_{i \in I} z_i$, seeks to minimize the training classification error. The resulting objective function allows the decision boundaries to increase or decrease in size as necessary to achieve both target-class generalization and outlier-class generalization without requiring the presence of data from the outlier class in the training set. The parameter λ controls the trade-off between the classification error and the decision boundary size.

The level set of a PCF is convex; therefore, only convex decision boundaries can be obtained using a PCF (Please see the convex decision boundary which is obtained using a PCF from **Figure 2.7.**). However, in many cases, the target class is non-convex. For this reason, in the O-PCF algorithm, the target class $\mathbf{A}$ is divided into k clusters in advance using the k-means algorithm. The cluster centers are obtained as a result of this step as $\mathbf{c}_r$ where $r = \{1, 2, ..., k\}$. Once the clusters $\mathbf{A}_r$ and the centers $\mathbf{c}_r$ have been obtained, the LP model given in Equation 4.2 is solved for each cluster to obtain the $\mathbf{w}_r$, ξ_r and γ_r parameters of corresponding PCFs. From each LP solution, a PCF is obtained; consequently, the number of PCFs obtained is equal to the number of clusters. Eventually, the final classifier is obtained as the point-wise minimum of these PCFs, as shown in Equation 4.5.

$$G(\mathbf{x}) = \min_{r=1,\ldots,k} \langle \mathbf{w}_r, (\mathbf{x} - \mathbf{c}_r) \rangle + \xi_r \left\| \mathbf{x} - \mathbf{c}_r \right\|_1 - \gamma_r \tag{4.5}$$

The sets **A** and **B** are separable if there exist a $G(x)$ such that

$$G(\mathbf{a}) \leq 0 \quad \forall \mathbf{a} \in \mathbf{A} \tag{4.6}$$

and

$$G(\mathbf{b}) > 0 \quad \forall \mathbf{b} \in \mathbf{B} \tag{4.7}$$

Equation 4.5 provides to label any test data point **x** as target class if the value of the **x** is negative in at least a PCF among k PCFs. In other words, in order to label the **x** as outlier, the value of the **x** should be positive in all k PCFs. Please see **Figure 4.1.**

The O-PCF algorithm is summarized in Algorithm 4.1.

Algorithm 4.1. *The O-PCF Algorithm:*

Input: Data points $\mathbf{a}^i \in \mathbf{A}$, $\lambda \in [0, \infty)$, $k \in [1, \infty)$.
Output: The set of PCFs separating the set **A** from the outliers.

Step 1: *(Clustering).* Divide the set **A** into clusters using the k-means algorithm. Obtain the clusters $\mathbf{A}_r$ and their centers $\mathbf{c}_r$, $r = 1, \ldots, k$, where:

$$\mathbf{A}_r = \{\mathbf{a}^1, \ldots, \mathbf{a}^{l_r}\}, \quad \mathbf{a}^i \in R^n, \quad i \in I_r = \{1, \ldots, l_r\}$$

Step 2: *(Computation of the PCFs).*
for $r = 1$ **to** k **do:**
Solve the following LP model to find g_r for cluster $\mathbf{A}_r$:

$$\min \sum_{i \in I_r} -(\langle \mathbf{w}_r, (\mathbf{a}^i - \mathbf{c}_r) \rangle + \xi_r \left\| \mathbf{a}^i - \mathbf{c}_r \right\|_1 - \gamma_r) + \lambda \sum_{i \in I_r} z_i$$

$$\text{s.t.}$$

$$\begin{aligned} \langle \mathbf{w}_r, (\mathbf{a}^i - \mathbf{c}_r) \rangle + \xi_r \left\| \mathbf{a}^i - \mathbf{c}_r \right\|_1 - \gamma_r &\leq z_i, \quad \forall i \in I_r \\ \xi, \gamma &\geq 1 \\ z_i &\geq 0, \qquad\quad \forall i \in I_r \end{aligned}$$

end for

Step 3: *(Obtaining the final classifier).*

$$G(x) = \min_{r=1,\ldots,k} \langle \mathbf{w}_r, (\mathbf{x} - \mathbf{c}_r) \rangle + \xi_r \left\| \mathbf{x} - \mathbf{c}_r \right\|_1 - \gamma_r$$

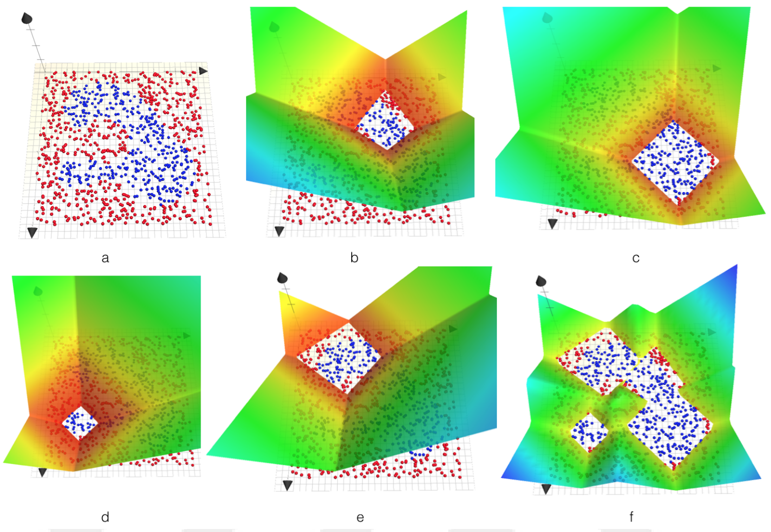

Figure 4.1. *O-PCF example: (a) Data points (blue corresponds to the target class); (b) PCF for cluster 1; (c) PCF for cluster 2; (d) PCF for cluster 3; (e) PCF for cluster 4; (f) Final classifier $G(x)$*

An illustrative example is given in **Figure 4.1.** The target and outlier classes are shown in **Figure 4.1.** - a. The blue and red data points correspond to the target and outlier classes, respectively. One can easily see that the target set is non-convex. In this illustrative example, Algorithm 4.1 is used to find classifiers with a k-means parameter of $k = 4$. Only blue data points are used in the training phase. The PCFs for clusters 1, 2, 3 and 4 are shown in **Figure 4.1.** - b, c, d and e, respectively. The final classifier, **Figure 4.1.** - f, is constructed using Equation 4.5 (Step 3 of Algorithm 4.1) and is the point-wise minimum of the PCFs depicted in **Figure 4.1.** - b, c, d and e.

In order to investigate the affect of λ selection, an illustrative example is created. The target class is shown with black diamond shape and the outlier class is shown with blue circle shape in 1-d space in **Figure 4.2.** The y-axis is the PCF value. The 1-dimensional PCF which is shown with the green dashed line is obtained by solving the O-PCF model when $\lambda = 2$. The PCF which is shown with

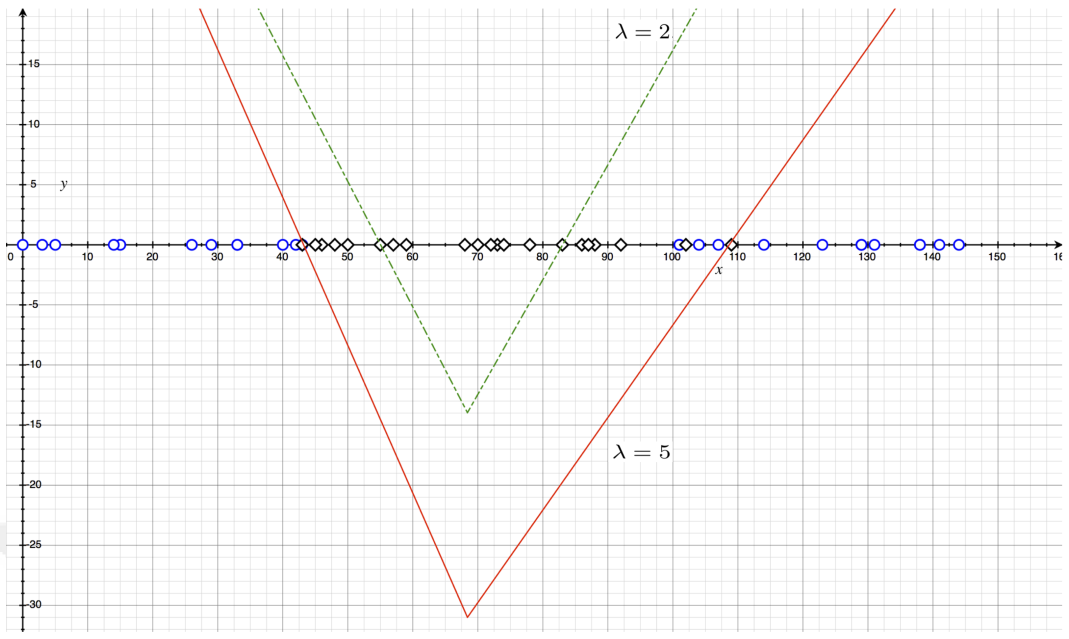

Figure 4.2. *Effect of different λ selections on a 1-dimensional PCF*

the red line is obtained when $\lambda = 5$. The value of k is set to 1 in this example. As described in Algorithm 4.1, the negative side of the PCF functions accept data points as target class. Thus, one can see that larger λ parameter forces O-PCF model to get wider decision boundary which increases the chance of acceptance of the data points as target class.

PAC (Probably Approximately Correct) learning theory was introduced by Valiant in [79]. This theory led to a huge amount of research areas in Computational Learning Theory (COLT). Valiant's learnability model was extended to learning classes of concepts defined by regions in Euclidean space, by Vapnik and Chervonenkis in [80]. According to Vapnik-Chervonenkis theory, the VC dimension (Vapnik-Chervonenkis dimension) is a measure of the capacity of classification functions [81]. The cardinality of the largest set of data points that the classification function can shatter is defined as VC dimension. It was shown that the essential condition for distribution-free learnability is the finiteness of the VC dimension in [80]. In [82], Pestov stated that PAC learnability is equivalent to finite VC dimension for every concept class.

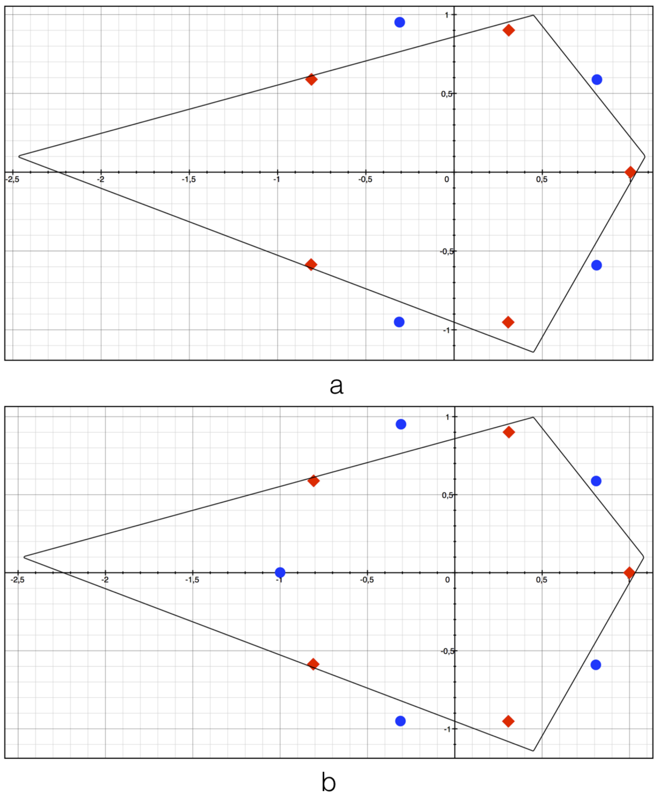

Figure 4.3. *An illustrative example of the VC dimension of a PCF in 2-d space (a) The PCF can shatter 9 data points; (b) The PCF can not shatter 10 data points*

In [83], it is proven that the VC dimension of polygons formed by intersecting at most s half-spaces $2s + 1$. In [29], Gasimov and Ozturk proved the level set of a PCF is an intersection of at most 2^n half-spaces and therefore is a convex polyhedron. Thus, it is clear that the VC dimension of a PCF is finite and $2 \times 2^n + 1$. There is an illustrative example about the VC dimension of a PCF in **Figure 4.3.** One can easily see that a PCF has up to 4 sides in 2-d space. There are two subsets depicted with blue-circle and red-diamond shape. In **Figure 4.3.**-a, there are 9 data points in the dataset. Clearly, the PCF can separate blue-circle and red-diamond subsets. However, when the dataset has 10 points as in **Figure 4.3.**-b, it is impossible to separate this combination of subsets with a PCF.

The VC dimension of a PCF is $2 \times 2^n + 1$ in n-dimensional space. However, the O-PCF algorithm generates k PCFs. Thus, when k is equal to the cardinality of the set $\mathbf{A}$, the O-PCF algorithm can shatter all data points in $\mathbf{A}$. The proof is provided in [29]. Clearly, the VC dimension of the O-PCF algorithm is infinite.

4.2. Experimental Results

In this section, the performance of the O-PCF algorithm is evaluated in comparison with other methods presented in the literature. The O-PCF algorithm is compared with the O-SVM [20], SVDD [22], the 1-nearest-neighbor method (1-NN), the Parzen density estimation method and the local Gaussian approximation method.

The O-PCF algorithm was implemented in Python 2.7. The source code is provided in [84]. The tests were performed on a computer with a 2.5 GHz Intel(R) Core(TM) i7 CPU and 16 GB of RAM. The Gurobi package [55] was used to solve the LP model.

The O-SVM, SVDD, 1-nearest-neighbor, Parzen density estimation and local Gaussian approximation methods were implemented in MATLAB. The LIBSVM package [18] was used to solve the O-SVM and the SVDD methods. The Gaussian RBF kernel was used as kernel function in the O-SVM and the SVDD. The formulations in [85] were used for the 1-nearest-neighbor, Parzen density estimation and local Gaussian approximation methods.

The experimental results are reported as F value with the best parameters obtained via grid search. The λ of the O-PCF was chosen from $\{0.1, 1, 10, 100\}$ and the number of clusters k was chosen from $\{1, 3, 5, 10, 15\}$. The width of the Gaussian RBF kernel σ was chosen from $\{0.1 \times \bar{d}, \bar{d}, 10 \times \bar{d}\}$. $\bar{d}$ is the mean distance between data points in the training set and was calculated as in the Equation 4.9. The parameter ν in O-SVM was chosen from $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. The cost parameter in SVDD was chosen from $\{1/n, 2/n, 5/n, 10/n, 0.1, 0.3, 0.5, 0.7, 0.9\}$, where n is the feature size. The smoothing parameter in the Parzen density estimation was chosen from $\{0.1, 0.5, 1, 2, 3, 5, 10, 100, 1000\}$. The threshold parameters in the 1-nearest-neighbor, Parzen density estimation and local Gaussian approximation were optimized with 25 different values, ranges from minimum value to maximum value in the prediction scores.

$$\mathbf{A} = \{\mathbf{a}^1, \ldots, \mathbf{a}^m\}, \quad \mathbf{a}^i \in R^n, \quad i \in I = \{1, \ldots, m\} \tag{4.8}$$

$$\bar{d} = \frac{1}{m^2} \sum_{i=1}^{m} \sum_{j=1}^{m} \|\mathbf{a}^i - \mathbf{a}^j\|_2 \tag{4.9}$$

In these tests, the O-PCF is compared with other one-class classification algorithms on 14 datasets from the UCI Machine Learning Dataset Repository [75] and LIBSVM [18]. The properties of the datasets are provided in **Table 4.1.** The number of dimensions and the sizes of the training and test sets are given in the second, third and fourth columns, respectively. If there is no independent identically distributed (i.i.d.) test set, this is indicated by a dash in the last column. The numbers in parentheses denote the sizes of each class. To evaluate the performance of one-class classifiers, the datasets are reorganized. For each dataset except Abalone and Wine Quality Red, the largest class in the dataset was considered to be the target class and the rest are considered outliers. The sample sizes for the classes are shown in parentheses in the third column. The first class was used as the target class, and the other was treated as the outlier class in the tests (except Abalone). The Abalone dataset was reorganized into three one-class classification problems. In each problem, one class is considered as the target class and the two others are treated as outliers.

For the four datasets, there is no independent identically distributed (i.i.d.) test set. Thus, half of the target class was used for training. The remainder of the target class and the outliers were used for testing. In the Svmguide 1, Satimage and Web datasets, i.i.d test sets are provided. The target in the training set was used for training. The other classes in the training set and all of the i.i.d test set were used for testing.

In the wine quality problem, the goal is to model wine quality and to detect the few excellent or poor wines. In the Abalone problem, the goal is to predict the sex of an abalone. Svmguide 1 is an astroparticle application. The Satimage (Statlog) dataset consists of the multi-spectral values of pixels in a satellite image and the classification associated with the central pixel in each neighborhood. The aim is to predict this classification, given the multi-spectral values [75]. In a web

Table 4.1. *Description of the datasets*

Dataset	# Dimensions	# Training samples	# Test samples
Wine Quality Red	11	1599 (681, 918)	-
Abalone	8	4177 (1307, 1342, 2649)	-
Svmguide 1	4	3098 (2000, 1089)	4000 (2000, 2000)
Satimage	36	4435 (1072, 3363)	2000 (461, 1549)
web-1a	300	2477 (2405,72)	47272 (45865,1407)
web-2a	300	3470 (3363,107)	46279 (44907,1372)
web-3a	300	4912 (4769,143)	44837 (43501,1336)
web-4a	300	7366 (7150,216)	42383 (41120,1263)
web-5a	300	9888 (9607,281)	39861 (38663,1198)
web-6a	300	17188 (16663,525)	32561 (31607,954)
web-7a	300	34692 (23952,740)	25057 (24318,739)
web-8a	300	49749 (48270,1479)	14951 (14497,454)
Skin	3	245057 (194198,50859)	-
Covertype	54	581012 (297711,283301)	-

problem, the objective is to predict whether a web page belongs to a given category based on the presence of certain keywords. The skin dataset consists randomly sampled B, G, R values from face images of various age groups and races. The task is to predict whether a pixel is a skin or non-skin. The goal of the Covertype dataset is to predict forest cover type from cartographic variables only.

The F value which is the harmonic mean of precision and recall is chosen as a performance measure. Precision, recall and F score are defined in Equation 4.10, Equation 4.11 and Equation 4.12, respectively.

$$Precision = \frac{tp}{tp + fp} \tag{4.10}$$

$$Recall = \frac{tp}{tp + fn} \tag{4.11}$$

$$F = 2 \times \frac{precision \times recall}{precision + recall} \tag{4.12}$$

tp, *fp* and *fn* respectively correspond to true positive, false positive and false negative in Equations 4.10 and 4.11. Larger F score achieves better classification in the tests.

Table 4.2. *Comparison of training times (s)*

Dataset	**O-PCF**	**O-SVM**	**SVDD**
Winequality Red	0.355	0.160	0.027
Abalone FvsIM	0.514	0.034	0.024
Abalone IvsFM	0.578	0.029	0.032
Abalone MvsIF	0.583	0.030	0.024
Svmguide 1	1.992	0.201	0.077
Satimage	3.156	0.103	0.088
web-1a	45.958	1.815	1.628
web-2a	70.033	4.107	3.886
web-3a	90.918	7.278	6.764
web-4a	147.498	49.204	31.769
web-5a	196.736	70.814	65.832
web-6a	415.115	287.771	254.99
web-7a	615.418	872.508	624.698
web-8a	2084.916	3171.187	3041.165
Skin	403.992	176.212	231.228
Covertype	1806.063	5186.955	4975.876

The training times are presented in **Table 4.2.** Since the 1-nearest-neighbor, Parzen density estimation and local Gaussian approximation methods are unsupervised methods, only training times of O-PCF, O-SVM and SVDD are presented. Although training times of O-SVM and SVDD are shorter in small datasets, O-PCF becomes more advantageous when the problem is larger as in Covertype dataset. Total required training times of 16 tests are shown in **Figure 4.4.** Since O-PCF requires the solution of a linear programming (LP) model rather than a quadratic programming (QP) model, O-PCF is faster in total.

The test times are presented in **Table 4.3.** O-PCF, O-SVM and SVDD test times are much shorter than unsupervised methods because the model is learned before testing. Total test times of 16 tests are shown in **Figure 4.5.** The graph is shown in two parts because the magnitude of supervised and unsupervised methods are different. Test time of O-PCF and SVDD is shorter than O-SVM.

In order to show the affect of cluster size (k) on training and test time in O-

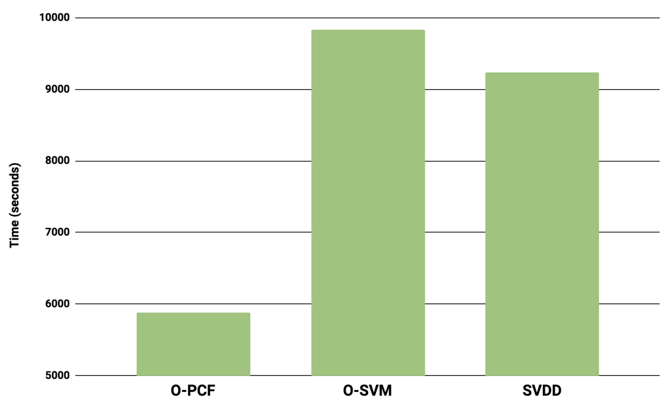

Figure 4.4. *Comparison of total training times (s)*

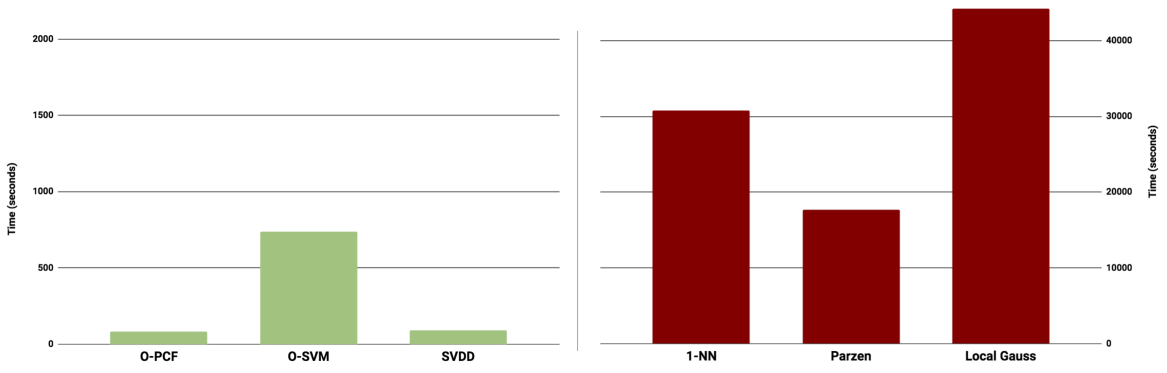

Figure 4.5. *Comparison of total test times (s)*

PCF, **Figure 4.6.** is presented. In this figure, the times of the Covertype dataset are presented when $\lambda = 1$. O-PCF divides the problem into k smaller problems and requires to solve k times linear programming (LP) model as described in Algorithm 4.1. According to **Figure 4.6.**, one can clearly see that it takes longer to solve one large LP than to solve k small LPs. As k increases the training time decreases. On the contrary, the test time increases linearly as k increases because k PCFs are calculated for each test point in the O-PCF. The experimental results in terms of the F values are compared in **Table 4.4.** The best result in each row is shown in bold.

On Wine Quality-Red dataset the best performance is obtained with the O-PCF algorithm. The performances of the other methods are similar. When class F and M are chosen as the target class in the Abalone dataset, the best performances

Table 4.3. *Comparison of test times (s)*

Dataset	O-PCF	O-SVM	SVDD	1-NN	Parzen	Local Gauss
Winequality Red	0.134	0.151	0.004	0.047	0.032	0.233
Abalone FvsIM	0.392	0.011	0.004	0.126	0.121	0.424
Abalone IvsFM	0.245	0.023	0.033	0.129	0.129	0.457
Abalone MvsIF	0.350	0.019	0.004	0.131	0.133	0.430
Svmguide 1	0.101	0.151	0.028	0.223	0.210	0.518
Satimage	0.249	0.085	0.080	0.556	0.352	3.540
web-1a	3.528	6.265	0.259	72.984	32.951	2455.471
web-2a	3.805	48.786	29.419	257.139	59.755	2894.671
web-3a	1.659	11.503	0.524	333.735	85.192	6309.65
web-4a	3.208	17.389	0.753	826.464	220.851	4097.988
web-5a	1.248	22.133	0.676	722.307	296.026	3284.168
web-6a	1.339	31.558	0.719	1183.768	443.195	2900.498
web-7a	1.836	37.575	0.705	1376.156	596.847	3707.631
web-8a	1.221	44.988	3.376	1951.342	945.313	2799.602
Skin	17.011	26.482	55.539	104.297	118.818	80.369
Covertype	45.973	490.087	1.286	23936.965	14836.181	15657.101

are achieved by O-PCF. When class I is considered the target class in the Abalone dataset, the performance of O-SVM is the best. The best performance on the Svmguide 1 problem is obtained using the O-PCF algorithm. The performance of the O-SVM is the best on the Satimage dataset but O-PCF, Parzen density estimation and local Gaussian approximation methods perform poorly. On web problems web-3a and web-6a, O-PCF achieves the best performance. O-PCF tied with 1-NN and Parzen density estimation on web-1a and tied with 1-NN, Parzen density estimation and local Gaussian approximation on web-4a and web-7a. The performance of the SVDD is the best on web-2a and web-5a. 1-NN achieves the best performance on web-8a. The best performance on the Skin problem is obtained using the O-SVM algorithm. O-PCF is superior on the Covertype dataset.

In 8 of the 16 tests, O-PCF outperforms the other methods. In conclusion, it can be said that the O-PCF algorithm outperforms the other methods in certain cases and is generally competitive with them.

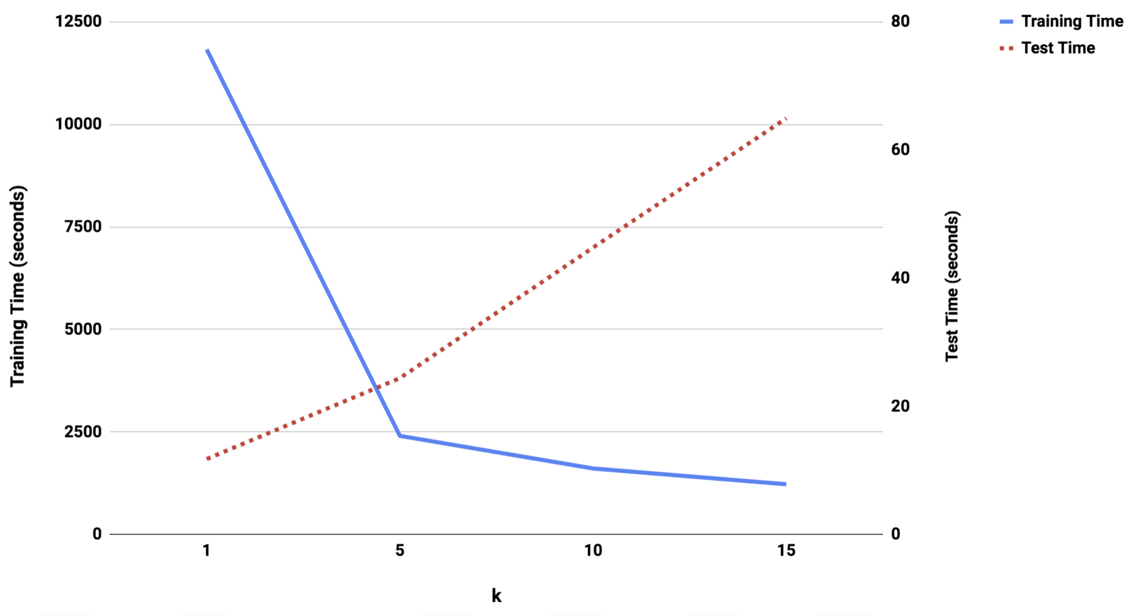

Figure 4.6. *Change of training and test time of the O-PCF according to different k values*

Table 4.4. *Comparison of F scores on the test datasets*

Dataset	**O-PCF**	**O-SVM**	**SVDD**	**1-NN**	**Parzen**	**Local Gauss**
Wine Quality-Red	**0.43621**	0.41816	0.43603	0.42561	0.42984	0.43312
Abalone FvsIM	**0.39358**	0.3825	0.33524	0.31834	0.31274	0.31438
Abalone IvsFM	0.43311	**0.5570**	0.45471	0.3564	0.32128	0.34955
Abalone MvsIF	**0.43098**	0.40193	0.37012	0.36681	0.36581	0.36618
Svmguide 1	**0.92141**	0.91626	0.89373	0.7489	0.56425	0.58106
Satimage	0.39731	**0.83292**	0.83188	0.61268	0.15831	0.15941
web-1a	**0.98413**	0.93138	0.98165	**0.98413**	**0.98413**	0.98411
web-2a	0.98380	0.98472	**0.98473**	0.98416	0.9838	0.98432
web-3a	**0.98337**	0.93202	0.9822	0.98336	0.98328	0.98326
web-4a	**0.98233**	0.93250	0.98214	**0.98233**	**0.98233**	**0.98233**
web-5a	0.98127	0.93031	**0.98131**	0.98123	0.98123	0.98123
web-6a	**0.97715**	0.92876	0.97576	0.97714	0.97714	0.97714
web-7a	**0.97049**	0.92281	0.96621	**0.97049**	**0.97049**	**0.97049**
web-8a	0.93752	0.89586	0.94028	**0.98348**	0.9375	0.96631
Skin	0.83193	**0.88612**	0.84504	0.79124	0.79246	0.78996
Covertype	**0.43621**	0.41816	0.43603	0.42561	0.42984	0.43312

4.3. Conclusions On The O-PCF

In this section, a novel one-class classification algorithm based on PCFs is developed. The classifier is trained using only data from the target class. In previous research, it has been shown that PCF-based classifiers are competitive with other well-known algorithms. With one PCF, only a convex set can be separated. Thus, to obtain non-convex decision boundaries, first the target class is divided into clusters using the k-means algorithm. Then, a PCF for each cluster is found by solving an LP model. Finally, the minimum of these PCFs yields the final classifier. Neither the use of k-means clustering for preprocessing nor the generation of a PCF by solving an LP model is a novel aspect of this section of the dissertation. Instead, the investigation of the use of PCF-based classifiers for one-class classification and the corresponding LP formulation are our contributions to the literature.

Test results obtained using the O-PCF algorithm are presented to compare the O-PCF with other related methods. The test results lead us to conclude that the O-PCF algorithm outperforms the other methods in many cases. In addition to the success of the O-PCF algorithm as seen in the reported tests, it also offers other advantages; e.g., it requires the solution of a linear programming (LP) model rather than a quadratic programming (QP) model, and it does not require any kernel function, as in the case of support vector machines. Furthermore, the classifiers obtained using PCF-based algorithms have very short testing times, which is important for real-time applications. Based on these observations, the O-PCF algorithm is expected to contribute significantly to the machine learning community. In future research, it is planned to apply the O-PCF algorithm to real-life problems, including object detection and healthcare applications.

5. AN ALGORITHM FOR CLUSTERWISE LINEAR REGRESSION IN VERY LARGE DATASETS

Clusterwise linear regression (CLR) is a technique for approximating a data using more than one linear function. It is based on the combination of two machine learning techniques: clustering and regression. There are various applications of CLR problems including applications in market segmentation [86, 87], stock-exchange [88], benefit segmentation [89] and rainfall prediction [90].

The CLR problems can be modeled using different approaches. Mixture models [88, 91] and optimization models such as the mixed integer nonlinear programming [92–94], nonsmooth optimization [95] and nonsmooth DC optimization models [96] are among them. Several algorithms have been developed for solving CLR problems based on these models. They include the extensions of classical clustering algorithms such as k-means [97, 98] and EM [91, 99]. Optimization methods have been extensively applied to solve CLR problems using their different optimization models [93–96, 100, 101]. Most of these algorithms are not applicable or highly inaccurate for solving CLR problems in datasets containing hundreds of thousands and more observations.

CLR is a global optimization problem. It has many local minimizers however the aim is to get global or near-global minimizers. It is imperative to develop special procedures for generating starting points when one applies local search methods for solving such problems. An incremental approach is applied in [95, 96, 100, 101] to find good starting points. Results of numerical experiments presented in these papers demonstrate that incremental CLR algorithms are able to find either global minimizers or solutions with high quality in the sense of overall fit function values. However, these algorithms are very time-consuming in datasets containing hundreds of thousands of observations. They might not produce a solution in a reasonable time.

This section presents an algorithm which is applicable to very large datasets. It is demonstrated that the use of the incremental approach allows one to significantly reduce the computational efforts in nonsmooth optimization based CLR algorithms. The new algorithm is designed using the incremental algorithm introduced in [95] and four procedures for reduction of computational cost. It should be noted that the similar algorithms can be designed using algorithms from [96, 100, 101] as they are also based on the incremental approach.

The proposed algorithm is tested using datasets for regression containing from tens of thousands to millions of observations. We also compare the proposed algorithm with other mainstream regression algorithms in the sense of both accuracy and prediction performance.

The structure of the section is as follows. The CLR, auxiliary CLR problems and the incremental algorithm are described in Subsection 5.1. Procedures for reducing computational effort and a new algorithm for large scale CLR problems are introduced in Subsection 5.2. Computational results are reported in Subsection 5.3. and Subsection 5.4. concludes the section.

5.1. Clusterwise Linear Regression Problem and an Incremental Algorithm for Its Solution

Let $\mathbf{A} = \{(\mathbf{a}^i, b_i) \in R^n \times R : \ i = 1, \dots, m\}$ be a given dataset where $\mathbf{a}^i \in R^n$ is an input and $b_i \in R$ is an output. For linear regression coefficients $(\mathbf{x}, y) \in R^n \times R$ and an observation $(\mathbf{a}^i, b_i) \in \mathbf{A}$ the squared regression error is defined as:

$$E_i(x, y) = \big(\langle \mathbf{x}, \mathbf{a}^i \rangle + y - b_i\big)^2 . \tag{5.1}$$

The notation $E_{ab}(\mathbf{x}, y)$ for an observation $(\mathbf{a}, b) \in \mathbf{A}$ is used. Let $\mathbf{A}^j, \ j = 1, \dots, k$ be clusters such that $\mathbf{A}^j \neq \emptyset, \ j = 1, \dots, k$ and

$$\mathbf{A}^j \bigcap \mathbf{A}^p = \emptyset, \ j, p = 1, \dots, k, \ p \neq j, \quad \mathbf{A} = \bigcup_{j=1}^{k} \mathbf{A}^j .$$

For the dataset $\mathbf{A}$, the CLR aims to find an optimal partition of $\mathbf{A}$ in k clusters and regression coefficients $\{\mathbf{x}^j, y_j\}, \ j = 1, \dots, k$ simultaneously within clusters in order to minimize the overall fit.

Let $\{\mathbf{x}^j, y_j\}$ be linear regression coefficients computed using only data points from the cluster $\mathbf{A}^j, \ j = 1, \dots, k$. We associate a data point with the cluster whose regression error at this point is smallest. Then the overall fit function is:

$$f_k(\mathbf{x}, y) = \sum_{i=1}^{m} \min_{j=1,\dots,k} E_i(\mathbf{x}^j, y_j), \tag{5.2}$$

where $\mathbf{x} = (x^1, \dots, x^k) \in R^{k \times n}$ and $\mathbf{y} \in R^k$. Thus the k-clusterwise linear regression (k-CLR) problem is formulated as follows:

$$\text{minimize} \;\; f_k(\mathbf{x}, \mathbf{y}) \;\; \text{subject to} \;\; \mathbf{x} \in R^{k \times n}, \;\; \mathbf{y} \in R^k. \tag{5.3}$$

In general, the objective function f_k in this problem is nonsmooth nonconvex.

An auxiliary CLR problem is used to find good starting points for each new linear function in the incremental algorithm. Given a solution $(\mathbf{x}^1, y_1, \cdots, \mathbf{x}^{k-1}, y_{k-1})$ to the $(k-1)$-CLR problem (5.3), $k > 1$, denote the regression error of the data point $(\mathbf{a}, b)$ at the $(k-1)$-th iteration by

$$r_{k-1}^{ab} = \min_{j=1 \dots k-1} E_{ab}(\mathbf{x}^j, y_j). \tag{5.4}$$

Then *the k-th auxiliary CLR function* is:

$$\bar{f}_k(\mathbf{u}, v) = \sum_{(\mathbf{a},b) \in \mathbf{A}} \min\{r_{k-1}^{ab}, E_{ab}(\mathbf{u}, v)\}, \;\; \mathbf{u} \in R^n, \;\; v \in R. \tag{5.5}$$

The problem:

$$\textit{minimize } \bar{f}_k(\mathbf{u}, v) \text{ subject to } \mathbf{u} \in R^n, v \in R \tag{5.6}$$

is called *the k-th auxiliary CLR problem.* This problem has $n+1$ variables.

Using the auxiliary CLR problem the following incremental algorithm is introduced in [95] to solve the CLR problem (5.3) with the given $k > 0$ number of linear functions. At each iteration of this algorithm, the Späth algorithm [98] is applied to solve both the auxiliary problem (5.6) and the CLR problem (5.3).

Algorithm 5.1. *Incremental algorithm for solving Problem (5.3):*

Initialization: Compute the linear regression function $(\mathbf{x}^1, y_1) \in R^n \times R$ of the whole set $\mathbf{A}$. Set $l = 1$.

Step 1: *(Computation of the next linear regression function)* Set $l = l + 1$. Let $(\mathbf{x}^1, y_1, \cdots, \mathbf{x}^{l-1}, y_{l-1})$ be the solution to the $(l-1)$-CLR problem. Apply the Späth algorithm to find a solution $(\bar{\mathbf{u}}, \bar{v}) \in R^n \times R$ to the l-th auxiliary CLR problem (5.6).

Step 2: *(Refinement of all linear regression functions)* Select $(\mathbf{x}^1, y_1, \cdots, \mathbf{x}^{l-1}, y_{l-1}, \bar{\mathbf{u}}, \bar{v})$ as a new initial solution, compute their respective clusters and apply the Späth algorithm to solve CLR problem (5.3) for $k = l$.

Step 3: *(Stopping criterion)* If $l = k$, then stop. Otherwise go to Step 1.

In Algorithm 5.1, Steps 1 and 2 are the most important steps. In Step 1, Problem (5.6) is solved to find initial solutions for the l-th linear regression function. In Step 2, the Späth algorithm starting from the solution found in Step 1 is applied to solve Problem (5.3) for $k = l$. The accuracy of Algorithm 5.1 heavily depends on these two steps, however Step 1 is the most important as it finds starting points.

Problem (5.6) is a global optimization problems and it is imperative to use many starting points to find its global or near global solutions when one applies a local method, like the Späth algorithm. A procedure for finding such starting points is introduced in [95]. At each iteration of the incremental algorithm, this procedure constructs starting points using results from the previous iteration. Assume that $\mathbf{A}^1, \ldots, \mathbf{A}^{l-1}$ are clusters found at the $(l-1)$-st iteration $(l > 1)$ of the incremental algorithm. Then this procedure proceeds as follows.

1. Take the cluster $\mathbf{A}^p$, $p = 1, \ldots, l-1$ and any point $(\mathbf{a}, b) \in \mathbf{A}^p$. If $\langle \mathbf{x}^j, a\rangle - y_j = 0$ then remove the point $(\mathbf{a}, b)$ from the list and choose the next point from $\mathbf{A}^p$.

2. Construct a hyperplane $(\mathbf{x}^{ab}, y_{ab}) \in R^n \times R$ passing through the point $(\mathbf{a}, b)$ and parallel to the hyperplane $(\mathbf{x}^j, y_j)$.

3. Compute a cluster around the hyperplane $(\mathbf{x}^{ab}, y_{ab})$ and update this hyperplane using points only from this cluster.

4. Construct the collection of l linear functions $\big[(\mathbf{x}^1, y_1), \ldots, (\mathbf{x}^{l-1}, y_{l-1}), (\mathbf{x}^{ab}, y_{ab})\big]$ and compute the value f_l^{ab} of the function (5.2).

5. Repeat steps 1-4 until all data points visited.

6. Compute the minimum value $f_{min}^l = \min\limits_{(\mathbf{a},b)\in\mathbf{A}} f_l^{ab}$ and for given $\gamma_1 > 1$ compute the threshold $f_t^l = \gamma_1 f_{min}^l$.

7. Compute the set $\mathbf{S}_{st}^l = \big\{(\mathbf{x}^{ab}, y_{ab}) : \ f_l^{ab} \le f_t^l\big\}$.

Now Steps 2 and 3 can be modified as follows.

- Step 2'. (Computation of the next linear regression function). Set $l = l + 1$. Let $(\mathbf{x}^1, y_1, \cdots, \mathbf{x}^{l-1}, y_{l-1})$ be the solution to the $(l-1)$-CLR problem. Take any $(\mathbf{u}, v) \in \mathbf{S}^l_{st}$ and apply the Späth algorithm starting from this point to find a solution $(\bar{\mathbf{u}}, \bar{v}) \in R^n \times R$ to the l-th auxiliary CLR problem (5.6). Repeat it for each point from the set $\mathbf{S}^l_{st}$ and compute the set $\mathbf{S}_{aux}$ of stationary points of the problem (5.6).

- Step 3'. (Refinement of all linear regression functions) Select any $(\bar{\mathbf{u}}, \bar{v}) \in \mathbf{S}_{aux}$ and construct $(\mathbf{x}^1, y_1, \cdots, \mathbf{x}^{l-1}, y_{l-1}, \bar{\mathbf{u}}, \bar{v})$ as an initial solution and apply the Späth algorithm to solve CLR problem (5.3) for $k = l$. Repeat this procedure for all $(\bar{\mathbf{u}}, \bar{v}) \in \mathbf{S}_{aux}$ and get the set $\mathbf{S}^l_{sol}$ of solutions to the CLR problem (5.3). For any $(\mathbf{x}, y) \in \mathbf{S}^l_{sol}$ compute the value f^l_{xy} of the function (5.2) and find $f^l_{min} = \min\limits_{(\mathbf{x},y)\in\mathbf{S}^l_{sol}} f^l_{xy}$. Find $(\bar{\mathbf{x}}, \bar{y}) \in \mathbf{S}^l_{sol}$ such that $f^l_{\bar{\mathbf{x}}\bar{y}} = f^l_{min}$ and accept as a solution to the CLR problem (5.3) for $k = l$.

5.2. Reduction Procedures

In this subsection four procedures to reduce the complexity of Algorithm 5.1 are introduced. In order to reduce the complexity of this algorithm the following approaches are used:

- Scheme 1: Reduction of the number of observations. Identify close points and remove all of them but one.
- Scheme 2: Reduction of the number of candidate points to be starting solutions for the auxiliary CLR problem.
- Scheme 3: Reduction of the number of local minimizers found by solving the auxiliary CLR problem.

5.2.1. Reduction of the number of observations

Since observations, which are close to each other, have almost the same contributions to the k-CLR function. Therefore, it is important to define proximity of observations. We use the following scheme. Compute the centroid $(\bar{\mathbf{a}}, \bar{b})$ of the set $\mathbf{A}$ in the $(n+1)$-dimensional space:

$$\bar{\mathbf{a}} = \frac{1}{m} \sum_{(\mathbf{a},b)\in\mathbf{A}} \mathbf{a}, \quad \bar{b} = \frac{1}{m} \sum_{(\mathbf{a},b)\in\mathbf{A}} b \tag{5.7}$$

and define the number

$$D_0 = \frac{1}{m} \sum_{(\mathbf{a},b)\in \mathbf{A}} \|(\mathbf{a}, b) - (\bar{\mathbf{a}}, \bar{b})\|^2. \tag{5.8}$$

Let

$$\varepsilon_1 = \hat{\varepsilon}_1 D_0 \tag{5.9}$$

where $\hat{\varepsilon}_1 = 10^{-4}$. If the squared Euclidean distance between two points $(\mathbf{a}^1, b_1)$ and $(\mathbf{a}^2, b_2)$:

$$\|(\mathbf{a}^1, b_1) - (\mathbf{a}^2, b_2)\|^2 \le \varepsilon_1 \tag{5.10}$$

then one of them is removed from the set $\mathbf{A}$. This procedure removes points which are already represented in the dataset. This procedure is illustrated in **Figure 5.1.**

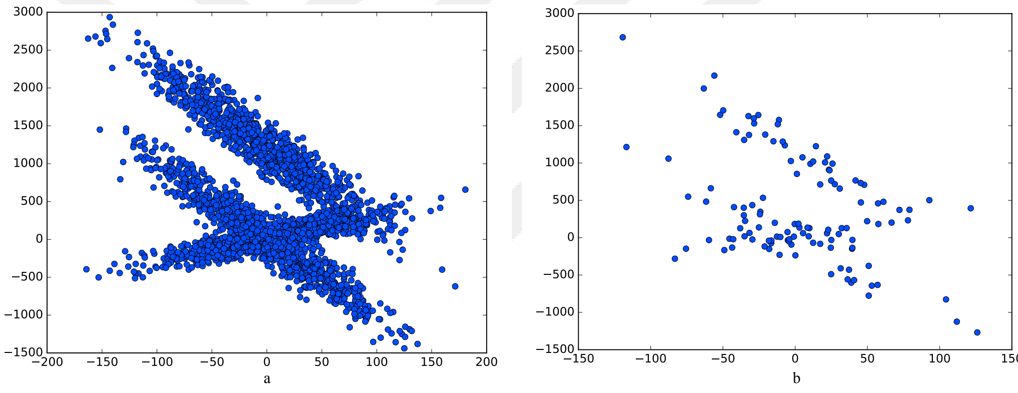

Figure 5.1. *Reduction the number of observations: (a) the whole dataset; (b) the reduced dataset*

5.2.2. Reduction of the number of starting points

In large datasets the cardinality of the set $\mathbf{S}^l_{st}$ in Step 2' of the modified Algorithm 5.1 can become very large. In the worst case, this number is equal to the number of observations in the dataset. However, not all points are good candidates to be used for calculation of the starting linear functions. Since at the l-th iteration of the incremental algorithm in the solving of the auxiliary problem the first $l-1$ linear functions are fixed, all observations whose regression errors are sufficiently small will never change their clusters and linear functions determined using these points will not significantly reduce the value of the objective function of the CLR problem. Therefore, these points can be removed from the list of points for computation of the starting linear functions.

The following procedure is proposed to identify such points. Take any cluster $\mathbf{A}^j,\ j = 1, \dots, l-1$ and corresponding regression coefficients $(\mathbf{x}^j, y_j) \in R^n \times R$ computed at the $(l-1)$-th iteration of the incremental algorithm. For this cluster compute the following number:

$$e_j = \frac{1}{|\mathbf{A}^j|} \sum_{(\mathbf{a},b)\in\mathbf{A}^j} E_{ab}(\mathbf{x}^j, y_j). \tag{5.11}$$

This number represents the average value of the regression error calculated using observations from the cluster $\mathbf{A}^j$. A point $(\mathbf{a}, b) \in \mathbf{A}^j$ is included to the set $\mathbf{S}^l_{st}$ if $E_{ab}(\mathbf{x}^j, y_j) > e_j$, otherwise this point is not considered for calculation of starting linear functions. This procedure is illustrated in **Figure 5.2.**

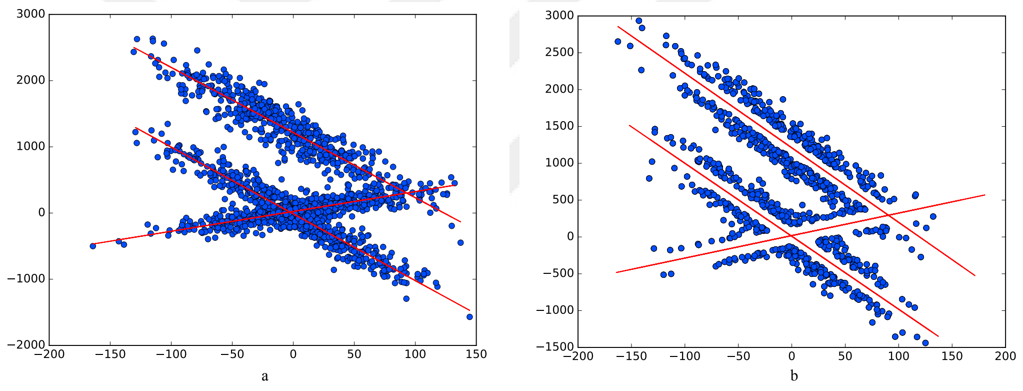

Figure 5.2. *Reduction the number of starting points: (a) the whole dataset with three linear functions; (b) Removing observations with small regression errors*

5.2.3. Reduction of the number of local minimizers of the auxiliary CLR problem

In Step 2' of the modified Algorithm 5.1, one uses the local search Späth algorithm to solve the auxiliary CLR problem starting from points from the set $\mathbf{S}^l_{st}$. Since any local method applied to solve the auxiliary CLR problem may arrive to the same or very close (belonging to the same basin of the objective function) stationary points, it is imperative to identify such stationary points and remove one of them from the set $\mathbf{S}_{aux}$. We propose the following procedure to identify such stationary points. Define the number

$$d_{aux} = \max\left\{\|(\mathbf{x}^1, y_1) - (\mathbf{x}^2, y_2)\| : \ (\mathbf{x}^1, y_1), (\mathbf{x}^2, y_2) \in \mathbf{S}_{aux}\right\} \tag{5.12}$$

which is diameter of the set $\mathbf{S}_{aux}$. Then the following tolerance is defined

$$\varepsilon_3 = d_{aux}\bar{\varepsilon}_3 \tag{5.13}$$

where $\bar{\varepsilon}_3 = 10^{-4}$.

If $\|(\mathbf{x}^1, y_1) - (\mathbf{x}^2, y_2)\| \leq \varepsilon_3$ for $(\mathbf{x}^1, y_1), (\mathbf{x}^2, y_2) \in \mathbf{S}_{aux}$ then the points which provides lower value of the auxiliary CLR function are kept and another one is removed. Using these procedures Algorithm 5.1 is modified as follows.

Algorithm 5.2. *Modified incremental algorithm for solving Problem (5.3):*

Initialization: Apply Procedure 1 to compute the reduced dataset $\bar{\mathbf{A}}$. Compute the linear regression function $(\mathbf{x}^1, y_1) \in R^n \times R$ for the set $\bar{\mathbf{A}}$. Set $l = 1$.

Step 1: *(Computation of the set of candidate linear functions)* Set $l = l + 1$. Let $(\mathbf{x}^1, y_1, \cdots, \mathbf{x}^{l-1}, y_{l-1})$ be the solution to the $(l-1)$-CLR problem. Compute the set $\mathbf{S}^l_{st}$ and apply Procedure 2 to reduce this set.

Step 2: *(Computation of the set of starting linear functions)* Take any $(\mathbf{u}, v) \in \mathbf{S}^l_{st}$ and apply the Späth algorithm starting from this point to find a solution $(\bar{\mathbf{u}}, \bar{v}) \in R^n \times R$ to the l-th auxiliary CLR problem (5.6). Repeat it for each point from the set $\mathbf{S}^l_{st}$ and compute the set $\mathbf{S}_{aux}$ of stationary points of the problem (5.6). Apply Procedure 3 to reduce the set $\mathbf{S}_{aux}$.

Step 3: *(Refinement of all linear regression functions)* Select any $(\bar{\mathbf{u}}, \bar{v}) \in \mathbf{S}_{aux}$ and construct $(\mathbf{x}^1, y_1, \cdots, \mathbf{x}^{l-1}, y_{l-1}, \bar{\mathbf{u}}, \bar{v})$ as an initial solution and apply the Späth algorithm to solve CLR problem (5.3) for $k = l$. Repeat this procedure for all $(\bar{\mathbf{u}}, \bar{v}) \in S_{aux}$ and get the set $\mathbf{S}^l_{sol}$ of solutions to the CLR problem (5.3). For any $(\mathbf{x}, y) \in \mathbf{S}^l_{sol}$ compute the value f^l_{xy} of the function (5.2) and find $f^l_{min} = \min\limits_{(\mathbf{x},y)\in\mathbf{S}^l_{sol}} f^l_{xy}$. Find $(\bar{\mathbf{x}}, \bar{y}) \in \mathbf{S}^l_{sol}$ such that $f^l_{\bar{x}\bar{y}} = f^l_{min}$ and accept as a solution to the CLR problem (5.3) for $k = l$.

Step 4:*(Stopping criterion)* If $l = k$, then stop. Otherwise go to Step 2.

5.3. Numerical Results And Discussions

In this section, the proposed algorithm using large datasets for regression is tested. The brief description of the datasets is given in **Table 5.1.** Detailed information on these datasets can be found in [102] and also in references given after names of each dataset.

Table 5.1. *The brief description of test datasets*

Dataset	**No. of reports**	**No. of input attributes**
Physicochemical Properties of Protein Tertiary Structure [102]	45730	9
BlogFeedback [103]	60021	280
Transcoding Time [104]	68784	18
YearPredictionMSD [102]	515345	90
Buzz in social media-Twitter [105]	583250	77
IHEPC_1	2075259	8
IHEPC_2	2075259	8
IHEPC_3	2075259	8

We present results using the proposed Algorithm 5.2 and compare it with the number of other CLR and regression algorithms: the multi-start Späth algorithm, the expectation-maximization (EM) CLR algorithm, Artificial Neural Networks (ANN), SVM for regression and also k-NN for regression algorithms. Algorithm 5.2, as well as the multi-start Späth algorithm, were implemented in Python 2.7. In order to solve linear regression problems in these algorithms, numpy.linalg.lstsq function is used which returns least-squares solutions. For all other algorithms, their R implementations are used. For example, the EM algorithm is applied using its R implementation Flexmix package and the ANN is applied using its R implementation "nnet". We apply the ANN without any hidden layer (ANN0) and with one hidden layer (ANN-Hid.). We also apply the SVM for regression without any kernel function (SVM-Lin) and, with the radial basis kernel function (SVM-RBF). Numerical experiments were carried out on a PC with processor Intel(R) Core(TM) i7 CPU 2.5 GHz and 16 GB RAM.

In order to present results the following notation is used in all tables in this section:

- k - the number of clusters (or linear regression functions);
- f - the value of the overall fit function f_k;
- t - the CPU time (in seconds);
- V0 - the version of the CLR algorithm without any reduction scheme;
- V1, V2, V3 - the versions of the CLR algorithm with Reduction schemes 1,2 and 3, respectively;
- V12, V13, V23 - the versions of the CLR algorithm with Reduction schemes (1,2), (1,3) and (2,3), respectively;
- V123 - the version of the CLR algorithm with all three reduction schemes.

Results for comparison of different reduction procedures are presented in **Table 5.2.** and this comparison is illustrated in **Figure 5.3.** We can see that the use of different reduction procedures does not lead any significant changes in the accuracy of solutions, however, their use leads to the significant reduction in computational time.

Table 5.2. *Results for comparison of different reduction procedures*

Vers.	***f***	***t***	***f***	***t***	***f***	***t***	***f***	***t***
	$k = 2$		$k = 3$		$k = 4$		$k = 5$	
V0	214398.17	4766.54	106427.24	9314.25	63121.89	13603.20	41742.91	17937.76
V1	214398.17	10.75	106427.24	22.85	63121.69	34.07	41742.93	43.05
V2	214398.17	1235.84	106427.24	2457.52	63121.89	4192.09	41742.91	6324.80
V3	214398.17	4683.23	106427.24	9291.70	63121.89	13307.43	41742.91	9291.70
V12	214398.17	7.26	106427.24	17.56	63121.89	22.32	41743.60	27.66
V13	214398.17	8.70	106427.24	17.31	63121.69	27.47	41742.93	36.01
V23	214398.17	689.80	106427.24	1504.41	63121.89	2844.80	41742.91	4390.04
V123	214398.17	4.42	106427.24	9.27	63121.89	14.13	41743.60	19.13

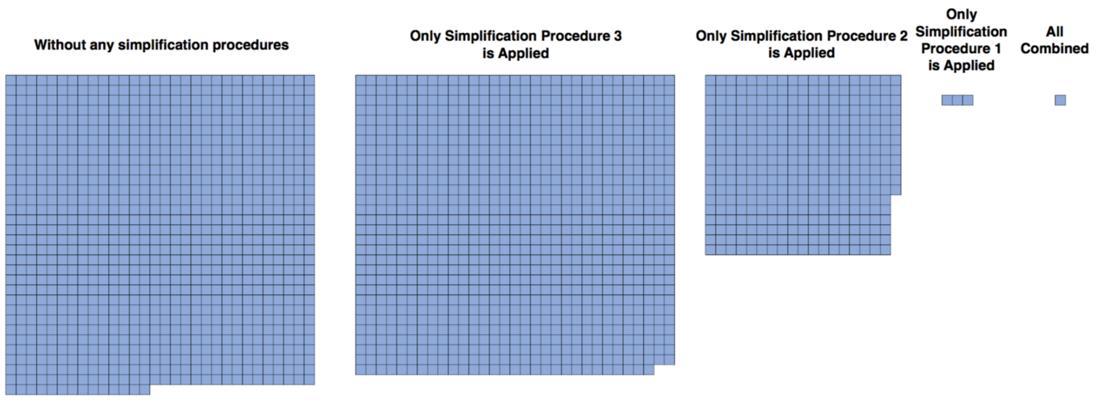
Without any simplification procedures
Only Simplification Procedure 3 is Applied
Only Simplification Procedure 2 is Applied
Only Simplification Procedure 1 is Applied
All Combined

Figure 5.3. *Comparison of different reduction procedures*

5.3.1. Results for approximating of datasets

Results for approximating of whole datasets are presented in **Table 5.3.** In this table average regression errors are presented. We can see that for the small number of clusters ($k = 2, 3$) there is not significant difference in accuracy of these two algorithms. However, we can observe different picture for larger values of the number of clusters. The proposed algorithm is more accurate than the multi-start Späth algorithm.

5.3.2. Prediction results by different algorithms

In this subsection, using different datasets, results on prediction performance of the proposed algorithm are reported and compared with other algorithms. All datasets are divided into the training set that consists of 80% of the whole data and the test set that consists of 20% of the data. We use three performance measures to compare prediction algorithms: RMSE - the Root Mean Squared Error; MAE - Mean Absolute Error and R^2.

Definitions of these measures follow. Assume that $Y_1, \ldots, Y_m,\ m \geq 1$ are observed values for some parameter Y and $F_1, \ldots, F_m$ are their forecast values.

1. The RMSE is defined as

$$RMSE = \left(\frac{1}{m} \sum_{i=1}^{m} (F_i - Y_i)^2 \right)^{1/2}; \tag{5.14}$$

Table 5.3. *Comparison of best average regression errors obtained by the Späth and Proposed algorithms*

k	**Protein**		**Blog**		**Time**		**Year**	
	Späth	Proposed	Späth	Proposed	Späth	Proposed	Späth	Proposed
2	4.688	4.688	195.691	199.489	30.532	30.532	26.074	26.074
3	2.327	2.327	77.888	69.498	14.627	14.627	11.610	11.610
4	1.379	1.380	43.526	38.365	8.875	8.874	7.385	7.387
5	0.911	0.913	26.689	20.651	6.006	6.069	4.869	4.866
6	0.649	0.649	16.894	12.614	4.385	4.013	3.514	3.511
7	0.486	0.486	11.664	6.585	3.284	2.919	2.637	2.641
8	0.377	0.377	6.847	4.227	2.420	2.315	2.055	2.054
9	0.306	0.311	5.582	2.833	2.055	1.834	1.755	1.650
10	0.283	0.250	4.738	2.048	1.700	1.489	1.517	1.353
k	**Twitter**		**IHEPC$_1$**		**IHEPC$_2$**		**IHEPC$_3$**	
	Späth	Proposed	Späth	Proposed	Späth	Proposed	Späth	Proposed
2	11468.025	11194.811	1.139	1.139	2.870	2.870	2.432	2.432
3	6595.273	5220.787	0.510	0.510	1.352	1.252	0.937	1.429
4	3904.329	3243.687	0.274	0.268	0.741	0.741	0.388	0.373
5	2984.147	2410.823	0.185	0.185	0.517	0.515	0.228	0.255
6	2244.647	1814.970	0.133	0.133	0.346	0.346	0.194	0.140
7	1784.184	1440.526	0.099	0.099	0.257	0.259	0.130	0.090
8	1636.819	1113.243	0.070	0.070	0.214	0.182	0.096	0.070
9	1214.427	893.356	0.054	0.056	0.143	0.143	0.080	0.055
10	1070.827	700.438	0.043	0.045	0.109	0.109	0.061	0.042

2. The MAE is:

$$MAE = \frac{1}{m}\sum_{i=1}^{m}|F_i - Y_i|; \tag{5.15}$$

3. R^2 is defined as:

$$CE = 1 - \left[\frac{\sum_{i=1}^{m}(Y_i - F_i)^2}{\sum_{i=1}^{m}(Y_i - \overline{Y})^2}\right]. \tag{5.16}$$

Here $\overline{Y}$ is the mean of observed values.

The small values of the RMSE and the MAE indicates small deviations of the predictions from actual observations. The R^2 can range from $-\infty$ to 1. An efficiency $R^2 = 1$ means a perfect prediction. An efficiency of 0 indicates that the model predictions are as accurate as the mean of the observed data and an efficiency $-\infty < R^2 < 0$ occurs when the observed mean is a better predictor than the model.

In CLR, several linear functions are used to approximate data. Therefore, it is not easy to choose linear functions for prediction. There are different approaches to use CLR for prediction. Such approaches include calculating of weights of linear functions using the number of data points in each cluster. Another approach is to apply k-NN to calculate weights. In [106], the selection of the linear functions for regression is formulated as a classification problem. Following this paper, we apply Random Forests [16] classifier for identifying linear functions for a given input data. The Random Forest classifier generates classification probabilities for this input data which are used to compute weights of linear functions. For the given input data $\mathbf{x}_i$ we have

$$Y_i = \sum_{j=1}^{k} p_i^j Y_i^j, \tag{5.17}$$

where p_i^j is the classification probability of classifying the test point $\mathbf{x}_i$ as jth cluster and Y_i^j is the prediction value of the $\mathbf{x}_i$ on the jth linear regression function and

$$\sum_{j=1}^{k} p_i^j = 1. \tag{5.18}$$

Training times for different algorithms are presented in **Table 5.4.** In this and subsequent tables “F” indicates that an algorithm fails to solve a problem. We can see that the proposed algorithm outperforms SVM-RBF algorithm and it requires comparable CPU time with the ANN-Hidden algorithm (exception is Blog dataset). Other algorithms require significantly less CPU time than the proposed algorithm.

Results for RMSE, MAE and R^2 performance measures on the training are reported in **Table 5.5.**, **Table 5.6.** and **Table 5.7.** These results clearly demonstrate that the proposed algorithm is the best approximating tool among all algorithms.

Results for RMSE, MAE and R^2 performance measures on the test are reported in **Table 5.8.**, **Table 5.9.** and **Table 5.10.** We can see that the proposed algorithm fails to produce any reasonable prediction in two datasets: Blog and Time. According to RMSE results, the proposed algorithm is the best prediction algorithm in five out of eight datasets and it is the second best prediction algo-

Table 5.4. *Training times of prediction algorithms (s)*

Dataset	**ANN0**	**ANN Hidden**	**SVM Lin.**	**SVM RBF**	**Reg. Tree**	**EM**	**Prop.**
Protein	0.37	38	0.68	2732	0.41	11	71
Blog	93	810	24	72733	3	F	36838
Time	0.89	816	2	2307	0.52	F	306
Year	125	5176	205	F	39	349	10954
Twitter	93	15510	841	F	19	1429	14542
IHEPC_1	16	12811	13	F	18	F	4503
IHEPC_2	18	6799	13	F	15	F	8673
IHEPC_3	17	4820	13	F	14	F	15555

Table 5.5. *RMSE results for the training set*

Dataset	**ANN0**	**ANN Hidden**	**SVM Lin.**	**SVM RBF**	**Reg. Tree**	**EM**	**Prop.**
Protein	5.18	4.55	5.23	3.67	5.39	5.39	0.50
Blog	29.40	27.87	29.40	26.48	25.98	F	1.43
Time	10.47	6.29	11.18	7.20	8.44	F	1.22
Year	9.55	8.82	F	F	10.41	9.66	1.16
Twitter	155.10	119.12	161.74	F	211.30	168.00	26.47
IHEPC_1	5.29	4.43	5.29	F	4.89	F	0.21
IHEPC_2	5.14	4.34	5.14	F	4.71	F	0.33
IHEPC_3	6.01	3.96	6.04	F	4.67	F	0.21

rithm in one dataset. According to the MAE performance measure, the proposed algorithm is the best predictor in four out of eight datasets and it is the second best in two datasets. Finally, according to R^2 performance measure, the proposed algorithm is the best prediction algorithm in five out of eight and it is the second best in two datasets.

Table 5.6. *MAE results for the training set*

Dataset	**ANN0**	**ANN Hidden**	**SVM Lin.**	**SVM RBF**	**Reg. Tree**	**EM**	**Prop.**
Protein	4.34	3.58	4.38	2.34	4.47	4.67	0.41
Blog	9.04	10.11	9.08	4.48	7.09	F	0.62
Time	6.49	4.05	7.05	3.81	5.14	F	0.79
Year	6.79	6.14	F	F	7.61	6.86	0.88
Twitter	45.56	41.72	46.15	F	82.20	49.19	10.29
IHEPC$_1$	2.46	1.38	2.49	F	1.40	F	0.03
IHEPC$_2$	2.25	1.53	2.27	F	1.67	F	0.06
IHEPC$_3$	4.60	2.20	4.64	F	2.67	F	0.08

Table 5.7. *R^2 results for the training set*

Dataset	**ANN0**	**ANN Hidden**	**SVM Lin.**	**SVM RBF**	**Reg. Tree**	**EM**	**Prop.**
Protein	0.282	0.446	0.269	0.640	0.223	0.223	0.993
Blog	0.364	0.429	0.364	0.484	0.503	F	0.998
Time	0.577	0.847	0.518	0.800	0.726	F	0.994
Year	0.237	0.349	F	F	0.093	0.220	0.989
Twitter	0.936	0.962	0.930	F	0.881	0.925	0.998
IHEPC$_1$	0.253	0.474	0.252	F	0.360	F	0.999
IHEPC$_2$	0.211	0.437	0.211	F	0.337	F	0.998
IHEPC$_3$	0.487	0.777	0.481	F	0.690	F	0.999

5.4. Conclusions on The Proposed Large Scale CLR Method

In this section, a new algorithm for solving the CLR problems in very large datasets is presented. This algorithm is based on the incremental approach and uses the Späth algorithm for solving CLR and auxiliary CLR problems at each iteration of the incremental algorithm. Using the auxiliary CLR problem a procedure is developed to generate good starting points for solving the CLR problem.

Three different procedures were introduced to reduce the computational cost

Table 5.8. *RMSE results for the test set*

Dataset	ANN0	ANN Hidden	SVM Lin.	SVM RBF	Reg. Tree	kNN	EM	Prop.
Protein	5.20	4.58	5.25	4.09	5.47	5.86	5.42	3.54
Blog	25.45	26.90	25.47	25.45	24.39	24.64	F	F
Time	F	F	12.45	10.20	8.63	7.75	F	F
Year	9.41	8.81	F	F	10.27	10.49	9.53	9.14
Twitter	128.08	124.98	132.09	F	189.10	134.10	135.54	119.27
$IHEPC_1$	4.61	4.06	4.61	F	4.21	4.47	F	3.91
$IHEPC_2$	4.40	3.85	4.39	F	3.93	4.27	4.68	3.75
$IHEPC_3$	5.88	4.11	5.90	F	4.65	4.79	6.80	3.64

Table 5.9. *MAE results for the test set*

Dataset	ANN0	ANN Hidden	SVM Lin.	SVM RBF	Reg. Tree	kNN	EM	Prop.
Protein	4.38	3.61	4.41	2.75	4.53	4.85	4.71	2.51
Blog	8.41	7.54	8.45	4.93	6.28	5.72	F	F
Time	F	F	8.44	5.25	5.19	4.59	F	F
Year	6.72	6.14	F	F	7.54	7.70	6.80	6.50
Twitter	45.01	41.73	46.03	F	82.43	42.79	49.08	42.00
$IHEPC_1$	2.03	1.37	2.07	F	1.03	0.98	F	0.97
$IHEPC_2$	1.88	1.25	1.86	F	1.31	1.32	1.18	1.23
$IHEPC_3$	4.94	2.47	4.57	F	2.75	2.59	4.34	1.90

of solving the CLR and auxiliary CLR problems. Using the first procedure, one identifies observations which have the same or very similar contributions to the overall fit function. The second procedure aims to reduce the number of starting points for solving the auxiliary CLR problem by identifying candidate starting points whose usage will not lead to any significant reduction in the value of the overall fit function. Finally, the third procedure removes stationary points of the auxiliary CLR problem which are close to each other keeping only one of them.

The new CLR algorithm is designed using these three procedures. It is im-

Table 5.10. *R^2 results for the test set*

Dataset	**ANN0**	**ANN Hidden**	**SVM Lin.**	**SVM RBF**	**Reg. Tree**	**kNN**	**EM**	**Prop.**
Protein	0.285	0.446	0.272	0.558	0.210	0.093	0.224	0.669
Blog	0.303	0.222	0.303	0.304	0.361	0.347	F	F
Time	F	F	0.281	0.518	0.655	0.721	F	F
Year	0.237	0.331	F	F	0.090	0.051	0.217	0.280
Twitter	0.947	0.951	0.945	F	0.888	0.944	0.943	0.955
IHEPC_1	0.279	0.440	0.278	F	0.399	0.324	F	0.482
IHEPC_2	0.164	0.359	0.168	F	0.334	0.212	0.054	0.393
IHEPC_3	0.511	0.761	0.508	F	0.694	0.676	0.346	0.813

plemented in Phyton and tested using datasets for regression containing from tens of thousands to several millions of observations and from a few to several hundred input variables. The proposed algorithm was compared with other CLR algorithms such as the EM and the multi start Späth algorithm using two evaluation metrics: accuracy and computational time. This algorithm was compared with some mainstream regression methods using their prediction performance.

Results of numerical experiments demonstrate that the proposed algorithm in small datasets used in our experiments is as accurate as other CLR algorithms, however, it is the only algorithm which can solve CLR problems in very large datasets in a reasonable time. The comparison with other regression algorithms using prediction performance shows that in very large datasets, the proposed algorithm outperforms other algorithms. However, this is not the case for small datasets. Based on numerical results, we can conclude that in very large datasets the proposed algorithm is the only algorithm which is able to solve CLR problems in real-time and it is the best prediction method in such datasets.

6. CONCLUSIONS

In this dissertation, the prediction methods are investigated. Specifically, the classification problem is adressed in Chapters 3. and 4., the regression problem is adressed in Chapter 5.. The proposed methods are developed for large-scale data, and they solve an optimization problem to obtain a solution.

Previous conic functions based algorithms require repeated solutions of LPs which is not convenient for large data sets. The total training time is significantly reduced with the ICF algorithm. The ICF algorithm achieves this by using two approaches. Firstly, it analytically obtains the classifiers without solving the LP sub problem when a cluster within a class is higly pure. If a cluster is not pure enough, it eliminates the data points which do not contribute to the classifier. Thus, the constraints in the LP are reduced by the number of eliminated data points. Finally, the classifiers obtained for all clusters are combined in order to get the final classifier. By this way, 64 % less training time is required than the previous implementation.

In order to solve one class classification problems, a new PCF based algorithm called one-class polyhedral conic functions algorithm is developed. One-class classification algorithms are used for detecting outliers/novelties. The O-PCF classifier which is the combination of k classifiers is trained using only data from the target class. It is assumed that no information about the outlier class is present. Such an algorithm can be applied to important real-life problems such as detecting a very rare disease, detecting a machine failure or detecting an accident priorly. In some cases, data points belonging to outlier class can be found easily. However, obtaining enough data points to represent the whole space can be very hard as in the object image detection problem. One-class classification algorithms can also be used in these cases. Developed O-PCF algorithm is compared with other well known one-class classification algorithms in the literature. The O-PCF achieved best results in the 8 of 16 tests. Moreover, its training and test times are too short compared to other algorithms.

The new proposed algorithm for CLR problem solves a nonsmooth optimization model to get the clusterwise linear regressors. The algorithm is designed for large scale problems. In order to apply the algorithm to large scale data sets, four acceleration methods are used. The acceleration methods allow one to significantly reduce CPU time without any significant loss of accuracy. Working with the large data was not possible in prior versions of this algorithm. Additionally, because the linear regression equation to be used for prediction can not be accurately determined among k equations, it is very hard to predict a target value by using clusterwise linear regression. In order to select the correct regression equation, a classification problem has been solved. The test results show that the developed algorithm achieves better results in many cases than the other algorithms in the literature.

It is obvious that the need for better algorithms to solve both classification and regression problems for many years will not decrease. Concepts such as Industry 4.0, internet of things, autonomous vehicles and dark factories are not possible without success in classification and regression area. It is planned to apply these novel algorithms to various real-life problems. We have already defined the studies about two promising areas: plant genetics and sales prediction. Besides these application areas, some theoretical studies are foreseen. The cluster centers and the classifiers can be found simultaneously in the O-PCF and in the ICF algorithms by solving a non-smooth and non-convex optimization problem. Furthermore, the development of new special classification problem solution methods can eliminate the need for LP solutions. Various norms can be used in order to define the PCFs. The PCF based algorithms can be used for feature selection. Selecting the correct regression function is still an open problem in clusterwise linear regression topic. Researchers who want to study in this area can further focus on these problems.

REFERENCES

[1] Aggarwal, C. C. (2015) *Data Mining: The Textbook*. Springer Publishing Company, Incorporated.

[2] Murphy, K. P. (2012) *Machine Learning: A Probabilistic Perspective*. The MIT Press.

[3] Idc digital universe study. Extracting Value from Chaos, sponsored by EMC, June 2011.

[4] Bottou, L. and Vapnik, V. (1992) Local learning algorithms. *Neural Computation*, **4**, 888–900.

[5] Sun, Y., Todorovic, S., and Goodison, S. (2010) Local-learning-based feature selection for high-dimensional data analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **32**, 1610–1626.

[6] Wu, M. and Schlkopf, B. (2007) Transductive classification via local learning regularization. Meila, M. and Shen, X. (eds.), *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, San Juan, Puerto Rico, 21–24 Mar, vol. 2 of *Proceedings of Machine Learning Research*, pp. 628–635, PMLR.

[7] Wu, M. and Schölkopf, B. (2007) A local learning approach for clustering. Schölkopf, B., Platt, J. C., and Hoffman, T. (eds.), *Advances in Neural Information Processing Systems 19*, pp. 1529–1536, MIT Press.

[8] Cimen, E., Ozturk, G., and Gerek, O. N. (2018) Incremental conic functions algorithm for large scale classification problems. *Digital Signal Processing*, **77**, 187 – 194, digital Signal Processing - SoftwareX - Joint Special Issue on Reproducible Research in Signal Processing.

[9] Cimen, E., Ozturk, G., and Gerek, O. N. (2017) ICF: An algorithm for large scale classification with conic functions. *SoftwareX*.

[10] Legendre, A. (1805) *Nouvelles méthodes pour la détermination des orbites des comètes*. Nineteenth Century Collections Online (NCCO): Science, Technology, and Medicine: 1780-1925, F. Didot.

[11] LaPlace, P. S. (1820) *Théorie analytique des probabilités*. Courcier.

[12] Fisher, R. A. (1936) The use of multiple measurements in taxonomic problems.

Annals of Eugenics, **7**, 179–188.

[13] Markov, A. (2006) Extension of the law of large numbers to quantities, depending on each other (1906). reprint. *Journal lectronique d'Histoire des Probabilits et de la Statistique [electronic only]*, **2**, Article 10, 12 p., electronic only–Article 10, 12 p., electronic only.

[14] Rosenblatt, F. (1958) The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, pp. 65–386.

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

[16] Breiman, L. (2001) Random forests. *Machine Learning*, **45**, 5–32.

[17] Cortes, C. and Vapnik, V. (1995) Support-vector networks. *Machine Learning*, **20**, 273–297.

[18] 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.

[19] Bennett, K. P. and Demiriz, A. (1999) Semi-supervised support vector machines. *Proceedings of the 1998 Conference on Advances in Neural Information Processing Systems II*, Cambridge, MA, USA, pp. 368–374, MIT Press.

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

[21] Schölkopf, B., Smola, A. J., Williamson, R. C., and Bartlett, P. L. (2000) New support vector algorithms. *Neural Comput.*, **12**, 1207–1245.

[22] Tax, D. M. and Duin, R. P. (1999) Support vector domain description. *Pattern Recognition Letters*, **20**, 1191 – 1199.

[23] Hastie, T., Tibshirani, R., and Friedman, J. (2009) *Overview of Supervised Learning*, pp. 9–41. Springer New York.

[24] Breiman, L., Friedman, J., Stone, C., and Olshen, R. (1984) *Classification and Regression Trees*. The Wadsworth and Brooks-Cole statistics-probability series, Taylor & Francis.

[25] Quinlan, J. R. (1993) *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc.

[26] Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., and Blake, A. (2011) Real-time human pose recognition in parts from single depth images. *CVPR 2011*, June, pp. 1297–1304.

[27] Fawagreh, K., Gaber, M. M., and Elyan, E. (2014) Random forests: from early developments to recent advancements. *Systems Science & Control Engineering*, **2**, 602–609.

[28] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986) Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chap. Learning Internal Representations by Error Propagation, pp. 318–362, MIT Press.

[29] Gasimov, R. N. and Ozturk, G. (2006) Separation via polyhedral conic functions. *Optimization Methods and Software*, **21**, 527–540.

[30] Ozturk, G. and Ciftci, M. T. (2015) Clustering based polyhedral conic functions algorithm in classification. *Journal of Industrial and Management Optimization*, **11**, 921–932.

[31] Ozturk, G., Bagirov, A. M., and Kasimbeyli, R. (2015) An incremental piecewise linear classifier based on polyhedral conic separation. *Machine Learning*, **101**, 397–413.

[32] Dordinejad, G. G. and evikalp, H. (2017) Cone vertex estimation in polyhedral conic classifiers. *2017 25th Signal Processing and Communications Applications Conference (SIU)*, May, pp. 1–4.

[33] MacQueen, J. (1967) Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, Berkeley, Calif., pp. 281–297, University of California Press.

[34] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977) Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, **39**, 1–38.

[35] Andrew, N. (2017), Mixtures of gaussians and the em algorithm.

[36] Ozturk, G. (2007), A new mathematical programming approach to solve classification problems.

[37] S., R. and P., N. (1995), Artificial intelligence: A modern approach (2nd ed.).

[38] Rosenblatt, F. (1958) The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, pp. 65–386.

[39] Rastogi, R. and Shim, K. (2000) Public: A decision tree classifier that integrates building and pruning. *Data Mining and Knowledge Discovery*, **4**, 315–344.

[40] Gehrke, J., Ramakrishnan, R., and Ganti, V. (2000) Rainforest—a framework for fast decision tree construction of large datasets. *Data Mining and Knowledge Discovery*, **4**, 127–162.

[41] Bennett, K. P. and Mangasarian, O. L. (1992) Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, **1**, 23–34.

[42] Astorino, A. and Gaudioso, M. (2002) Polyhedral separability through successive lp. *Journal of Optimization Theory and Applications*, **112**, 265–293.

[43] Bagirov, A. M. (2005) Max min separability. *Optimization Methods and Software*, **20**, 277–296.

[44] Uney, F. and Turkay, M. (2006) A mixed-integer programming approach to multi-class data classification problem. *European Journal of Operational Research*, **173**, 910 – 920.

[45] A., P. (2012) Support vector machine - a survey. *International Journal of Emerging Technology and Advanced Engineering*, **2**, 82 – 85.

[46] Astorino, A., Fuduli, A., and Gorgone, E. (2008) Non-smoothness in classification problems. *Optimization Methods and Software*, **23**, 675–688.

[47] Astorino, A., Fuduli, A., and Gaudioso, M. (2016) Nonlinear programming for classification problems in machine learning. *AIP Conference Proceedings*, **1776**, 040004.

[48] Astorino, A. and Fuduli, A. (2015) Support vector machine polyhedral separability in semisupervised learning. *Journal of Optimization Theory and Applications*, **164**, 1039–1050.

[49] Astorino, A. and Gaudioso, M. (2005) Ellipsoidal separation for classification problems. *Optimization Methods and Software*, **20**, 267–276.

[50] Huang, X., Suykens, J. A. K., Wang, S., Hornegger, J., and Maier, A. (2017) Classification with truncated l1 distance kernel. *IEEE Transactions on Neural*

Networks and Learning Systems, **PP**, 1–6.

[51] Bagirov, A. M., Ugon, J., Webb, D., Ozturk, G., and Kasimbeyli, R. (2013) A novel piecewise linear classifier based on polyhedral conic and max–min separabilities. *TOP*, **21**, 3–24.

[52] Cimen, E. and Ozturk, G. (2016) Arrhythmia classification via k-means based polyhedral conic functions algorithm. *2016 International Conference on Computational Science and Computational Intelligence*, December, pp. 798–802.

[53] Cimen, E. (2013) *Gesture Recognition with Polyhedral Conic Functions based Classifiers*. Master's thesis, Graduate School of Sciences, Anadolu University.

[54] Alpaydn, E. (2010), Introduction to machine learning.

[55] Gurobi Optimization, I. (2016), Gurobi optimizer reference manual.

[56] Abalone dataset. Https://archive.ics.uci.edu/ml/datasets/Abalone.

[57] Page block classification dataset. Https://archive.ics.uci.edu/ml/datasets /Page+Blocks+Classification.

[58] Statlog (landsat satellite) dataset. Https://archive.ics.uci.edu/ml/datasets /Statlog+(Landsat+Satellite).

[59] Statlog (shuttle) dataset. Https://archive.ics.uci.edu/ml/datasets /Statlog+(Shuttle).

[60] Covertype dataset. Https://archive.ics.uci.edu/ml/datasets/Covertype.

[61] Dozono, H. and Nakakuni, M. (2009) *Analysis of Robustness of Pareto Learning SOM to Variances of Input Vectors*, pp. 836–844. Springer Berlin Heidelberg.

[62] Tax, D. M. (2001), One-class classification.

[63] Moya, M. M., Koch, M. W., and Hostetler, L. D. (1993) One-class classifier networks for target recognition applications. Tech. rep., Sandia National Labs., Albuquerque, NM (United States).

[64] Japkowicz, N. (1999) *Concept-Learning in the absence of counterexamples: an autoassociation-based approach to classification*. Ph.D. thesis, New Brunswick Rutgers, The State University of New Jersey.

[65] Zhang, Y., Meratnia, N., and Havinga, P. (2009) Adaptive and online one-class support vector machine-based outlier detection techniques for wireless sensor networks. *2009 International Conference on Advanced Information*

Networking and Applications Workshops, May, pp. 990–995.

[66] Campbell, C. and Bennett, K. P. (2000) A linear programming approach to novelty detection. *Proceedings of the 13th International Conference on Neural Information Processing Systems*, Cambridge, MA, USA, pp. 374–380, NIPS'00, MIT Press.

[67] Zhang, R., Zhang, S., Muthuraman, S., and Jiang, J. (2007) One class support vector machine for anomaly detection in the communication network performance data. *Proceedings of the 5th Conference on Applied Electromagnetics, Wireless and Optical Communications*, Stevens Point, Wisconsin, USA, pp. 31–37, Electroscience'07, World Scientific and Engineering Academy and Society (WSEAS).

[68] Erfani, S. M., Rajasegarar, S., Karunasekera, S., and Leckie, C. (2016) High-dimensional and large-scale anomaly detection using a linear one-class {SVM} with deep learning. *Pattern Recognition*, **58**, 121 – 134.

[69] Yu, H. (2003) Svmc: Single-class classification with support vector machines. *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, San Francisco, CA, USA, pp. 567–572, IJCAI'03, Morgan Kaufmann Publishers Inc.

[70] Yu, H. (2005) Single-class classification with mapping convergence. *Machine Learning*, **61**, 49–69.

[71] Liu, B., Dai, Y., Li, X., Lee, W. S., and Yu, P. S. (2003) Building text classifiers using positive and unlabeled examples. *Third IEEE International Conference on Data Mining*, pp. 179–186.

[72] Elkan, C. and Noto, K. (2008) Learning classifiers from only positive and unlabeled data. *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, pp. 213–220, KDD '08, ACM.

[73] Manevitz, L. M. and Yousef, M. (2002) One-class svms for document classification. *J. Mach. Learn. Res.*, **2**, 139–154.

[74] Zhu, F., Yang, J., Gao, C., Xu, S., Ye, N., and Yin, T. (2016) A weighted one-class support vector machine. *Neurocomputing*, **189**, 1 – 10.

[75] Lichman, M. (2013), UCI machine learning repository.

[76] Hao, P.-Y. (2008) Fuzzy one-class support vector machines. *Fuzzy Sets and Systems*, **159**, 2317 – 2336.

[77] Khan, S. S. and Madden, M. G. (2013) One-class classification: Taxonomy of study and review of techniques. *CoRR*, **abs/1312.0049**.

[78] Cevikalp, H. and Triggs, B. (2017) Polyhedral conic classifiers for visual object detection and classification. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July.

[79] Valiant, L. G. (1984) A theory of the learnable. *Commun. ACM*, **27**, 1134–1142.

[80] Vapnik, V. and Chervonenkis, A. (1971) On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Its Applications*, **16**, 264–280.

[81] Vapnik, V., Levin, E., and Cun, Y. L. (1994) Measuring the vc-dimension of a learning machine. *Neural Computation*, **6**, 851–876.

[82] Pestov, V. (2011) Pac learnability versus vc dimension: a footnote to a basic result of statistical learning. *CoRR*, **abs/1104.2097**.

[83] Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M. K. (1989) Learnability and the vapnik-chervonenkis dimension. *J. ACM*, **36**, 929–965.

[84] (2017), Github: Python 2.7 code for o-pcf algorithm. Https://github.com/emrecimen/O-PCF-Algorithm.

[85] de Ridder, D., Tax, D., and Duin, R. (1998) An experimental comparison of one-class classification methods. *Proceedings of the 4th Annual Conference of the Advanced School for Computing and Imaging*, Delft.

[86] Andrews, R. L., Brusco, M. J., and Currim, I. S. (2010) Amalgamation of partitions from multiple segmentation bases: A comparison of non-model-based and model-based methods. *European Journal of Operational Research*, **201**, 608 – 618.

[87] Boztug, Y. and Reutterer, T. (2008) A combined approach for segment-specific market basket analysis. *European Journal of Operational Research*, **187**, 294 – 312.

[88] Preda, C. and Saporta, G. (2005) Clusterwise pls regression on a stochastic process. *Computational Statistics & Data Analysis*, **49**, 99 – 108.

[89] Wedel, M. and Kistemaker, C. (1989) Consumer benefit segmentation using clusterwise linear regression. *International Journal of Research in Marketing*, **6**, 45 – 59.

[90] Bagirov, A., Mahmood, A., and Barton, A. (2017) Prediction of monthly rainfall in victoria, australia: Clusterwise linear regression approach. *Atmospheric Research*, **188**, 20 – 29.

[91] DeSarbo, W. S. and Cron, W. L. (1988) A maximum likelihood methodology for clusterwise linear regression. *Journal of Classification*, **5**, 249–282.

[92] Lau, K.-N., Leung, P.-L., and Tse, K.-K. (1999) A mathematical programming approach to clusterwise regression model and its extensions. *European Journal of Operational Research*, **116**, 640 – 652.

[93] Carbonneau, R. A., Caporossi, G., and Hansen, P. (2011) Globally optimal clusterwise regression by mixed logical-quadratic programming. *European Journal of Operational Research*, **212**, 213 – 222.

[94] DeSarbo, W. S., Oliver, R. L., and Rangaswamy, A. (1989) A simulated annealing methodology for clusterwise linear regression. *Psychometrika*, **54**, 707–736.

[95] Bagirov, A. M., Ugon, J., and Mirzayeva, H. (2013) Nonsmooth nonconvex optimization approach to clusterwise linear regression problems. *European Journal of Operational Research*, **229**, 132 – 142.

[96] Bagirov, A. and Ugon, J. (2018) Nonsmooth DC programming approach to clusterwise linear regression: optimality conditions and algorithms. *Optimization Methods and Software*, **33**, 194–219.

[97] Späth, H. (1979) Algorithm 39 clusterwise linear regression. *Computing*, **22**, 367–373.

[98] Späth, H. (1982) A fast algorithm for clusterwise linear regression. *Computing*, **29**, 175–181.

[99] Gaffney, S. and Smyth, P. (1999) Trajectory clustering with mixtures of regression models. *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, pp. 63–72, KDD ’99, ACM.

[100] Bagirov, A., Ugon, J., and Mirzayeva, H. (2015) An algorithm for clusterwise

linear regression based on smoothing techniques. *Optimization Letters*, **9**, 375–390.

[101] Bagirov, A., Ugon, J., and Mirzayeva, H. (2015) Nonsmooth optimization algorithm for solving clusterwise linear regression problems. *Journal of Optimization Theory and Applications*, **164**, 755–780.

[102] Newman, C. B. D. and Merz, C. (1998), UCI repository of machine learning databases.

[103] Buza, K. (2014), Feedback prediction for blogs.

[104] Deneke, T., Haile, H., Lafond, S., and Lilius, J. (2014) Video transcoding time prediction for proactive load balancing. *2014 IEEE International Conference on Multimedia and Expo (ICME)*, July, pp. 1–6.

[105] Kawala, F., Douzal-Chouakria, A., Gaussier, E., and Dimert, E. (2013) Prédictions d'activité dans les réseaux sociaux en ligne. *4ième conférence sur les modèles et l'analyse des réseaux : Approches mathématiques et informatiques*, France, Oct., p. 16.

[106] Gitman, I., Chen, J., Lei, E., and Dubrawski, A. (2018) Novel prediction techniques based on clusterwise linear regression. *CoRR*, **abs/1804.10742**.

RESUME

Name Surname	: Emre ÇİMEN
Foreign Language	: English
Place of Birth and Year	: Eskişehir / 1988
E-Mail	: ecimen@eskisehir.edu.tr

Education:

- 2006–2010, Bachelor's degree, Anadolu University, Electrical and Electronics Engineering, Eskişehir, Turkey.
- 2017–2011, Bachelor's degree, Anadolu University, Industrial Engineering, Eskişehir, Turkey.
- 2011–2013, Master of Science, Anadolu University, Industrial Engineering, Eskişehir, Turkey.
- 2013–2018, PhD, Eskişehir Technical University, Industrial Engineering, Eskişehir, Turkey.

Professional Background:

- 2011–2018, Research Assistant, Anadolu University, Industrial Engineering Department.
- 2013–Present, Research & Development Manager, Incir R&D Ltd.
- 2015, Visiting Researcher, Federation University Australia, School of Science, Engineering and Information Technology.
- 2018–Present, Research Assistant, Eskisehir Technical University, Industrial Engineering Department.

Journal Articles and Selected Proceedings:

- Cimen E., Ozturk G., Gerek O.N., (2018). Incremental conic functions algorithm for large scale classification problems. Elsevier: Digital Signal Processing, Special Issue on Reproducible Research in Signal Processing, Vol.77, p. 187-194.

- Cimen E., Ozturk G., Gerek O.N., (2017). ICF: An algorithm for large scale classification with conic functions. SoftwareX, (In press).
- 2017, Cimen E., Ozturk G., Separation via Weighted-Norm Cones, 4th Conference on Optimization Methods and Software, Havana, Cuba.
- 2016, Cimen E., Ozturk G., Arrhythmia Classification via k-Means based Polyhedral Conic Functions Algorithm. The 2016 International Conference on Computational Science and Computational Intelligence, CSCI'16, Las Vegas, USA.
- 2015, Cimen E., Ozturk G., A PCF based preprocessing for linear SVM. 27th European Conference on Operational Research, Glasgow, Scotland.
- 2014, Cimen E., Ozturk G., An AHP Model to Design Mobile Applications. International Symposium of the Analytic Hierarchy Process., Washington D.C., USA.
- 2013, Cimen E., Ozturk G., Feature Selection for Classification Using Multi-Objective Optimization. 22nd International Conference on Multiple Criteria Decision Making, Malaga, Spain.

Awards:

- 2006–2011, 8 Times Honour and 2 Times High Honour Student, Anadolu University.

Professional Association Memberships:

- Operational Research Society of Turkey (ORST)