OPTIMIZATION APPROACHES FOR CLASSIFICATION AND FEATURE SELECTION USING OVERLAPPING HYPERBOXES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

DERYA AKBULUT

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
INDUSTRIAL ENGINEERING

FEBRUARY 2019

Approval of the thesis:

OPTIMIZATION APPROACHES FOR CLASSIFICATION AND FEATURE SELECTION USING OVERLAPPING HYPERBOXES

submitted by **DERYA AKBULUT** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Industrial Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Yasemin Serin
Head of Department, **Industrial Engineering** _____

Prof. Dr. Nur Evin Özdemirel
Supervisor, **Industrial Engineering, METU** _____

Assoc. Prof. Dr. Cem İyigün
Co-Supervisor, **Industrial Engineering, METU** _____

Examining Committee Members:

Prof. Dr. Sinan Gürel
Industrial Engineering, METU _____

Prof. Dr. Nur Evin Özdemirel
Industrial Engineering, METU _____

Prof. Dr. Murat Caner Testik
Industrial Engineering, Hacettepe University _____

Assist. Prof. Dr. Mustafa Kemal Tural
Industrial Engineering, METU _____

Assist. Prof. Dr. Mustafa Gökçe Baydoğan
Industrial Engineering, Boğaziçi University _____

Date: 15.02.2019

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Derya Akbulut

Signature:

ABSTRACT

OPTIMIZATION APPROACHES FOR CLASSIFICATION AND FEATURE SELECTION USING OVERLAPPING HYPERBOXES

Akbulut, Derya
Doctor of Philosophy, Industrial Engineering
Supervisor: Prof. Dr. Nur Evin Özdemirel
Co-Supervisor: Assoc. Prof. Dr. Cem İyigün

February 2019, 191 pages

In this thesis, an optimization approach is proposed for the binary classification problem. A mixed integer programming (MIP) model formulation is used to generate hyperboxes as classifiers. The hyperboxes are determined by lower and upper bounds on the feature values, and overlapping of hyperboxes is allowed to reach a balance between misclassification and overfitting. For the test phase, distance-based heuristic algorithms are also developed to classify the overlap and uncovered samples that are not classified by the hyperboxes. A matheuristic, namely Hyperbox Classification for Binary classes (HCB), is developed based on the MIP formulation. In each iteration of the HCB algorithm, a fixed number of hyperboxes are generated using the MIP model, and unclassified sample size is reduced by a hyperbox trimming algorithm. Although HCB controls the number of hyperboxes in a greedy manner, it provides an overall hyperbox configuration with no misclassification at the end of the training phase. HCB is extended as HCB-f with the addition of feature selection property. Starting with a single feature, HCB-f inserts features and hyperboxes to the model iteratively. When the algorithm terminates, only the set of inserted features are used for classification, hence they are selected.

Keywords: Classification, feature selection, hyperbox, binary class, data mining

ÖZ

SINIFLANDIRMA VE ÖZELLİK SEÇİMİ İÇİN ÖRTÜŞEBİLEN HİPER KUTU KULLANAN OPTİMİZASYON YAKLAŞIMLARI

Akbulut, Derya
Doktora, Endüstri Mühendisliği
Tez Danışmanı: Prof. Dr. Nur Evin Özdemirel
Ortak Tez Danışmanı: Doç. Dr. Cem İyigün

Şubat 2019, 191 sayfa

Bu tezde, ikili sınıflandırma problemleri için bir optimizasyon yaklaşımı önerilmiştir. Sınıflandırıcı olarak hiper kutu oluşturmak için karma tamsayılı programlama (KTP) model formülasyonu kullanılmıştır. Hiper kutular, özellik değerlerinin alt ve üst sınırlarından oluşur. Kutuların örtüşmesine izin verilerek, hatalı sınıflandırma ile aşırı uyum arasında bir denge oluşturulmaya çalışılmıştır. Test aşaması için, kutuların örtüşen alanlarına düşen veya kutularla kapsanmayan örnekleri sınıflandırmak için mesafeye dayalı sezgisel algoritmalar da geliştirilmiştir. İkili sınıflar için hiper kutu sınıflandırması yöntemi olarak, KTP formülastonuna dayanan bir matematiksel sezgisel (HCB) geliştirilmiştir. HCB algoritmasının her aşamasında, KTP modeli kullanılarak hiper kutu(lar) üretilir ve sınıflandırılmamış örnek sayısı geliştirilen kutu kırpma algoritmasıyla azaltılır. HCB hiper kutu sayısını obur sezgisel yöntemle kontrol etmesine rağmen, öğrenme aşaması sonunda hatasız bir hiper kutu yapılandırması sağlar. HCB algoritmasına özellik seçimi kabiliyetinin de eklenmesiyle, HCB-f geliştirilmiştir. HCB-f algoritması tek bir özellik kullanımıyla başlayarak, modele her yinelemede yeni hiper kutu(lar) ya da yeni bir özellik dahil eder. Algoritma sonlandığında, özellikler seçilmiş olur.

Anahtar Kelimeler: Sınıflandırma, özellik seçimi, hiper kutu, ikili sınıflandırma, veri madenciliği

To my family…

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisors Prof. Dr. Nur Evin Özdemirel and Assoc. Prof. Dr. Cem İyigün for their motivating guidance throughout my Ph.D. study, for their being accessible no matter how busy schedules they had, and for their patience during all stages of my thesis writing.

I am grateful to my thesis monitoring committee members Assist. Prof. Dr. Banu Lokman and Assist. Prof. Dr. Mustafa Gökçe Baydoğan for their valuable advice and suggestions they gave during the meetings in last four years.

I would also like to thank Prof. Dr. Sinan Güler, Prof. Dr. Murat Caner Testik, and Assist. Prof. Dr. Mustafa Kemal Tural for their valuable comments and suggestions that they provided before finalizing my Ph.D study.

I would like to express my deepest thanks to all my colleagues at Cankaya University, who have been sharing my work load for all these years, and always provided the motivational support that I need.

My sincere thanks also goes to my FRIENDs for their invaluable friendships, all the support that they had provided to me, and for their tolerance to my busy times.

Finally, I would like to thank my family for every moment that they spent for me throughout my whole life, and for all the supports that they had provided to me.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

Data mining, by the most general definition, is the effort to extract valuable knowledge from a dataset. In the early 1980's, the term '*data mining'* was initially used for the studies on the analysis of some experimental data, especially for regression analysis, as in the study of Lovell (1982). Today, with the accelerated developments in computing, the amount of data waiting to be transformed into knowledge has extremely increased in almost every field, such as engineering, medicine, finance, and so on. Data mining is now a research area in its own right, which covers the learning techniques for extracting knowledge.

Learning in data mining can mainly be performed as unsupervised or supervised learning. In unsupervised learning, the information is tried to be extracted from the data, without any prior knowledge. For example, in *clustering*, which is an unsupervised learning technique, the problem is partitioning the dataset into groups, without any information on the number or characteristics (labels) of the groups. On the other hand, in supervised learning, prior information on the data labels is available. Here, the aim is to learn the rules/conditions/formulations from the data, which would lead to the given prior information.

Classification is a supervised learning technique in data mining. The dataset in a classification problem consists of samples that are described in terms of features, where each sample belongs to a known class. The aim is learning the relationship between the features and the classes, and then expressing this relationship as a *classifier* in order to predict the class of a newly observed sample.

Traditionally, the classification problem is attacked by heuristic methods of machine learning such as classification trees. Although several optimization approaches have also been developed, optimization techniques of operations research, in particular mathematical programming, have so far remained unexplored in this area. The main

purpose of this study is to develop such new optimization approaches for the classification problem.

Developments in data collection and computer storage result in datasets having a large, sometimes even huge, number of features potentially useful for classification. However, many a time, only a small subset of these features is found to be sufficient to correctly classify the samples in a dataset. This study also aims at developing optimization methods for selecting the features that are needed for classification.

The main objective of the study is to classify new samples correctly. However, extracting the relationships from the data may not be sufficient for developing a good classifier. In some cases, the relationships can be modeled and the model may 100 percent correctly determine the class labels of samples in the given data. However, the same model may misclassify a newly observed sample. This may occur because the developed model is excessively dependent on the given data and not general enough to correctly classify new samples. This situation is called *overfitting*. In order to deal with overfitting, classification is performed in two main phases, namely training and testing. The dataset is divided into two subsets as the *training dataset* and the *testing dataset* (class labels are known in both subsets). In the first phase, classification is performed on the training dataset, and the classifier is extracted. In the second phase, the classifier is applied to the testing dataset. If the classifier performs well on the training dataset, but not so well on the testing dataset, then the model may be overfitting the data. For a good classification method, the classifier must perform well on both training and testing datasets. For this, it should occasionally allow overlapping regions allocated to classes in feature space. Samples that fall into such overlapping regions cannot (and should not) be classified with certainly using the classifier.

In this study, optimization approaches based on dividing the numerical feature space into hyperboxes are proposed for the classification problem. To begin with, based on the training dataset, an initial mixed integer programming (MIP) model, Model-MOB, extracts the information on the minimum and the maximum values of features for each class, thereby defining the hyperboxes. Model-MOB minimizes the weighted sum of the number of misclassified samples, the number of overlap samples, and the number of hyperboxes used for classification. This model not only minimizes the

misclassification of samples, but also tries to keep a balance between misclassification and overfitting. It also facilitates selection of features that play a significant role in determination of the class labels.

Model-MOB is successful in classification of small datasets. However, due to high computational complexity, it cannot be solved to optimality for large datasets within reasonable times. Best solutions reached within limited time result in low classification accuracy. To deal with this problem, an alternative formulation, Model-MO is proposed. The main difference of this model from Model-MOB is that Model-MO minimizes only the number of misclassified samples and the number of overlap samples, by the help of pre-allocation of hyperboxes to the classes. Although Model-MO outperforms Model-MOB, the allocation of hyperboxes is critically important to keep a balance between the classification quality and the run time complexity. Therefore, a matheuristic algorithm is developed, which iteratively uses Model-MO to classify binary class datasets. Analyzing the preliminary results obtained by this matheuristic, it is seen that hyperbox allocation decisions and sample-to-hyperbox assignment constraints have some drawbacks on the results. Thus, for the final version of the matheuristic, namely Hyperbox Classification of Binary classes (HCB), a new model, Model-MOU is proposed and hyperbox allocation decisions are revised. Model-MOU does not enforce a sample to be assigned to a hyperbox, but it minimizes the number of unassigned samples in the objective function.

The main idea of HCB matheuristic is adding new hyperboxes into the model whenever they are needed, instead of introducing a large set of hyperboxes from the beginning. In each iteration, Model-MOU is solved with a free hyperbox for each class to which samples can be assigned. Based on the model results, hyperbox(es) are trimmed such that no misclassified samples are contained in them. Constraints for sample-to-hyperbox assignments and hyperbox boundaries of the trimmed hyperbox(es) found in the current iteration are inserted into the Model-MOU for the next iteration. Also, a new free hyperbox is introduced instead of the trimmed one. Throughout the iterations, the generated constraints restrict the search space whereas the addition of new hyperboxes expands it. In addition, the number of samples waiting

to be assigned decreases. Overall, the complexity of the problem is kept within reasonable limits.

However, the number of samples and number of hyperboxes are not the only factors that create complexity. Although these are controlled, as the number of features increases, HCB matheuristic still fails to classify datasets since Model-MOU cannot be solved within the time limits. Therefore, HCB is extended to HCB-f, which starts with only one feature. The remaining features are added to the model one after another whenever they are needed, based on a feature importance ranking obtained by Model-MOU. This new matheuristic, HCB-f, does not only controls the complexity but also selects important features, since it ends up only with using the features that are needed to classify the samples.

To sum up the contributions of this study, optimization methods of operation research are used for solving the classification problem through a matheuristic while keeping computational complexity under control. Explicit and interpretable classification rules are extracted by the construction of hyperboxes. In addition, a balance between the misclassification and overfitting is maintained since the overlapping hyperboxes are allowed. Contribution of generated hyperboxes (and respective classification rules) to classification accuracy is also measured by their power and purity. Finally, simultaneous feature selection becomes possible with the implementation of HCB-f.

The rest of the report is organized as follows. The scope of the study and detailed problem definition are given in Chapter 2, including the relevant research questions. Chapter 3 overviews the data mining literature on the classification problem. Solution approaches to the problem are categorized, and optimization methods are reviewed with special emphasis on the hyperbox approach. Chapter 4 is allocated to the mathematical modelling formulations Model-MOB and Model-MO for classification of multiclass datasets, along with the heuristic method used to classify overlap and uncovered samples in the test phase. Experimental results of both models are presented, and a comparison of the models is made in this chapter. In Chapter 5, the development process of HCB matheuristic is described. In Chapter 6, classification results with HCB matheuristic and contribution of hyperboxes to classification are reported. Classification performance of HCB is also compared to classification and

regression trees (CART) in this chapter. Chapter 7 is dedicated to HCB-f which is capable of performing feature selection along with classification. The report is finally concluded in Chapter 8 with a discussion on the developed methods and future research directions.

CHAPTER 2

PROBLEM DEFINITION

Consider an assignment problem with *N* workers, each having different skill levels on *M* different skills. There are *K* main tasks to be performed. Each task requires different levels on different skills. Given these skill requirements, one can easily model this combinatorial optimization problem, to assign workers to tasks in the best way. Now consider the same problem from a different perspective. Assume worker-to-task assignments are given and we know the skill levels of the workers. This time the aim is to extract the skill requirements of the tasks, which is not an assignment problem but a classification problem. In data mining terminology, each worker is a sample and their skills are features, whereas tasks correspond to classes. Since there is such a close relationship between the two problems, it is not improbable to model and solve a classification problem using optimization methods.

The main objective of the study is to minimize misclassification of the samples. However, an additional consideration is to keep under control the potential overfitting due to optimization used in the training phase, while minimizing misclassification. As optimization searches for the 'best possible' solution for the training data, the overfitting of the data is expected. For example, assume that the objective of the optimization model is to minimize the misclassification and the training result has no misclassification as illustrated in Figure 1a. In the test phase, when a new sample that belongs to the square class is observed as in Figure 1b, it will be misclassified according to the overfitting solution in Figure 1a.

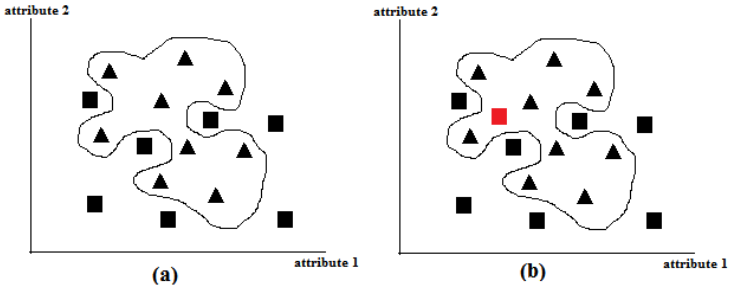


Figure 2.1. Overfitting of the data

As the number of features increases in classification, an additional research question concerning the selection of the features arises. In some datasets, hundreds or even thousands of features may exist. For these datasets, selecting the most relevant features has a major effect on the problem complexity, interpretability and classifier performance. Saeys and Larrañaga (2007) present a review on feature selection techniques applied in the bioinformatics, since the datasets in this field are high dimensional in terms of features. A more general and recent review is also available in the study by Chandrashekar and Sahin (2014). An optimization model which is used in the training phase should also give some useful information on the relevance of the features as part of a classifier.

In addition to that, as in most of the combinatorial optimization problems, there is a time complexity issue for the problem. In fact, the optimization model is to be run only once for the training phase, and waiting for hours to obtain the solution may be acceptable. However, as the number of features and number of samples increase it may be impossible to find even a single feasible solution for the problem due to complexity. Hence, obtaining a valid result in reasonable times is also an objective for this study. The proposed model should simultaneously make it easier to find a feasible solution, while minimizing misclassification and avoiding overfitting.

Finally, unlike the artificial neural network and support vector machine approaches, which work as ‘black boxes’, it is aimed to have high interpretability on the model results to provide ease of understanding the solution and clarity of results.

To sum up, the study proposes optimization approaches for the hyperbox classification problem, which

- minimizes misclassification error
- controls overfitting due to optimization in training phase
- guides feature selection
- has high interpretability in terms of the results and
- can be solved in reasonable time.

The proposed optimization approach is based on dividing the feature space into hyperboxes. In this approach, based on the training data, a mixed integer programming (MIP) model extracts the information on the minimum and the maximum values of features for each class, thereby defining the hyperboxes. For large datasets where solving the MIP model becomes intractable, matheuristic methods are developed. By the use of these matheuristic methods all research objectives listed above are satisfied.

Before the further details on the models and matheuristics, the related literature on the subject is given in the following chapter.

CHAPTER 3

LITERATURE REVIEW

Data mining literature offers two main methodologies as supervised and unsupervised learning depending on the data available in a given problem. For example, classification is a supervised learning problem as class labels are available whereas clustering is an unsupervised learning scheme. Classification, which is the main topic of this thesis, is widely studied in many different fields. Some common applications of classification can be listed as customer target marketing, medical disease diagnosis, multimedia data analysis, biological data analysis, document categorization and filtering, and social network analysis as a trendy topic. For example, it is possible to track the spam mails or to detect a disease using classification methods (Aggarwal, 2014). There are many different methods in the literature for data classification. In Figure 3.1, some of these methods are grouped under three main approaches, as statistical, data driven, and optimization based.

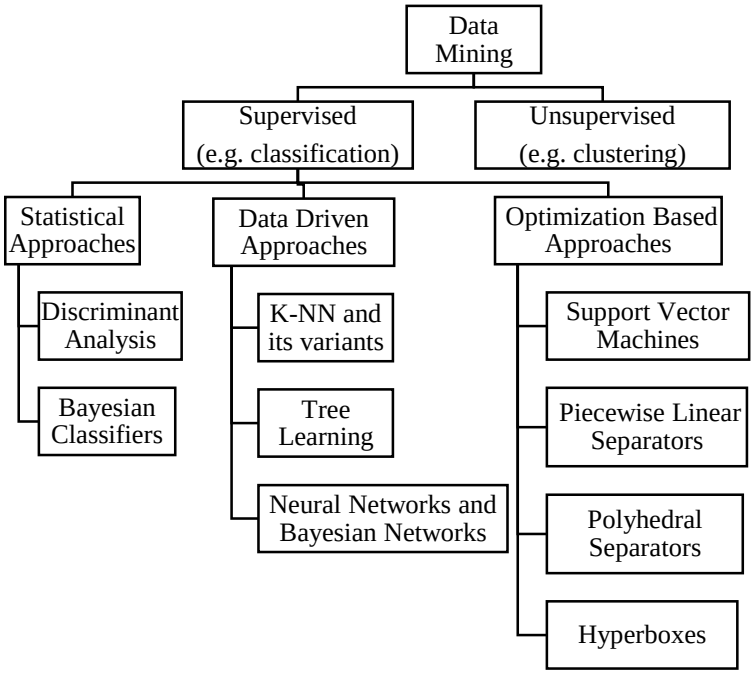


Figure 3.1. Classification Approaches in Data Mining

3.1. Statistical and Data Driven Classification Methods

Statistical methods construct parameterized functions as classifiers. On the other hand, data driven approaches iteratively generate the rules or conditions that classify the data, without using a parametric function (Engels, 1999).

Discriminant analysis is a statistical method that uses a linear combination of features as the classifier to predict the class of a sample. While constructing this linear combination, samples are projected along with the class means and co-variances. It was introduced as a method to separate only two classes (Fisher, 1936). Later, it is extended for multi-class problems by Brown (1947). Briefly, within-class variance and covariance matrix of the dataset are used. There are also some other variants as discriminant correspondence analysis for categorical features (Scholkopf and Muller, 1999), and canonical discriminant analysis for multi-class problems (Glhan, 1968).

Bayesian classification is also a statistical approach that uses a probabilistic model as a classifier. Based on the class labels, it extracts the joint probability distribution of the class and the features. Naïve Bayesian classification is a simplified version of Bayesian classification with the assumption of independence between feature pairs.

In K-nearest neighbor (kNN), which is the earliest data driven approach, each sample is considered as a point in space. The main idea is that members of the same class must be close to each other. To predict the class of a new sample, its *k* nearest neighbors in the space are determined based on a distance metric. Then, the new sample is assigned to the dominant class containing most of these *k* nearest neighbor (Cover and Hart, 1967).

Tree learning uses a tree structure to classify the samples based on their feature levels. The structure can be used for both numerical and categorical features. The root node starts with a feature and samples are sorted based on feature levels. The end nodes are the decision nodes for class label. The main challenge for this method is to construct the near optimal trees. A multi disciplinary survey on the topic is presented by Murthy (1998).

In artificial neural networks (ANN), feature levels of samples are inputs and the class labels are outputs. The input moves one way in the output direction, passing through the nodes in the hidden layer(s) of the network. There are some weights assigned to select the next node. In the training phase, ANNs search for these weights to correctly classify the samples. A review on ANNs is available in Zhang (2000). In Bayesian networks, the nodes represent the features, and instead of weights in ANNs, conditional probabilities are used to pass through the nodes.

Kotsiantis (2007) provide a review of classification techniques including the ones mentioned above. He compares them in terms of accuracy, speed of learning and classification, interpretability, dealing with overfitting, and so on. However, it is concluded that, none of the methods dominate the others yet in terms of these criteria. For example, tree learning algorithms perform quite well in terms of speed, and they are highly interpretable, but their accuracy is in general not so good. On the other hand, ANNs are more accurate, but they are not interpretable due to the lack of transparency in the process.

3.2. Optimization Based Classification Methods

The statistical methods and data driven approaches for classification are heuristics that do not search for the global optimality in terms of misclassification error. On the other hand, there are some optimization based approaches for classification. It is possible to contextualize optimization based classification methods under four main categories, namely support vector machines, piece wise linear separators, polyhedral separators, and hyperbox classifiers.

3.2.1. Support Vector Machines

Support vector machines (SVMs) are introduced in 1992 (Boser et. al., 1992) and widely used in classification studies since then. The main idea of the SVMs is to find a separating hyperplane for the classes. Figure 3.2 illustrates a hyperplane separating two classes, where the circled samples of each class closest to the hyper plane are the support vectors. The objective of the optimization problem is to maximize the distance (*margin*) between support vectors, in order to obtain the best separation among the

classes. The original form of the SVMs is generated for linearly separable classes. However, there are some methods such as using kernel functions to map the data in higher dimensions, and then obtaining separable datasets. A specific survey on SVMs is presented by (Burges, 1998). Several books on the topic are also available, for example Cristianini et al. (2000), and Steinwart and Chrristmann (2008).

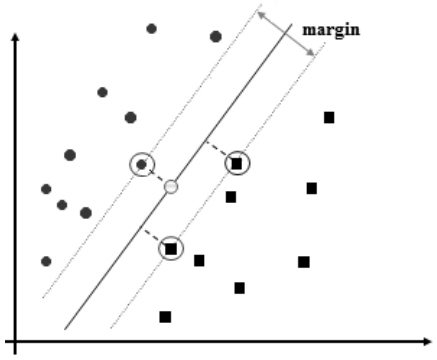


Figure 3.2. SVM Classifier

3.2.2. Piecewise Linear Separators and Polyhedral Separators

For datasets where the convex hulls of the classes intersect, the classes are not linearly separable. Piecewise linear separators are generated to overcome this problem. One of the earliest approaches used to design a piecewise linear separator was proposed by Sklansky and Michelotti (1980). In the approach, some parameters are required to be specified by the user. Later, Park and Sklansky (1989) provided an extension of the study with no requirement of parameter specification. Schulmeister and Wysotzki (1994) proposed decomposing the convex hulls of the classes into subclasses when they intersect. Then, adding linear separators on these subclasses, they obtained piecewise separators. A general framework to construct piecewise linear separators can also be seen in the study of Demyanov (2005).

A different approach to classification problem is proposed by approximating a class using polyhedral sets (Astorino and Gaudioso, 2002). This approach works for two-class datasets. A polyhedral set defines a single class and the region outside defines the other class. Orsenigo and Vercellis (2007) use a MIP model to generate polyhedral classifiers that accurately predict the classes for small sized datasets, by maximizing a weighted sum of accuracy, compactness, and active features.

Min-max separability is introduced by Bagirov (2005). It is a generalized version of linear and polyhedral separability. In this approach, two sets are separated using continuous piecewise linear functions. Gasimov and Ozturk (2006) propose a similar approach, namely polyhedral conic separation. In this approach a class can be defined using more than one polyhedral set, which is the main difference from the polyhedral separation. A combination of polyhedral conic separators and min-max separators is also proposed by Bagirov et al. (2011) as a two stage global optimization algorithm. Visualization of these separators are given in Figure 3.3.

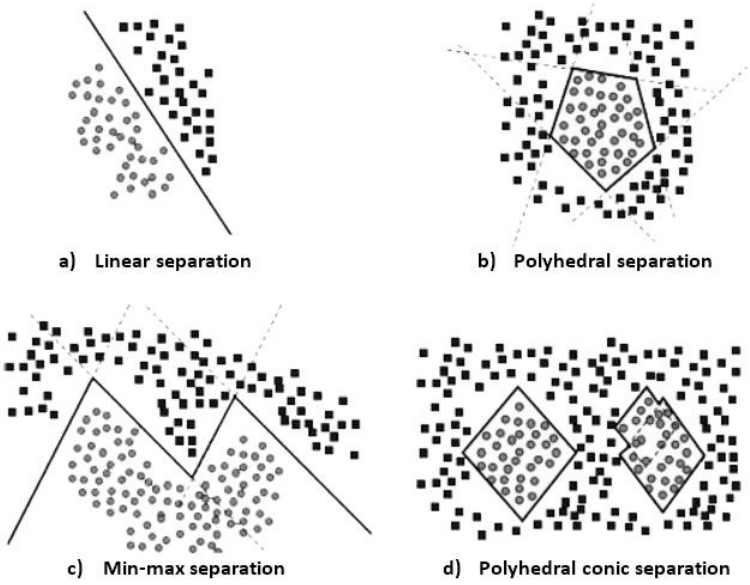


Figure 3.3. Geometric Interpretation of Separation Algorithm, (Gasimov and Ozturk, 2006)

3.2.3. Hyperbox Classification

To the best of our knowledge, hyperbox classifiers were first to be generated by fuzzy min-max neural networks in the study of Simpson (1991). However, hyperbox classification approach using mathematical programming is proposed by Uney and Turkay (2006) for optimization in classification problems. The main difference from the optimization based methods described above is that the separation of the classes is not performed by a function of features. Instead, a hyperbox is defined by the

intersection of intervals each defined on a feature. Each class is then defined by a union of some of these hyperboxes. Figure 3.4 illustrates an example of hyperbox classification in two dimensions using two features *A1* and *A2*.

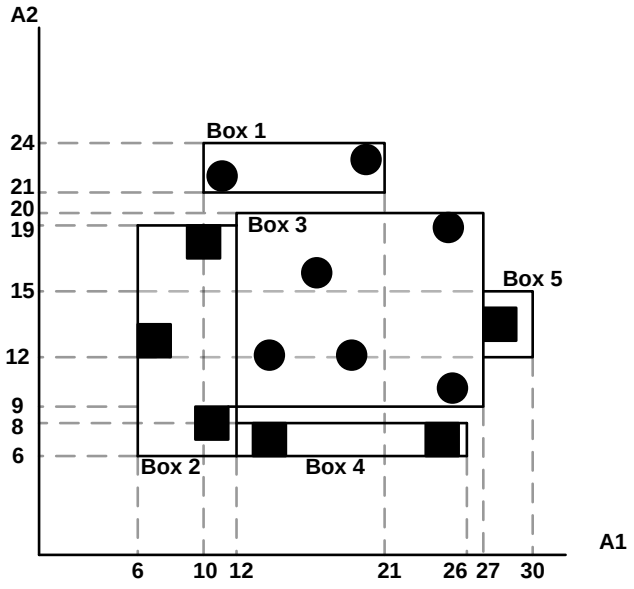


Figure 3.4. Hyperboxes

According to the figure, there are two classes: circles and squares. The regions that correspond to the classes are defined by 5 different boxes. The solution can be interpreted as follows:

Circle class: (Box 1) ∪ (Box 3) where

$$[(10 \leq A1 \leq 21) \text{ and } (21 \leq A2 \leq 24)]$$

or

$$[(12 \leq A1 \leq 27) \text{ and } (9 \leq A2 \leq 20)]$$

Square class: (Box 2) ∪ (Box 4) ∪ (Box 5) where

$$[(6 \leq A1 \leq 12) \text{ and } (6 \leq A2 \leq 19)]$$

or

$$[(12 \leq A1 \leq 26) \text{ and } (6 \leq A2 \leq 8)]$$

or

$$[(27 \leq A1 \leq 30) \text{ and } (12 \leq A2 \leq 15)].$$

In this way, regions are related with the classes in the training phase. For multi-feature datasets, two dimensional boxes are not sufficient to define the region, thus the term hyperbox is used. Uney and Turkay (2006), formulated the problem as a MIP model presented below.

Model UT

$$\min z = \sum_i \sum_k \left(yp1_{ik} + yp2_{ik} \right) + c \sum_l yb_l \qquad (0)$$

$$\sum_l ypb_{il} = 1 \qquad \forall i \qquad (1)$$

$$\sum_k ypc_{ik} = 1 \qquad \forall i \qquad (2)$$

$$\sum_l ypb_{il} = \sum_k ypc_{ik} \qquad \forall i \qquad (3)$$

$$\sum_k ybc_{lk} \le yb_l \qquad \forall l \qquad (4)$$

$$ybc_{lk} - \sum_i ypb_{il} \le 0 \qquad \forall l,k \qquad (5)$$

$$ybc_{lk} - \sum_i ypc_{ik} \le 0 \qquad \forall l,k \qquad (6)$$

$$\sum_n ypbn_{ilmn} - ypbm_{ilm} \le N - 1 \qquad \forall i,l,m \qquad (7)$$

$$\sum_m ypbm_{ilm} - ypb_{il} \le M - 1 \qquad \forall i,l \qquad (8)$$

$$ypc_{ik} - yp1_{ik} \le 0 \qquad \forall i,k \notin D_{ik} \qquad (9)$$

$$ypc_{ik} - yp2_{ik} \ge 1 \qquad \forall i,k \in D_{ik} \qquad (10)$$

$$XD_{lkmn} \le a_{im} ypb_{il} \qquad \forall i,k,l,m,n \mid n = lo \qquad (11)$$

$$XD_{lkmn} \ge a_{im} ypb_{il} \qquad \forall i,k,l,m,n \mid n = up \qquad (12)$$

$$XD_{lkmn} \le Qybc_{lk} \qquad \forall k,l,m,n \qquad (13)$$

$$\sum_k XD_{lkmn} = X_{lmn} \qquad \forall l,m,n \qquad (14)$$

$$ypbn_{ilmn} \geq \frac{1}{Q}\left(\mathrm{X}_{\mathrm{lmn}} - a_{im}\right) \qquad \forall i,l,m,n \mid n = up \tag{15}$$

$$ypbn_{ilmn} \leq \frac{1}{Q}\left(a_{im} - \mathrm{X}_{\mathrm{lmn}}\right) \qquad \forall i,l,m,n \mid n = lo \tag{16}$$

$$XD_{lkmn}, X_{lmn} \geq 0$$

$$yb_l, ybc_{lk}, yp1_{ik}, yp2_{ik}, ypb_{il}, ypbn_{ilmn}, ypc_{ik}, ypbm_{ilm} \in \{0,1\}$$

Decision variables

Binary (takes the value 1 if):

yb_l : Box l is used

ypb_{il} : Point i is in box l

ybc_{lk} : Box l represents class k

ypc_{ik} : Point i is assigned to class k

$ypbn_{ilmn}$: Point i is within bound n with respect to feature m of box l

$ypbm_{ilm}$: Point i is within bounds of feature m of box l

$yp1_{ik}$: Type 1 misclassification of point i

$yp2_{ik}$: Type 2 misclassification of point i

Non negative

XD_{lkmn} : Value of bound n for box l of class k on feature m

X_{lmn} : Value of bound n for box l on feature m

Parameters:

a_{im} : Value of feature m of point i

c: Weight for the total number of hyperboxes

Q: Sufficiently large number

Objective function (0) minimizes the sum of total number of misclassified samples and total number of hyperboxes used as classifier.

The explanations of the constraint sets from (1) through (16) are as follows.

(1) Each point is assigned to one box.

(2) Each box is assigned to one class.

(3) Total number of boxes point *i* is assigned to = total number of classes point *i* is assigned to.

(4) If a box is assigned to a class, then the box is used.

(5) If a box is assigned to a class, then at least one point exists in that box.

(6) If a box is assigned to a class, then at least one point exists in that class.

(7) For a feature, if a point is within lower and upper bounds of box, then the feature of that point is in that box.

(8) If all features of a point are within the same box, then the point is in that box.

(9) If a point is assigned to a different class, then misclassification of type 1 occurs for that point.

(10) If a point is not assigned to its class, then misclassification of type 2 occurs for that point.

(11) Lower bound of a box is less than all feature values of points assigned to that box.

(12) Upper bound of a box is greater than all feature values of points assigned to that box.

(13) Bounds of a box exist if the box is assigned to a class.

(14) Relates two continuous variables.

(15) Point *i* is below upper bound of box *l* for feature *m*, if the related bound is greater than the feature value.

(16) Point *i* is above lower bound of box *l* for feature *m*, if the related bound is less than the feature value.

The details of the model and some experimental results are reported in Uney and Turkay (2006). The model is further discussed and revised in Chapter 4.

Model UT is not a time efficient model for large scale problems. There are some later studies for hyperbox classification that try to resolve this time complexity issue. Xu and Papageorgiou (2009) formulate the same problem without using the box index. Instead, they initially assume that each class is enclosed by one hyperbox. After solving the MIP model, one additional box is allocated to each of the classes yielding misclassified samples, if there are any. Then, the model is solved again. With the addition of the hyperboxes iteratively, the number of misclassified samples is tried to be minimized. The algorithm terminates when two successive iterations yield the same number of misclassified points. The algorithm produces the solutions faster. However, a global optimum may not be reached due to the greedy iterative structure. On the other hand, by the addition of a new hyperbox, the assignment scheme of the preceding iteration is completely changed, which is ineffective in terms of time usage. Maskooki (2013) addressed this issue and proposed an alternative algorithm, which keeps the assignments obtained in an iteration, and adds a new box only for the misclassified samples. In a later study, misclassified samples are weighted in order to enforce the algorithm for classifying them correctly (Yang et. al., 2015).

There are also some box-based approaches developed for binary class datasets. Given a nonhomogeneous box that covers samples of both classes, Eckstein et al. (2002) work on the maximum box problem where they try to find a pure box containing largest number of correctly classified samples. In the study, a branch and bound algorithm is applied to generate the maximum box in terms of the number of points covered. By iteratively generating the maximum boxes, their algorithm ends up with a set of boxes to be used as classifiers. Serafini (2014) focuses on generating boxes for only one of the two classes first, and then the remaining class, instead of simultaneously generating them for both classes. A large set of preliminary boxes is initially given and the minimum number of boxes that cover all the samples of a class are selected by using a MIP model formulation. As an extension, the model is modified to select a family of boxes which, not only consists of the minimum number of boxes, but also has samples that are redundantly covered by as many boxes as possible. This redundancy is used as a measure of certainty of the class label in the study. Despite the implementation of an optimization model, it is not guaranteed to obtain the optimal

number of boxes, because the boxes of the second class are generated sequentially based on the box family of the first class. Hammer et al. (2004) and Spinelli (2014) start with a trivial initial solution of boxes, and then reduce the number of boxes by pruning them.

In this thesis, a mixed integer programming model formulation is proposed to generate overlapping hyperboxes as classifiers. In this model hyperbox-to-class assignments are to be determined along with the sample-to-hyperbox assignments. Then, a revised model is proposed in which the hyperbox-to-class assignments are given as a problem input. This proposed model formulation is then used in a matheuristic, namely HCB. HCB iteratively generates additional hyperboxes as they are needed. These additional boxes are used to classify the unclassified portion of the dataset. Sample-to-hyperbox assignments constraints of the model are relaxed to allow unassigned samples to be classified in later iterations of HCB. Mathematical model formulations are presented in Chapter 4, and HCB matheuristic is described in Chapter 5.

3.3. Feature Selection in Classification

The number of features that define the samples has a major effect on the complexity of classification problems. For the datasets having a large number of features, obtaining accurate results in reasonable times may be intractable. For these cases, selecting a subset of features and using them for classification instead of all features improves the performance of the classification algorithm. However, it is important to select the proper subset of features, without the loss of essential information on the dataset.

Feature selection is a widely studied topic and there are several methods proposed in the literature. Chandrashekar and Sahin (2014) provide a survey on feature selection and mainly focus on filter methods, wrapper methods, and embedded methods. Filter methods are based on the idea of ranking features with respect to their relevance scores, and this filtering process is performed before applying the classification algorithm. Wrapper methods search for the best feature subset that maximizes the

classifier performance. This subset may be defined by the sequential addition of the features one by one, or with the use of heuristic search algorithms. There are also some mathematical model formulations used to determine the best subset by maximizing discrimination power of the selected subset (Bertolazzi et al., 2010). However, wrapper methods are not time efficient since the dataset is used for training with each subset. Embedded methods search for the subset in a similar way as in wrapper methods, but they also consider the feature dependencies. Embedded methods incorporate feature selection and classifier training. Thus, they are more time efficient compared to the wrapper methods. Details of these algorithms can be seen in the surveys of Chandrashekar and Sahin (2014), and Kumar and Vinz (2014).

There are also some specialized reviews on feature selection for high dimensional data (Saeys et al., 2007), (Singh et al., 2016). These reviews also categorize the feature selection approaches under filter, wrapper, and embedded methods. For most of the studies, feature selection method and classification method are independent. For example, Fisher score can be used as a filtering method for feature selection and classification tree or SVM can be used for the classification part. Li et al. (2017) review feature selection on various datasets and present an online repository that provides experimental results of feature selection algorithms under different classification methods.

For the HCB-f algorithm proposed in this thesis, a preprocessing for feature ranking is performed as in filter methods. However, the method to compute the ranks of features is completely different from the known filter methods, since the ranks are obtained by solving a MIP formulation which generates hyperbox classifiers using each feature individually. As in the wrapper or embedded methods, the HCB-f algorithm ends up with a set of features that maximizes the classifier performance. However, HCB-f adds the features into the model in their rank order whenever a new feature is required, instead of generating several subsets of features and then searching for the best performance over these sets. Feature selection method of HCB-f is described in Chapter 7 in detail.

CHAPTER 4

MATHEMATICAL MODEL FORMULATIONS FOR CLASSIFICATION

The general scheme of the solution approach is to use an optimization model that generates lower and upper bounds on each feature to form hyperboxes, and that gives the information of sample-to-hyperbox and hyperbox-to-class assignments. With the resulting hyperboxes, any sample in the training phase will be correctly classified, misclassified, or an *overlap* will occur (which falls into overlapping regions of multiple hyperboxes of different classes). In the test phase, samples are classified with respect to the hyperboxes which they fit into. However, it is not possible to make a certain decision on the class of an overlap sample. In addition to that, there may also be some *uncovered* test samples which do not fit into any hyperbox. For classification of these uncovered and overlap samples, heuristic approaches are developed for use in the test phase.

To begin with, Model UT, which is a hyperbox generating optimization model, is analyzed in detail, and some logical problems are observed concerning the constraints. First of all, in constraint set (3), the left hand side of the equality is equal to 1 due to constraint (1), and the right hand side is also equal to 1 due to constraint (2), which makes constraint (3) redundant. In addition, constraint set (12) implies that, if $ypb_{il} = 1$ for an *i,l* pair then, $XD_{lkmn} \geq a_{im}$ must be satisfied for all *k, m.* This causes constraint (13) to make $ybc_{lk} = 1$ for all *k*. However, this result contradicts the constraint (4), since left hand side sums up to $K > 1 > ypb_{il}$. This leads to infeasibility of the model. To eliminate infeasibility, some modifications are proposed in constraint sets (11) and (12). These modifications are shown in equations (11a) and (12a), respectively.

$$XD_{lkm1} \leq a_{im} + Q\left(1 - ypb_{il}\right) \qquad \forall i,k,l,m \qquad (11a)$$

$$XD_{lkm2} \geq a_{im} - Q\left(1 - ypb_{il}\right) - Q\left(1 - ybc_{lk}\right) \qquad \forall i,k,l,m \qquad (12a)$$

Two additional constraints are still needed after the modifications. One constraint is to prevent obtaining a trivial and nonsense solution where $ybc_{lk} = 0$, for all *l, k*. The other is to ensure that, if point *i* is assigned to class k and point *i* is assigned to box *l*, then box *l* must be assigned to class *k*. These two constraints are shown in equations (17) and (18), respectively.

$$\sum_{k} ybc_{lk} \geq 1 \qquad \forall l \tag{17}$$

$$ybc_{lk} \geq ypb_{il} + ypc_{ik} - 1 \qquad \forall i,l,k \tag{18}$$

The revised model is referred to as UT-R throughout the rest of the thesis. Model UT-R is capable of generating the hyperboxes. However, it still has too many integer variables and constraints, which is not time efficient for large size problems. In order to eliminate these problems, a new mathematical formulation is proposed to be used in the training phase and the results are compared with Model UT-R.

4.1. Mathematical Model Formulation for Hyperbox Generation: Model-MOB

The proposed MIP formulation is based on the idea of hyperboxes. The model aims to find the lower and upper bounds of the hyperboxes on each feature, and gives the information of sample-to-hyperbox, and hyperbox-to-class assignments. For example, samples of class 1 (triangles) are assigned to box 1 and samples of class 2 (rectangles) are assigned to box 2 as in Figure 4.1.

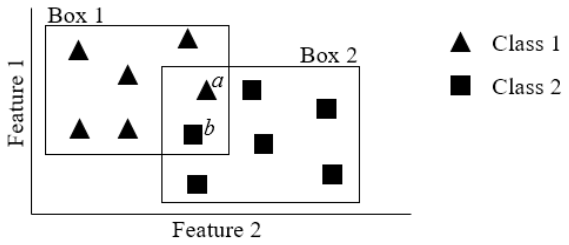


Figure 4.1. Overlapping hyperboxes for classification

If box 1 is assigned to class 1 and box 2 is assigned to class 2, then there are no misclassified samples. But sample *a* also falls in box 2 and sample *b* also falls in box 1 although they are assigned to different hyperboxes, and they are considered as

overlap samples. In the proposed model the number of overlap samples is to be minimized in the objective function, along with the misclassified samples.

Indices, decision variables, parameters, and the model formulation are given below.

Indices

i : sample $\quad i = 1,2,..., N$

l : box $\quad l = 1,2,..., L$

k : class $\quad k = 1,2,..., K$

m : feature $\quad m = 1,2,..., M$

n : bound $\quad n = 1$ (lower), $n = 2$ (upper)

In the training data, the number of samples (N), the number of classes (K), the number of features (M), and the number of bounds (lower and upper) are known. However, the only information about the number of hyperboxes (L) is its minimum, which is equal to K. It is possible to solve the model with a large number of hyperboxes, but using an excessively large L increases the time complexity of the model. In order to deal with this situation, the number L can be tested experimentally at different levels.

Decision variables:

$$Y_{ilk} = \begin{cases} 1, & \text{if sample } i \text{ is assigned to box } l \text{ that belongs to class } k \\ 0, & \text{otherwise} \end{cases}$$

$$BC_{lk} = \begin{cases} 1, & \text{if box } l \text{ is assigned to class } k \\ 0, & \text{otherwise} \end{cases}$$

$$E_{ik} = \begin{cases} 1, & \text{if sample } i \text{ is misclassified in class } k \text{ which is not the class of sample } i \\ 0, & \text{otherwise} \end{cases}$$

$$B_l = \begin{cases} 1, & \text{if box } l \text{ is used} \\ 0, & \text{otherwise} \end{cases}$$

$$O1_{ilm} = \begin{cases} 1, & \text{if sample } i \text{ falls in box } l \text{ wrt. lower bound of feature } m \\ 0, & \text{otherwise} \end{cases}$$

$$O2_{ilm} = \begin{cases} 1, & \text{if sample } i \text{ falls in box } l \text{ wrt. upper bound of feature } m \\ 0, & \text{otherwise} \end{cases}$$

Non negative:

X_{lmn} : Value of bound n of box l for feature m

O_{ilm} : Positive if sample i falls in box l wrt. feauture m

Tl_{il} : Positive if sample i falls in box l, that it is not assigned to

T_i : Number of boxes that sample i falls in, but not assigned to

In this model, sample-to-box-to-class assignment is controlled with binary variable Y_{ilk}. With the use of this variable, it is not necessary to use another binary variable BC_{lk} for box-to-class assignment. However, use of BC_{lk} helps to significantly decrease the number of constraints in the model formulation. O_{ilm} and Tl_{il} are binary variables and T_i is an integer variable by definition. However, defining them as continuous nonnegative variables does not affect their integrality in the solution due to constraint sets (30), (31), and (32) with the minimization of the third term in the objective function.

Parameters

Parameters of the model are derived from the training data, and denoted as follows.

$$d_{ik} = \begin{cases} 1, & \text{if sample } i \text{ belongs to class } k \\ 0, & \text{otherwise} \end{cases}$$

a_{im} : Real value of feature m of sample i

Note that a_{im} value can be a negative due to the nature of dataset. Since X_{lmn} is defined as a positive variable in the model, values of a_{im} parameters can be shifted as $a_{im} = a_{im} + \left| \min_{i,m}(a_{im}) \right| \quad \forall\ i = 1,..., N, m = 1,..., M$ if there are negative a_{im} values in the dataset.

Q : A big number (must be greater than the largest a_{im} value)

ε : A small number

In addition, there are three cost coefficients c_1, c_2, and c_3 used as multipliers in the objective function.

Model-MOB

$$\min z = c_1 \sum_{l=1}^{L} B_l + c_2 \sum_{i=1}^{N} \sum_{k=1}^{K} E_{ik} + c_3 \sum_{i=1}^{N} T_i \tag{19}$$

Subject to

$$\sum_{l=1}^{L} \sum_{k=1}^{K} Y_{ilk} = 1 \qquad \forall i = 1,...,N \tag{20}$$

$$\sum_{l=1}^{L} \sum_{i=1}^{N} Y_{ilk} \geq 1 \qquad \forall k = 1,...,K \tag{21}$$

$$BC_{lk} \geq Y_{ilk} \qquad \forall i,l,k \tag{22}$$

$$B_l \geq BC_{lk} \qquad \forall l,k \tag{23}$$

$$\sum_{k=1}^{K} BC_{lk} \leq 1 \qquad \forall l = 1,...,L \tag{24}$$

$$X_{lm2} \geq a_{im} \sum_{k=1}^{K} Y_{ilk} \qquad \forall i,l,m \tag{25}$$

$$X_{lm1} \leq a_{im} + Q(1 - \sum_{k=1}^{K} Y_{ilk}) \qquad \forall i,l,m \tag{26}$$

$$\sum_{l=1}^{L} Y_{ilk} - E_{ik} \leq d_{ik} \qquad \forall i,k \tag{27}$$

$$a_{im}(1 - \sum_{k=1}^{K} Y_{ilk}) - X_{lm1} + \varepsilon \leq Q(O1_{ilm}) + Q(1 - B_l) + \sum_{k=1}^{K} Q(Y_{ilk}) \qquad \forall i,l,m \tag{28}$$

$$X_{lm2} - a_{im}(1 - \sum_{k=1}^{K} Y_{ilk}) + \varepsilon \leq Q(O2_{ilm}) + Q(1 - B_l) + \sum_{k=1}^{K} Q(Y_{ilk}) \qquad \forall i,l,m \tag{29}$$

$$O_{ilm} + 1 \geq O1_{ilm} + O2_{ilm} \qquad \forall i,l,m \tag{30}$$

$$\sum_{m=1}^{M} O_{ilm} - \sum_{k=1}^{K} Y_{ilk} + 1 \leq M + Tl_{il} \qquad \forall i,l \tag{31}$$

$$T_i = \sum_{l=1}^{L} Tl_{il} \qquad \forall i \tag{32}$$

Objective function (19) minimizes the weighted sum of total number of boxes used, total number of samples misclassified, and total number of boxes each sample falls into. The coefficients c_1, c_2, and c_3 reflect the importance of the respective terms.

Constraint set (20) is to ensure that each sample is assigned exactly to one box and one class. By the use of constraint set (21), there is at least one sample and one box assigned to each class. In set (22), the consistency of box-to-class and sample-to-box-to-class assignments is guaranteed. If sample *i* is assigned to the box *l* of class *k*, then box *l* is assigned to class *k.* Assignment of a box to a class implies the use of that box in constraint set (23). Constraint set (24) states that a box can be assigned to at most one class. All feature values of a sample must be within lower and upper bounds of its assigned hyperbox. This is guaranteed in constraint sets (25) and (26) for upper and lower bounds, respectively. Constraint set (27) computes the number of misclassified samples. If sample *i* is assigned to class *k*, although it does not belong to that class in the training data, then E_{ik} takes the value of 1.

Constraints (20) – (27) ensure the assignment of samples to hyperboxes, hyperboxes to classes, and determine the hyperbox bounds. However, it is possible to see a solution as illustrated in Figure 4.1. This solution does not violate these sets of constraints, and there is no misclassification. The problem here is that samples *a* and *b* fall in the overlapping regions of the two boxes. For a solution with overlapping hyperboxes, the class of a sample that falls into the overlapping region would in fact be undecided, which can be generally encountered in many real-world datasets. Therefore, it would be beneficial to allow such overlaps and detect them.

The model in Uney and Turkay (2006) avoids overlaps by enforcing the model to strictly separate boxes from each other. Hence, each sample must fall in the region of at most one box. Constraint set (8) for this purpose makes sense due to the nature of the problem; however, this is an unnecessarily hard constraint. Moreover, as the size of the problem increases, it becomes intractable and impossible to obtain any feasible solution for hours with non-overlapping hyperboxes. Also, when all the boxes are forced to be separated, the solution becomes strongly data dependent and this may cause overfitting. When the constraint is removed from the model, it terminates in relatively much shorter time but yields a low quality solution with overlapping boxes. This makes the result useless for the test phase of the classification problem.

Constraint sets (28) through (32) in Model-MOB are formulated in order to handle the tradeoff involving the misclassification, overfitting, and time complexity. In this model, the hyperboxes are not forced to be separated completely. Instead variable T_i is used to count the number of extra hyperboxes that a sample can fit in. The model then tries to minimize this. However, for the cases where complete separation of hyperboxes is impossible or too hard, the model can still find a feasible solution. Constraint set (28) is active when box *l* is used but sample *i* is not explicitly assigned to box *l*. $O1_{ilm}$ takes the value of 1 if sample *i* is above the lower bound of box *l* with respect to feature *m*. Set (29) does the same for the upper bound. If $O1_{ilm}$ and $O2_{ilm}$ are both equal to 1, O_{ilm} also becomes 1 due to constraint set (30), which means that sample *i* is also in the region of box *l* with respect to feature *m*. According to constraint set (31), if a sample is in the region of a box in terms of all features although it is not assigned to that box, Tl_{il} takes the value of 1 indicating an overlap. Finally, constraint set (32) counts the total number of additional boxes that the sample can possibly be assigned to as described above.

4.2. Interpretation of Model Results for Classification

The proposed model generates the hyperboxes which classify the samples in the training dataset, minimizing the misclassification and overlap. In practice, the ultimate aim of this model is to use the generated hyperbox bounds for classification of the new samples in the test phase. If a test sample falls into the region of a generated hyperbox, then it is labeled with the class of that hyperbox. However, these bounds are strongly dependent on the training dataset. Thus, some uncovered or overlap samples may occur in the test phase. A distance-based heuristic, namely classification of overlap and uncovered samples (COUS), is developed for the classification of these samples.

4.2.1. Classification of Overlap Samples using COUS

Overlap samples can be observed both in training and testing phases, because the model does not completely restrict the overlap, but only tries to minimize the number

of overlap samples. On the other hand, it is possible to observe an overlap sample in the test phase even if there is no overlap sample in the training phase. Because the model does not restrict the overlap geometrically, but it counts overlaps only if a sample falls into an overlapping region of hyperboxes. As it is mentioned before, any enforcement in the solution approach for the classification of an overlap sample contradicts with the motivation of this study, which is to prevent overfitting. In a real world problem, any overlap sample in the output should be interpreted as a suspicious sample, which does not clearly belong to any class. Actually, it is a member of an additional class with the *undecided* label and it should be examined by subject matter experts. However, overlap samples may also be caused by the deficiencies of the solution approach. Thus, we still try to assign any overlap sample to one of the classes.

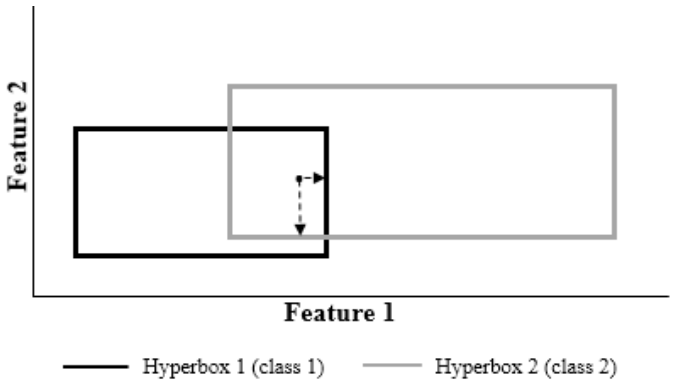


Figure 4.2. Overlap Sample Classification

Our overlap sample assignment in test phase is illustrated in Figure 4.2. In this case, as the sample is closer to the boundary of a hyperbox, it is that much closer to falling outside of that box. Based on this fact, the shortest distance of the sample to the boundary of each hyperbox is computed, and the sample is assigned to the class of the hyperbox having the maximum shortest distance. In the figure, the shortest distances for each hyperbox are shown with dashed arrows. When these distances are compared, the overlap sample belongs to class 2. The pseudo code of the algorithm is given below, where $Shortest_{il}$ is the shortest distance of sample *i* to box *l*, and Max_i is the maximum shortest distance for sample *i*.

For $i \in$ Set of overlap samples
  For $l \in$ Set of hyperboxes that cover sample i
    $Max_i = 0$
    $Shortest_{il} =$ **Large_Number**
    For m = 1 to M
      $dist_{ilm} = \min\left(\left|X_{lm1} - a_{im}\right|, \left|X_{lm2} - a_{im}\right|\right)$
      If ($dist_{ilm} \leq Shortest_{il}$)
        $Shortest_{il} = dist_{ilm}$
      End If
    Next m
    If ($Shortest_{il} \geq Max_i$)
      $Max_i = Shortest_{il}$
      Class of sample i = Class of box l
    End If
  Next l
Next i

4.2.2. Classification of Uncovered samples using COUS

Uncovered samples differ from the overlap samples as they can be observed only in testing phase. Hyperboxes generated by the model cover some regions within the feature space, and it is expected to have some uncovered regions outside these hyperboxes. For a test sample that falls in an uncovered region, the class label should be determined. In this study, it is aimed to assign an uncovered sample to the class of the closest hyperbox, without changing the boundaries of the hyperbox. The closeness of a sample to a hyperbox is measured by Euclidean distance metric only based on the features that do not cover that sample. Figure 4.3 illustrates the case of an uncovered sample. According to the figure, the sample does not fit into either of the two hyperboxes. However, it is seen that hyperbox 1 covers the sample in terms of feature 2 bounds. Thus, it is not meaningful to use feature 2 in computing sample's distance to hyperbox 1. For this case, Euclidean distances between sample and hyperboxes are computed using only feature 1 for hyperbox 1, and both features 1 and 2 for hyperbox 2. In the figure, these distances are shown with dashed arrows, and the sample must be classified as class 1 since it is closer to hyperbox 1.

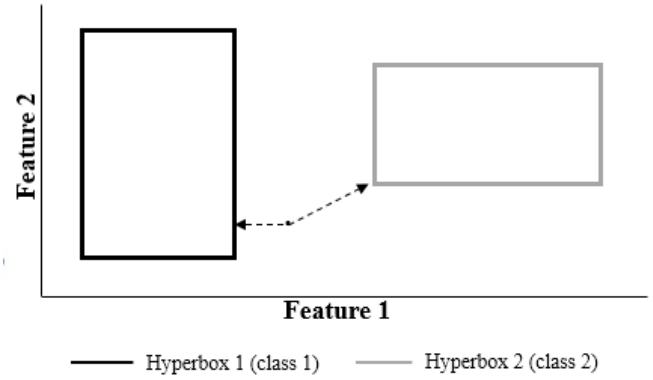


Figure 4.3. Uncovered Sample Classification

The algorithm for the classification of the uncovered samples can be summarized as follows, where L is the actual number of boxes generated by the model, M is the number of features, F is the set of features that do not cover sample, $dist_{ilm}$ is the distance of sample i to box l in terms of feature m, and $Distance_{il}$ is the Euclidean distance between sample i and box l.

For $\forall i \in$ Set of uncovered samples
  $F = \{\}$
  $MinDistance =$ **Large_Number**
  For $l = 1$ to L
    For $m = 1$ to M
      If $(X_{lm1} \le a_{im} \le X_{lm2})$
        $dist_{ilm} = 0$
      Else
        $dist_{ilm} = \min\left(\left|X_{lm1} - a_{im}\right|, \left|X_{lm2} - a_{im}\right|\right)$
        $F = F \cup \{m\}$
      End If
    Next m
    $Distance_{il} = \sqrt{\sum_{m \in F} \left(dist_{ilm}\right)^2}$
    If $(Distance_{il} \le MinDistance)$
      $MinDistance = Distance_{il}$
      Class of sample i = class of box l
    End If
  Next l
Next i

4.3. Experimental Results of Model-MOB

The MIP formulation is implemented in GAMS and solved using CPLEX 12.0 solver for the datasets that are described below. Three identical desktop PCs with Intel Core i5 3.0 GHz processor and 4.00 GB RAM are used for the runs, and the run time is limited with three hours as a termination rule.

Table 4.1. Dataset Properties

Dataset	Number of samples	Number of classes	Number of features
Simulated 1 (S1)	60	2	2
Simulated 2 (S2)	60	2	2
Iris	150	3	4
Wine	178	3	13
Seed	210	3	7
Glass	214	6	9
Breast Cancer (Wisc.)	683	2	9
Sonar	208	2	61
Ionosphere	351	2	35
Ecoli	336	5	7

Ten datasets that are given in Table 4.1 are used in the experiments. *Simulated 1* and *Simulated 2* are specifically generated for understanding the model behavior and ease of presenting solutions visually, and illustrated in Figure 4.4 and Figure 4.5, respectively. All of the remaining datasets are selected form UCI Machine Learning Repository (Dua & Karra, 2017).

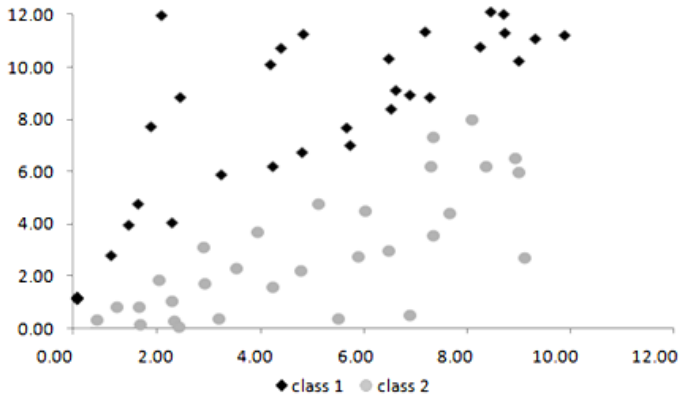


Figure 4.4. Simulated data 1

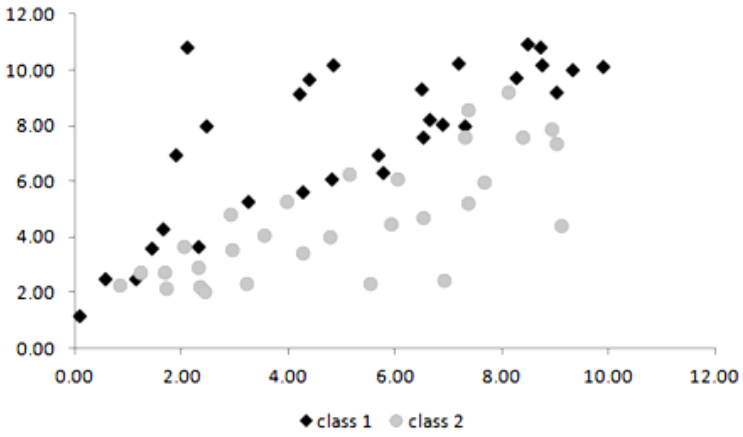


Figure 4.5. Simulated data 2

As it is illustrated in Figure 4.4, *Simulated 1* with features x and y is separated with the $x = y$ line into two classes each having thirty samples. For class 1, $x < y$ and for class 2, $x > y$. For this type of pattern, it is expected to have multiple hyper boxes to represent each class, although SVM can easily separate these two classes. *Simulated 2* is a modified version of *Simulated 1*, such that value of feature y for class 2 samples are is increased by 2. In this way, an overlapping region is obtained between the two classes over the $x = y$ line as seen in Figure 4.5.

For all datasets except the simulated ones, 80% of the data is used for training, and 20% for testing. Thirty additional samples having the same characteristic are generated for the test phase of simulated datasets.

In order to see the performance of Model-MOB, some preliminary experiments are conducted and the results are compared with Model UT-R. Weight of misclassification in UT-R is taken as 1, the weights c_1, c_2, and c_3 in MOB are also taken as 1 for all runs. Note that coefficient c in Model UT-R is equivalent to c_1 in Model-MOB. The summary of the results is presented in Table 4.2.

Table 4.2. Comparison of Model-UTR and Model-MOB

	Model-MOB					Model UT-R			
	z	**R. Gap**	**B**	**M**	**O**	**z**	**R. Gap**	**B**	**M**
S 1	7.00	0.56	5	1	1	7.00	0.56	7	0
S 2	12.00	0.75	6	6	0	12.00	0.75	5	7
Iris	5.00	0.40	4	0	1	5.00	0.40	5	0
Wine	15.00	0.80	3	0	12	N/A	N/A	N/A	N/A

z: Best integer obtained in one hour, **R.Gap**: Reported relative gap, **B:** Number of hyperboxes used in the solution, **M:** Number of misclassified samples, **O:** Number of overlap samples.

For *Simulated 1*, *Simulated 2*, and IRIS datasets, the differences in the results are not significant. However, for the WINE dataset, model UT-R cannot find any feasible solutions within the one-hour time limit. The main reason for this may be the impossibility of obtaining a non-overlapping solution with Model UT-R.

For further experimental analysis, Model-MOB is solved under different settings of objective function coefficients as described in Section 4.3.1. Then, for each setting, testing is performed using the model's solution.

4.3.1. Choice of objective function coefficients

As it is mentioned in Section 4.1, there are three coefficients c_1, c_2, and c_3 used as multipliers in the objective function for the number of boxes, the number of misclassified samples, and the number of overlap samples, respectively. Each of these coefficients are set to three different levels: 0.01, 0.1, and 1. Thus Model-MOB is run 3^3 times for each dataset. All of the runs are terminated due to the three-hour time limit without reaching optimality.

Table 4.3 illustrates the classification results for *Simulated 1*. Results of Model-MOB are listed under the *Training* columns. For the computation of accuracy in the training phase, overlap samples are counted as misclassified. *Model-MOB results* of the test phase are obtained only by the use of hyperboxes. Then, COUS heuristic is applied for the classification of uncovered and overlap samples, and resulting misclassified samples are used in the computation of test accuracy under *COUS results*.

Table 4.3. Classification Results for Simulated 1 dataset

	Coefficients			Training				Test				
								Model-MOB results			COUS results	
Run	**c_1**	**c_2**	**c_3**	**B**	**M**	**O**	**A**	**M**	**O**	**U**	**M**	**A**
1	1.00	1.00	1.00	6	0	1	0.983	1	2	5	2	0.933
2	1.00	1.00	0.10	3	0	17	0.717	4	17	2	11	0.633
3	1.00	1.00	0.01	2	0	30	0.500	4	18	0	10	0.667
4	1.00	0.10	1.00	2	10	0	0.833	8	0	5	9	0.700
5	1.00	0.10	0.10	2	10	0	0.833	8	0	5	9	0.700
6	1.00	0.10	0.01	2	0	30	0.500	4	18	0	10	0.667
7	1.00	0.01	1.00	2	10	0	0.833	8	0	5	9	0.700
8	1.00	0.01	0.10	2	10	0	0.833	8	0	5	9	0.700
9	1.00	0.01	0.01	2	10	0	0.833	8	0	5	9	0.700
10	0.10	1.00	1.00	7	0	0	1.000	0	0	8	1	0.967
11	0.10	1.00	0.10	6	0	1	0.983	1	2	5	2	0.933
12	0.10	1.00	0.01	3	0	17	0.717	5	16	2	12	0.600
13	0.10	0.10	1.00	5	2	0	0.967	3	0	6	3	0.900
14	0.10	0.10	0.10	5	1	1	0.967	3	0	6	3	0.900
15	0.10	0.10	0.01	3	0	17	0.717	4	17	2	12	0.600
16	0.10	0.01	1.00	2	10	0	0.833	8	0	5	9	0.700
17	0.10	0.01	0.10	2	10	0	0.833	8	0	5	9	0.700
18	0.10	0.01	0.01	2	10	0	0.833	8	0	5	9	0.700
19	0.01	1.00	1.00	7	0	0	1.000	1	0	7	1	0.967
20	0.01	1.00	0.10	7	0	0	1.000	1	0	7	1	0.967
21	0.01	1.00	0.01	7	0	0	1.000	0	0	8	1	0.967
22	0.01	0.10	1.00	7	0	0	1.000	1	0	8	1	0.967
23	0.01	0.10	0.10	7	0	0	1.000	0	0	8	1	0.967
24	0.01	0.10	0.01	6	0	1	0.983	1	2	5	2	0.933
25	0.01	0.01	1.00	6	1	0	0.983	1	0	8	1	0.967
26	0.01	0.01	0.10	5	2	0	0.967	3	0	6	3	0.900
27	0.01	0.01	0.01	6	1	0	0.983	1	0	8	1	0.967

B: Number of hyperboxes used in the solution, **M:** Number of misclassified samples, **O:** Number of overlap samples, **U:** Number of uncovered samples, **A:** Accuracy

Among the datasets given in Table 4.1, the model does not perform well on *Glass, Breast Cancer, Sonar, Ionosphere,* and *Ecoli.* For these datasets, the best training accuracy among 27 settings is below 30% on the overall, and the worst accuracy is less than 10% as seen in Table 4.4.

Table 4.4. Results of Datasets which do not perform well with Model-MOB

	Accuracy*		
	Worst	**Best**	**L**
Glass	0.000	0.263	10
Sonar	0.000	0.042	8
Breast cancer	0.076	0.238	8
Ionosphere	0.000	0.000	8
Ecoli	0.000	0.219	10

*Accuracy is the fraction of correctly classified samples
L: Upper limit on hyperbox index

This is mainly caused by the model complexity due to the large number of samples, classes, and/or features. The model terminates prematurely at the end of three hours. Another factor effective on these results is the type of the data. It is observed that the model performs better on the datasets with real valued features, whereas binary or integer features result in poor performance. Since the results are not satisfactory, these datasets are not analyzed any further for the selection of objective function coefficients.

On the other hand, accuracies are over 80% for *Simulated 1*, *Simulated 2*, *Iris*, *Wine*, and *Seed* datasets. Details of 27 runs for each of these datasets are given in the Appendix A. When the accuracies versus objective function coefficients are analyzed, both for training and testing phases, it is seen that the setting of the 19th run where c_1 = 0.01, c_2 = 1.00, and c_3= 1.00 performs better on the overall. For the rest of the study, objective function coefficients are fixed at these levels. Table 4.5 illustrates the results for this setting, where A is the accuracy, and U is the number of uncovered samples.

The results are not unexpected, since minimizing the number of hyperboxes is not a primary aim, but used only to prevent overfitting with too many hyperboxes (even a hyperbox for every single sample). However, it is necessary to use a finite number of hyperboxes in Model-MOB. (The maximum possible number of hyperboxes in the model is limited with the parameter L.) On the other hand, setting c_1 to zero corrupts the run time performance of the model since it is effective on the best possible bound computation in CPLEX.

Table 4.5. Classification results for the selected settings

	Training results				Test results				
					Model-MOB results			COUS results	
	B	**M**	**O**	**A**	**M**	**O**	**U**	**M**	**A**
S1	7	0	0	1.000	1	0	7	1	0.967
S2	14	0	0	1.000	3	0	15	5	0.833
Iris	4	0	0	1.000	1	1	2	1	0.967
Wine	3	0	7	0.951	0	1	11	0	1.000
Seed	3	0	33	0.805	2	9	3	3	0.929

B: Number of hyperboxes used in the solution, **M:** Number of misclassified samples, **O:** Number of overlap samples, **U:** Number of uncovered samples, **A:** Accuracy

4.3.2. Cross-validation of the results

Using the objective function coefficient setting selected above, 5-fold cross validations are performed for *Iris*, *Wine*, and *Seed* datasets. Using five folds is consistent with the 80% training and 20% testing partitions used before. Thus, results of the runs given in Table 4.5 are taken as the first fold, and four more replications are conducted for the remaining folds. However, for the simulated datasets, 60 samples were used as the training subset and 30 samples in the test subset. In order to be consistent with this two-to-one ratio, and to have five replications as for the other datasets, four additional replications are conducted by random sub-sampling for simulated datasets.

Detailed results including the misclassified, overlapped, and uncovered sample information for the conducted replications can be seen in Appendix B. Summary of the cross-validation results are given in Table 4.6 and Table 4.7 for the training and test phases, respectively. Table 4.6 also includes the information about hyperbox usage. Upper limit on hyperbox index is given in column *L*, and the average number of hyperboxes used in the solutions are given in column *B*. The number *L* is decided intuitively considering the total number of samples, features, and classes in the dataset.

Table 4.6. Cross-validation results for the training phase of Model-MOB

	Accuracy				
Dataset	**Mean**	**StDev**	**Min - Max**	**L**	**B**
Simulated 1	1.000	0.000	1.000 – 1.000	20	6.8
Simulated 2	1.000	0.000	1.000 – 1.000	20	10.6
Iris	0.995	0.007	0.983 – 1.000	10	4.8
Wine	0.948	0.013	0.930 – 0.965	10	3.2
Seed	0.798	0.024	0.768 – 0.833	10	3.4

Table 4.7. Cross-validation results for the test phase of Model MOB

	Accuracy		
Dataset	**Mean**	**StDev**	**Min - Max**
Simulated 1	0.947	0.038	0.900 – 1.000
Simulated 2	0.840	0.028	0.800 – 0.867
Iris	0.967	0.024	0.933 – 1.000
Wine	0.950	0.041	0.889 – 1.000
Seed	0.906	0.027	0.867 – 0.933

When the results are analyzed, it is seen that training and test accuracies are fairly stable. In terms of the mean values the largest differences between training and test phases are observed in the *Simulated 2*, and *Seed* datasets. For *Simulated 2*, at least 11 hyperboxes are generated by Model-MOB in the training phase replications, and none of the samples are misclassified or overlap. As the number of the hyperboxes increases, they become smaller and highly dependent on the data, and this causes larger regions which are not covered by any of the hyperboxes. Thus, more uncovered samples occur in the test phase, and any misclassification of these samples decreases the accuracy. In fact, this is a good example of overfitting, which justifies minimizing the number of hyperboxes in the objective function, and allowing overlap.

On the other hand, the situation for the *Seed* dataset is just the opposite. This time, the accuracy of test phase is significantly better. The reason for this is the number of overlap samples obtained in the training phase, which constitutes the largest overlap ratio among all datasets. The overlap samples obtained in the test phase are classified using the heuristic, however they are directly counted as misclassified in the training phase. This is also an indicator of the performance of the heuristic technique that we use.

4.3.3. Classification Performance of COUS

The performance of COUS is also analyzed in the training phase of the experiments. The 3^3 runs performed to select objective function coefficients result in larger number of overlap and uncovered samples for undesirable coefficient settings. This situation provides an opportunity to test the heuristic on larger samples. Table 4.8 summarizes the total number of overlap and uncovered samples obtained in 27 runs for each dataset. Misclassification column indicates the incorrectly classified samples by COUS, and resulting accuracy is given in the last columns.

Table 4.8. Overlap and uncovered sample classification of 27 runs using COUS

	Overlap			**Uncovered**		
Datasets	**Total**	**Misclass.**	**Accuracy**	**Total**	**Misclass.**	**Accuracy**
Simulated 1	92	35	0.620	141	12	0.915
Simulated 2	94	56	0.404	241	24	0.900
Iris	32	14	0.563	33	0	1.000
Wine	19	7	0.632	319	0	1.000
Seed	227	82	0.639	76	6	0.921

According to the results, COUS is more accurate for uncovered samples. It is not surprising to have lower accuracies for the overlap sample classifications, since their characteristic is totally different from the uncovered samples. An uncovered sample is caused by uncovered regions of the sample space. This means, hyperbox classifier cannot find a chance to label some regions due to lack of training samples in those regions. On the other hand, overlapping regions are defined by the hyperbox classifier, and they are some sort of undecided regions as mentioned before. The relatively lower accuracy of the heuristic is consistent with the undecided situation of these overlap samples.

Beside the undecided case of the overlap samples, still some extra information about their classes is provided by the hyperboxes. For the datasets having more than 3 classes, some of the classes can be eliminated for the sample if overlapping hyperboxes do not belong to that class. For example, *Iris*, *Wine*, and *Seed* has 4, 3, and 7 classes respectively. For a uniformly random assignment, the respective expected accuracies would be 1/4, 1/3, and 1/7. However, when we analyze the results in detail, it is seen that only two classes cause the overlap on the average. Accuracies which are around 50% indicate the indifference of overlap samples between those two classes.

4.3.4. Discussion on the results of Model-MOB

The behavior of Model-MOB, which is proposed to classify datasets in the training phase, is analyzed over 10 different datasets. Although the classification ability of the generated hyperboxes is good for half of these datasets, it is not possible to state the same for the remaining half, namely *Breast cancer*, *Glass*, *Sonar*, *Ionosphere*, and *Ecoli*.

The type of features has an important effect since it becomes harder to define valid hyperboxes for integer features, especially when the range of the feature values is narrow. On the other hand, Model-MOB cannot perform well for some of these five datasets although they have a fairly wide range. For example, *Breast cancer* dataset has 9 integer-valued features in the interval [1,10]. However, in the training phase, 546 samples with 9 features and 2 classes result in a high run time complexity for the model. Beside these values, L hyperboxes are allowed in the model initially as the limit of index set. Deciding on the value of L is important since smaller L values yield more misclassifications and overlaps, whereas larger L values increase the search space dramatically. Along with searching the optimal number of hyperboxes, the solver also searches for the best combination of those hyperboxes as if it was different to use the first or last so many hyperboxes.

In the literature, there are some studies that consider constructing a time-efficient model, which starts with a certain number of hyperboxes and increase this number iteratively as needed (See Section 3.2.3). Xu and Papageorgiou (2009) map hyperboxes to classes and samples to hyperboxes before solving the MIP model. They propose a compact model ready to be solved iteratively, where an additional hyperbox is introduced and required mappings are performed before each new iteration. However, these preliminary mappings may overly restrict the model. Based on the idea of Xu and Papageorgiou (2009), but leaving sample-to-hyperbox assignments to the model, an alternative mathematical formulation is proposed in this study as described in Section 4.4.

4.4. A Relaxed Mathematical Model Formulation: Model-MO

Without a major difference in the basic mechanism of Model-MOB, a new mixed integer programming model is formulated with fewer variables and constraints. This model, dropping the term for the number of boxes from the objective function, minimizes only the total number of misclassified and overlap samples. This choice comes naturally since the number of boxes can be controlled externally. Also our experiments with Model-MOB shows that the best results are obtained when the objective function coefficient for the number of boxes is smaller ($c_1 = 0.01$) than the classification and overlap coefficients ($c_2 = c_3 = 1$). This new model is called Model-MO in the rest of the study.

In Model-MO, in addition to initializing the value L for the number of hyperboxes that can be used, hyperboxes are also allocated to the classes. However, this produces only an upper bound rather than a mapping, since the model does not necessarily use all of these allocated hyperboxes. Moreover, the samples are free to be assigned to any of L hyperboxes, and overlapping is still allowed.

Decision variables and parameters of Model-MO are almost the same as those in the Model-MOB. Only a new parameter set is defined as

$$Box_{kl} = \begin{cases} 1, & \text{if box } l \text{ is allocated to class } k \\ 0, & \text{otherwise} \end{cases}$$

Since the set of hyperboxes a class can use is determined with this parameter, only the sample-to-hyperbox assignments are needed to be made. Thus, decision variables B_l and BC_{lk} used in Model-MOB are not required when the Box_{kl} parameters are used. Then, the assignment variable of Model-MO is defined as

$$Y_{il} = \begin{cases} 1, & \text{if sample } i \text{ is in box } l \\ 0, & \text{otherwise} \end{cases}$$

When a sample i is assigned to hyperbox l, it is simultaneously labeled as class k, where $Box_{kl} = 1$.

After these modifications and eliminations, the number of binary variables in Model-MO is $L(1+NK)$ less than the number of binary variables in Model-MOB. The objective function and the constraints of Model-MO are given below.

Model-MO

$$\min z = c_2 \sum_{i=1}^{N} \sum_{k=1}^{K} E_{ik} + c_3 \sum_{i=1}^{N} t_i \tag{33}$$

Subject to

$$\sum_{l=1}^{L} Y_{il} = 1 \qquad \forall i \tag{34}$$

$$X_{lm2} \geq a_{im} Y_{il} \qquad \forall i,l,m \tag{35}$$

$$X_{lm1} \leq a_{im} + Q(1 - Y_{il}) \qquad \forall i,l,m \tag{36}$$

$$\sum_{l=1}^{L} Y_{il} Box_{kl} - E_{ik} \leq d_{ik} \qquad \forall i,k \tag{37}$$

$$X_{lm2} - a_{im}(1 - Y_{il}) + \varepsilon \leq Q(O2_{ilm} + Y_{il}) \qquad \forall i,l,m \tag{38}$$

$$a_{im}(1 - Y_{il}) - X_{lm1} + \varepsilon \leq Q(O1_{ilm} + Y_{il}) \qquad \forall i,l,m \tag{39}$$

$$O_{ilm} + 1 \geq O1_{ilm} + O2_{ilm} \qquad \forall i,l,m \tag{40}$$

$$\sum_{m=1}^{M} O_{ilm} - Y_{il} + 1 \leq M + tl_{il} \qquad \forall i,l \tag{41}$$

$$t_i = \sum_{l=1}^{L} tl_{il} \qquad \forall i \tag{42}$$

$$t_i + \sum_{k=1}^{K} E_{ik} \leq 1 \qquad \forall i \tag{43}$$

$$Y_{il}, O1_{ilm}, O2_{ilm}, E_{ik} \in \{0,1\},\ X_{ilm}, O_{ilm}, tl_{il}, t_i \geq 0 \tag{44}$$

Objective function given in equation (33) minimizes the weighted sum of misclassifications and overlap samples. Each sample is assigned to one hyperbox by the constraint set (34). Constraints (35) through (42) are the equivalents of constraints (25) through (32) of Model-MOB. Constraint set (43) prevents a sample from being both overlap and misclassified. This constraint was not included in Model-MOB, since

the sample cannot be overlap and misclassified at the same time for an optimal solution. However, it is used as a valid inequality for Model-MO and guarantees to prevent this situation even for the non-optimal solutions. In addition, particularly with the changes in constraints (38) and (39), there are significantly fewer constraints in Model-MO.

4.5. Experimental Results of Model-MO

In Model-MOB, the *L* value is set before running the model, and the hyperbox allocations to the classes are performed by the model. For Model-MO, allocation of hyperboxes to the classes is given using parameters. The *L* value and hyperbox allocation is tested under two different scenarios. Experiments are conducted on the same folds of datasets that are used with Model-MOB, and run time is limited with three hours as a termination rule.

In *scenario 1,* in order to compare the two models, *L* is set to the same value as in the experiments of Model-MOB, and the hyperboxes are allocated to the classes in proportion with the sample sizes of the classes. Summary of the results are given in Table 4.9. It is observed that, some of the hyperboxes are not used although the total number of hyperboxes is not minimized in the objective function. Average number of used hyperboxes are given in column *B*. For an unused hyperbox, the lower limits on the features are found to be greater than the upper limits, and none of the samples are assigned to that infeasible box.

Table 4.9. Scenario 1 Cross-validation results for the training phase of Model-MO

	Accuracy				
Dataset	**Mean**	**StDev**	**Min - Max**	**L**	**B**
Simulated 1*	1.000	0.000	1.000 – 1.000	20	6.8
Simulated 2*	0.993	0.015	0.967 – 1.000	20	12.2
Iris*	1.000	0.000	1.000 – 1.000	10	6.8
Wine	0.965	0.023	0.937 – 0.986	8	4
Seed	0.894	0.035	0.833 – 0.923	10	3.2
Breast Cancer	0.610	0.255	0.244 – 0.802	8	3
Ecoli	0.425	0.070	0.361 – 0.537	10	5.2
Sonar	0.548	0.073	0.452 – 0.654	8	2.4
Ionosphere	0.641	0.017	0.614 – 0.657	8	2.4
Glass	0.392	0.051	0.316 – 0.450	10	6.8

*optimal solutions

In *scenario 2*, only two hyperboxes are allocated to each class. In this way, it is aimed to see the model performance when the problem size is not unnecessarily increased by the unused hyperboxes. As seen in Table 4.10, better results are obtained compared to *scenario 1*.

Table 4.10. Scenario 2 Cross-validation results for the training phase of Model-MO

	Accuracy				
Dataset	**Mean**	**StDev**	**Min - Max**	**L**	**B**
Simulated 1*	0.937	0.000	0.933 – 0.950	4	4
Simulated 2*	0.887	0.059	0.833 – 0.983	4	4
Iris*	0.990	0.004	0.983 – 0.917	6	5
Wine	0.972	0.021	0.937 – 0.986	6	4
Seed	0.960	0.009	0.952 – 0.970	6	4.8
Breast Cancer	0.759	0.059	0.670 – 0.830	4	2.5
Ecoli	0.842	0.112	0.851 – 0.920	10	6.4
Sonar	0.613	0.036	0.560 – 0.682	4	3.4
Ionosphere	0.641	0.017	0.614 – 0.657	4	2
Glass	0.429	0.070	0.380 – 0.544	10	7.4

*optimal solutions

As the *L* values are restricted in *scenario 2*, the accuracies decreased for *Simulated 1*, *Simulated 2*, and *Iris* datasets. This was expected since the optimal accuracies were obtained by using more hyperboxes in *scenario 1*. However, *scenario 2* yields better results for other datasets within the given time limit, since the problem size is reduced. It should be noted that, resulting hyperbox usage is not more than two hyperboxes per class in *scenario 1* for these datasets. The detailed results of *scenario 1* and *scenario 2* are given in Appendix C.

4.6. Comparison of Model-MO with Model-MOB

Performance of Model-MOB and Model-MO are compared in terms of classification accuracy. Figure 4.6 illustrates the average accuracies of cross-fold validations for each model and scenario. Since the cross-fold validation is not performed with Model-MOB for the last five datasets, the relevant bars in the graph is the result of a single fold for those datasets.

According to the graph, with the exception of the first three datasets solved to optimality by Model-MOB, Model-MO over-performs Model-MOB especially with *scenario 2*. However, even with Model-MO, it is still not possible to obtain the optimal solutions within three hours and some results are not satisfying enough.

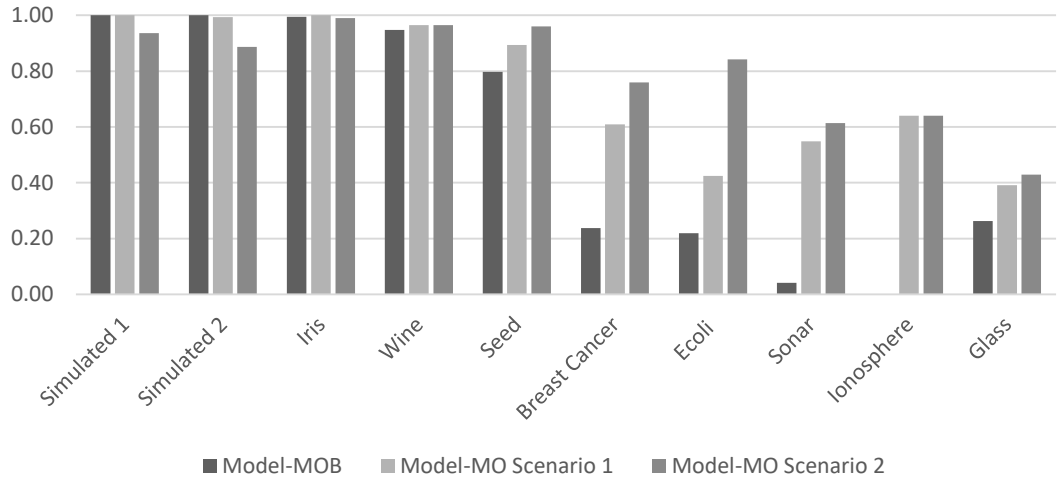


Figure 4.6. Comparison of classification accuracies for Model-MOB and Model-MO

Obtaining higher accuracies in *scenario 2* indicates that limiting the number of hyperboxes reduces the complexity significantly, allowing Model-MO to find better solutions within the time limit. However, the limit on the number of hyperboxes for each class must be determined in a more systematic manner with an algorithm that uses Model-MO and determines the hyperbox limits iteratively. There should be a way of determining when a new hyperbox is needed for each class, and hyperboxes should be added iteratively upon need to keep complexity under control. This observation constitutes the main motivation for a heuristic approach, namely the Hyperbox Classification for Binary Classes, which is presented in the following chapter.

CHAPTER 5

A MATHEURISTIC FOR HYPERBOX CLASSIFICATION FOR BINARY CLASSES: HCB

Both Model-MOB and Model-MO are comprehensive models developed to generate all necessary hyperboxes simultaneously using the complete training set. Experimental results that are presented in Section 4.6 indicate that Model-MO outperforms Model-MOB. However, Model-MO still has some complexity issues and cannot perform well as the limit on the total number of hyperboxes used increases. Obtaining higher accuracies despite the usage of fewer hyperboxes is due to this complexity problem. Hyperboxes are computationally expensive; therefore, they should be allocated to classes carefully. On the other hand, the number of hyperboxes is not the only dimension that increases the complexity. The number of features and sample size of the dataset are also effective on the run time complexity. In order to overcome this complexity problem, a matheuristic is developed in this study namely Hyperbox Classification for Binary classes (HCB).

In Section 5.1, the motivation for the HCB is given in the light of some preliminary tests. Sections 5.2 through 5.5 describe the development process of HCB, and the matheuristic HCB is finalized as described in Section 5.6.

5.1. Motivation for the HCB Matheuristic

Upper bound on the total number of hyperboxes (L), the number of features (M), and the number of samples (N) are the three main dimensions that define the run time complexity. In order to observe effects of all these dimensions, some preliminary runs are conducted for the *Breast Cancer* dataset. In the first iteration, one hyperbox is allocated to each class. The optimal solution is obtained in 1403 seconds. The number of misclassified and overlap samples are given as the first iteration in Table 5.1.

When the resulting decision variables at *iteration 1* are analyzed, it is observed that some of the features are neither used in defining the hyperbox boundaries, nor act a part in overlap prevention. For example, lower and upper bounds on the 4th feature (F4) are set to natural bounds of that feature for the hyperboxes of both classes, and these hyperboxes are highly overlapping in terms of F5. Using these information four features out of nine can be eliminated since they are found ineffective on the classification of samples. With further analysis of the results, it is observed that some bounds on values of individual features are sufficient to classify the samples. For example, if the value of a sample's F7 is greater than or equal to 8, then that sample belongs to the 2nd class. Hence, it may be possible to pre-classify and so eliminate some samples from the dataset before running the model in further iterations, if such information about the hyperbox boundaries is used as classification rules.

In order to see the effect of feature selection and sample elimination, a second iteration is conducted, again with a single hyperbox per class. In this iteration features 4, 5, 6, and 8 that are found to be ineffective in iteration 1 are removed from the dataset. In addition, by using the rules found in iteration 1, samples having an F7 value greater than or equal to 8 are pre-classified as 2nd class, and samples having an F3 value less than or equal to 1 are pre-classified as 1st class. Only by using these two rules, 314 samples out of 546 can be pre-classified. These samples do not need to be processed by the model anymore, and they can be eliminated from iteration 2. First two iterations are summarized in Table 5.1.

Table 5.1. *Result of iteration 2: elimination of features and samples*

			Model-MO Results						
Iter.	**N**	**F**	**M**	**O**	**A**	**CPU time (sec)**	**Eliminated features**	**Generated rules**	**Eliminated samples**
1	546	9	39	0	92.86	1403	(4-5-6-8)	$F7 \geq 8 \Longrightarrow c2$ $F3 \leq 1 \Longrightarrow c1$	314
2	232	5	36	0	93.41	79	-	-	-

N: number of samples, **F**: number of features, **M**: number of misclassified samples, **O**: overlap samples, **A**: percent accuracy

To sum up, a modified dataset having 5 features and 232 samples is used in iteration 2 with one free hyperbox for each class, and the optimal solution is obtained in just 79 seconds, with a higher accuracy. Note that it is impossible to obtain fewer than 39 misclassified samples using the original dataset and one hyperbox per class, since *iteration 1* solution is also optimal. The point here is that, the rules generated in *iteration 1* actually represent two hyperboxes. Thus, one additional hyperbox for each class used in *iteration 2*, are the third and fourth hyperboxes, and these additional hyperboxes are used for classification of the remaining samples left after elimination.

In *iteration 3,* some other arbitrary runs are conducted on the modified dataset, to see the effect of adding new hyperboxes to classes. Table 5.2 illustrates the results of these alternative *iteration 3* runs. The initial point of all these four runs is the end of *iteration 1.* In the first run, one more additional hyperbox is allocated to class 1, however this additional hyperbox is not used by the model, and the same result as before is obtained in a longer time. In the second run, one additional hyperbox is allocated to class 2, the optimal solution is obtained in approximately 4000 seconds, and misclassified samples are significantly reduced. In the third run, two additional hyperboxes are allocated to class 1. It is seen that, when 2 more hyperboxes are available, the model chooses to use all of them, and the classification error again significantly decreases compared to *iteration 2.* However, a proven optimal solution cannot be obtained within three hours. In the last run, an additional hyperbox is allocated to each class, and the accuracy is decreased to 87 percent, as the model is not solved optimally.

Table 5.2. Results of iteration 3: addition of hyperboxes

	Free box allocated to					**CPU time**
Run	**Class 1**	**Class 2**	**M**	**O**	**A**	**(sec)**
1	2	1	36	0	93.41	4970
2	1	2	25	0	95.42	4006
3	3	1	26	1	95.24	10800
4	2	2	11	59	87.18	10800

The results of these iterations show that feature selection and sample elimination is effective on the model performance. On the other hand, selection of the class to which a new hyperbox will be added also has a significant effect on the performance as seen from alternative runs made in *iteration 3*.

Originally, a large set of hyperboxes, and all the samples with the full set of features are introduced to Model-MO with their labels. As mentioned in Section 4.4, there are no explicit "use/do not use" decision variables for a feature or for a hyperbox. However, the model is capable of making these decisions by setting the hyperbox boundaries accordingly. If the bounds for a feature are not set different from the natural bounds of that feature for a hyperbox, then it means that that feature is redundant for that box. Moreover, the model may set infeasible boundary values for a hyperbox (i.e. lower bound is greater than the upper bound), meaning that the hyperbox is not in use.

It should be noted that the options of the solver (Cplex) is adjusted to perform branching on sample-to-hyperbox assignments first, and hyperbox boundaries are then computed based on the samples assigned to that box. Sample-to-hyperbox assignment decision is followed by branching on misclassification variables, which has a higher priority than overlap control variables. Hence, first the misclassifications are resolved, then the overlaps are determined. However, due to the complexity problem, these branches cannot be searched completely and solution terminates before reaching optimality.

Matheuristic HCB is developed to deal with this complexity problem. The initial version of HCB still uses Model-MO to generate the hyperbox boundaries. However, instead of leaving the branching decision completely to the model solution step, it starts with a minimal set of hyperboxes, and determines when to open a new hyperbox for a class, based on the results of Model-MO at each iteration. HCB also determines the correctly classified samples and fixes their sample-to-hyperbox assignments before a new iteration. Sections 5.2, 5.3 and 5.4 describe the initial versions of iterative hyperbox addition procedure, fixing sample-to-hyperbox assignments, and using these procedures in coordination with Model-MO, respectively. With the modifications reported in Section 5.5, the matheuristic HCB is finalized as in Section 5.6.

5.2. Iterative Addition of Hyperboxes

Based on the results that are obtained in Section 5.1, further experiments are conducted using four different datasets, namely *Simulated 1, Simulated 2, Simulated 3, and Simulated 4*. First two of these simulated datasets are already presented before. *Simulated 3* and *Simulated 4* are newly generated datasets having two features and 100 samples (50 samples in each class). *Simulated 3* is specifically created such that samples can be classified by using only one of the features. *Simulated 4* is a modified version of *Simulated 3* requiring both features for correct classification.

For all datasets, the same systematic is applied throughout the iterations. In the first iteration a single run is conducted by allocating one hyperbox per class. In the second iteration, two runs are conducted, one with an additional hyperbox allocation to class 1, and the other with an additional hyperbox allocation to class 2. Following the same manner in all iterations, a binary tree-like structure is obtained as in Figure 5.1.

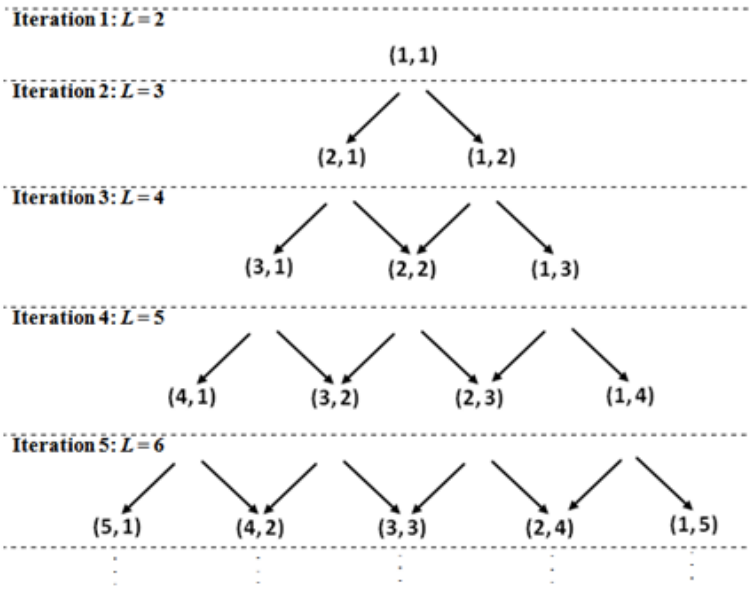


Figure 5.1. Illustration of hyperbox allocation to (class1, class2) throughout the iterations

In the figure, each level represents a single iteration. At any new iteration, the total number L of hyperboxes is increased by one, and runs are conducted for these different allocation schemes. The numbers written in parentheses describe the number of hyperboxes allocated to class 1 and class 2, respectively. The left and the right children

of any node are obtained by allocating one new hyperbox to class 1 and class 2, respectively. For any of the nodes having more than one hyperbox allocated per class, there are exactly two parent nodes. Thus, the structure is not a tree and there are some undirected cycles between the nodes.

Starting from the root node and moving through a branch to select a single node per iteration, it is possible to obtain different paths, which yield different hyperbox allocation schemes at the final iteration. By the application of HCB, it is aimed to find a good path that results in a high classification accuracy. In order to examine the behavior of alternative hyperbox allocation schemes, Model-MO runs for the datasets are conducted on each node. Note that, for a complete set of N iterations where N+1 hyperboxes are allocated between the classes at the final iteration, there are $N(N+1)/2$ runs are to be conducted.

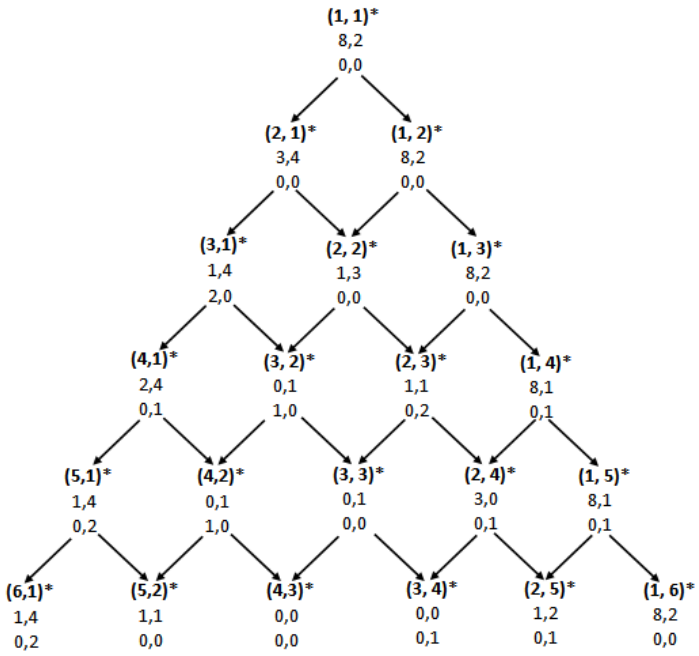


Figure 5.2. Results for Simulated 1 dataset for all nodes

Figure 5.2 illustrates the results for *Simulated 1* dataset for all nodes. The three rows of each node stand for hyperbox allocation to two classes (optimal Model-MO results are indicated with an asterisk next to the right parenthesis), misclassified samples in two classes, and overlap samples in two classes, respectively. For each row, the left

and the right of the comma represent class 1 and class 2, respectively. For example, at node (2,4), there are two hyperboxes allocated to class 1 and four hyperboxes allocated to class 2, and there are three misclassified samples of class 1 and one overlap sample of class 2 in the result. The 100 percent accuracy is obtained by allocating four hyperboxes to class 1, and three hyperboxes to class 2.

The results of *Simulated 2, Simulated 3,* and *Simulated 4* datasets are given in the Appendix D. The examination of those results reveal some general properties. First of all, the overlap figures are commonly less than the misclassification figures, although they are equally weighted in the objective function. The reason of this is the branching strategy that is set in the Cplex solver options. The branching is first performed on the misclassification decision variables before branching on the overlap related decision variables. On the other hand, some properties are important since they guide us for hyperbox addition within the HCB. These properties can be summarized as follows.

1. It is in general better to allocate one more hyperbox to the class with the highest number of misclassified samples.
2. If the newly added hyperbox is not used in the solution, then it is worth trying to add it to the other class.
3. If the accuracy is not improved with the addition of a hyperbox to a class, then it is worth trying to add it to the other class.
4. The run time complexity increases in each iteration due to the increment in the total number of hyperboxes. Thus, it may become impossible for the model to reach optimality within the time limit for some iterations. This may result in obtaining lower accuracies than the ones obtained in the earlier iterations.

The first three properties above are used for node selection to obtain a good path throughout the iterations. On the other hand, the last property indicates that iterative addition of hyperboxes is not enough to deal with the complexity issues. As mentioned before, feature selection and sample elimination should also be considered in HCB. The method and the resulting algorithm described in the following subsection deal with reducing the complexity by sample elimination.

5.3. The Hyperbox Trimming Algorithm and Fixing Sample-to-Hyperbox Assignments

In the experimental results that are reported above, Model-MO is solved from scratch at each node. In fact, it is possible to use the information of correctly classified samples and their enclosing hyperboxes generated in an earlier iteration to reduce the problem size in the subsequent iterations. Figure 5.3 illustrates the hyperboxes obtained in the root node (1,1) of *Simulated 3* dataset. As it is seen in Figure 5.3, there are no misclassified samples within hyperbox of class 1. It is possible to fix the boundary variables of this 'pure' hyperbox and pre-classify the samples belonging to that hyperbox before the following iteration.

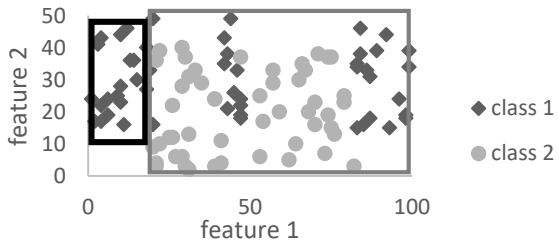


Figure 5.3. Resulting hyperbox at (1,1) of Simulated 3 dataset

On the other hand, it is not always possible to obtain such pure hyperboxes. Figure 5.4 illustrates the root node result for *Simulated 2* dataset. There are 15 misclassified samples, nine from class 1 and six from class 2. For such situations, it is still possible to extract some information for classification. The hyperbox of class 1 may be trimmed through the dashed line, and the left portion of the hyperbox with no misclassification may be fixed before the next iteration, together with the class information of the samples within that portion.

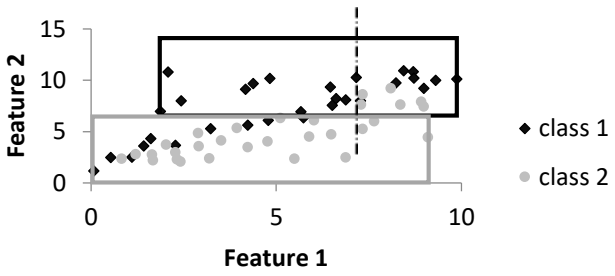


Figure 5.4. Hyperbox with misclassified samples

Hyperbox Trimming Algorithm (HTA) is developed to fix a portion of the hyperbox where all enclosed samples are correctly classified. This trimming operation is to be performed only for a single hyperbox, which has the smallest ratio of misclassification, where α_l denotes the misclassification ratio of hyperbox *l*. Before describing the algorithm, some definitions and initializations are to be made as follows.

index of the hyperbox with minimum misclassification ratio:

$$l = \arg\min_{l'=1,...,L} \{\alpha_{l'}\}$$

set of misclassified samples within hyperbox *l*:

$$E = \left\{ i \mid Y_{il} = 1, \sum_{k=1}^{K} E_{ik} = 1, i = 1,..., N \right\}$$

set of correctly classified samples within hyperbox *l*:

$$C = \{ i \mid Y_{\text{il}} = 1, E_{ik} = 1, i = 1,..., N, k = 1,..., K \}$$

set of correctly classified samples that falls out of the box after trimming on X_{lmn}:

$$R_{mn} = \emptyset \qquad m = 1,..., M, n = 1,2$$

number of correctly classified samples that falls out of the hyperbox after trimming on X_{lmn}:

$$Count_{mn} = 0 \qquad m = 1,..., M, n = 1,2$$

Based on these definitions, the pseudo code of HTA is given in Figure 5.5. Main loop of the algorithm is performed until all misclassified samples are left out of the trimmed hyperbox. In fact, trimming is equivalent to tightening lower or upper bound values X_{lmn} of the initial hyperbox in terms of each feature. Through the lines 2 to 6, the misclassified samples closest to the lower and upper bounds, and their feature values are determined.

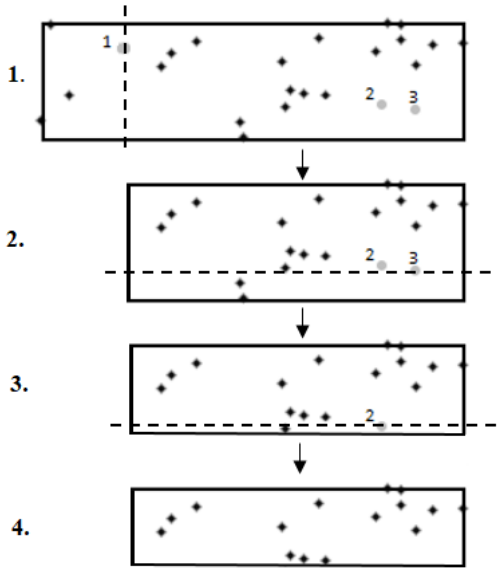
After each trimming operation some correctly classified samples may also fall out of the updated hyperbox. These samples are counted within the loop between lines 7 and 16 for all bounds of features, and the sets of these samples are determined. Between

lines 19 and 28, the bound of a feature is selected to be trimmed, such that the minimum number of correctly classified samples is left out of the updated hyperbox. Finally, the related X_{lmn} value of the hyperbox, set *E*, and set *C* are updated.

```
0     Initialize E, C, R_mn, Count_mn
1     While E ≠ Ø
2           For m = 1 to M
3                 trim_m1 = min_{i∈E}(a_im) + ε
4                 sample_m1 = {i' | a_im = min_{i∈E}(a_im), i' ∈ E}
5                 trim_m2 = max_{i∈E}(a_im) − ε
6                 sample_m2 = {i' | a_im = max_{i∈E}(a_im), i' ∈ E}
7                 For ∀i ∈ C
8                       If (a_im < trim_m1)
9                             count_m1 = count_m1 + 1
10                            R_m1 = R_m1 ∪ {i}
11                      End if
12                      If (a_im > trim_m2)
13                            count_m2 = count_m2 + 1
14                            R_m2 = R_m2 ∪ {i}
15                      End if
16                next i
17          next m
18          miscount = |C|
19          For m = 1 to M
20                For n = 1 to 2
21                      If (count_mn ≤ miscount)
22                            miscount = count_mn
23                            m' = m
24                            n' = n
25                            Bound = Trim_mn
26                      End If
27                next n
28          next m
29          X_lm'n' = Bound
30          E = E / {sample_mn}
31          C = C / R_mn
32    End while
```

Figure 5.5. Hypebox Trimming Algorithm (HTA)

Figure 5.6 illustrates steps in the main loop of HTA. For a hyperbox shown in the figure, horizontal axis represents feature 1 and vertical axis feature 2. First, the misclassified samples closest to the lower and upper boundaries are determined. For example, sample 1 is closest to the lower bound of feature 1, so $count_{1,1}$ value is computed only for this sample, and found to be equal to 3. In other words, three correctly classified samples will be trimmed out, if the lower bound of feature 1 is updated with respect to sample 1.



$i \epsilon E$	$Count_{1,1}$	$Count_{1,2}$	$Count_{2,1}$	$Count_{2,2}$
1	3	-	-	8
2	-	-	-	-
3	-	3	3	-

$i \epsilon E$	$Count_{1,1}$	$Count_{1,2}$	$Count_{2,1}$	$Count_{2,2}$
2	12	-	-	15
3	-	3	2	-

$i \epsilon E$	$Count_{1,1}$	$Count_{1,2}$	$Count_{2,1}$	$Count_{2,2}$
2	10	6	1	15

$E = \emptyset$

Figure 5.6. Illustration of HTA algorithm

As it is seen in the first table of Figure 5.6, the closest misclassified samples to the boundaries are sample 1 and sample 3, where sample 1 is closest to the lower bound of feature 1 and upper bound of feature 2, and sample 3 is closest to the lower bound of feature 2 and upper bound of feature 1. The minimum of *count* values computed for these four bound values is equal to 3, and it is observed on three different bounds. Hence, sample 1 is selected arbitrarily, and the box is trimmed at lower bound of feature 1, resulting in the 2nd hyperbox. Since there are still misclassified samples, count values are recalculated. This time, a minimum of 2 correct samples are left out if the box is trimmed at sample 3 by the lower bound of feature 2. In the 3rd hyperbox, there is only 1 remaining misclassified sample, and all *count* values are computed for that one. The trim is performed on the lower bound of feature 2. Finally, 4th hyperbox does not contain any misclassified samples, and the algorithm terminates.

HTA is a greedy algorithm, so it does not guarantee the maximum possible correctly classified samples conserved within the modified hyperbox. However, it is a fast algorithm as only 2M bounds are checked for each misclassified sample. In addition, HTA is applicable to multi-feature datasets, and it is possible to update all boundaries in all dimensions.

Application of HTA results in a misclassification free hyperbox. Thus, re-classification of the samples in that hyperbox will be unnecessary for the subsequent iterations. In order to avoid this unnecessary effort, sample-to-hyperbox assignments for the trimmed box can be fixed throughout the rest of the iterations. This fixing reduces run time of the Model-MO due to elimination of decision variables required for classification of these samples.

5.4. Putting HTA, Addition of Hyperboxes, and Sample-to-Hyperbox Assignments Together

Based on the four properties listed at the end of Section 5.2, the branching rules for hyperbox allocation are generated. In addition, using the results of HTA, sample elimination becomes possible since their hyperbox assignments are fixed. With the use of Model-MO followed by HTA and hyperbox addition, a matheuristic algorithm called ICB is developed, which constitutes an initial version of HCB. Figure 5.7 illustrates the flowchart of the ICB algorithm.

The algorithm starts with allocating a single hyperbox for each class and solving Model-MO. Unless the recently allocated hyperbox remains unused, or the total number of misclassified and overlap samples (M + O) stays unimproved, a single iteration of the ICB will be performed for each increment of the total number of hyperboxes.

In other words, in general for a node shown in Figure 5.2, Model-MO is solved, HTA is run, sample-to-hyperbox assignments are fixed, and one of the two child nodes is selected depending on the misclassified and overlap sample quantities of the classes. This corresponds to taking one of the two branches emanating from the current node in Figure 5.2.

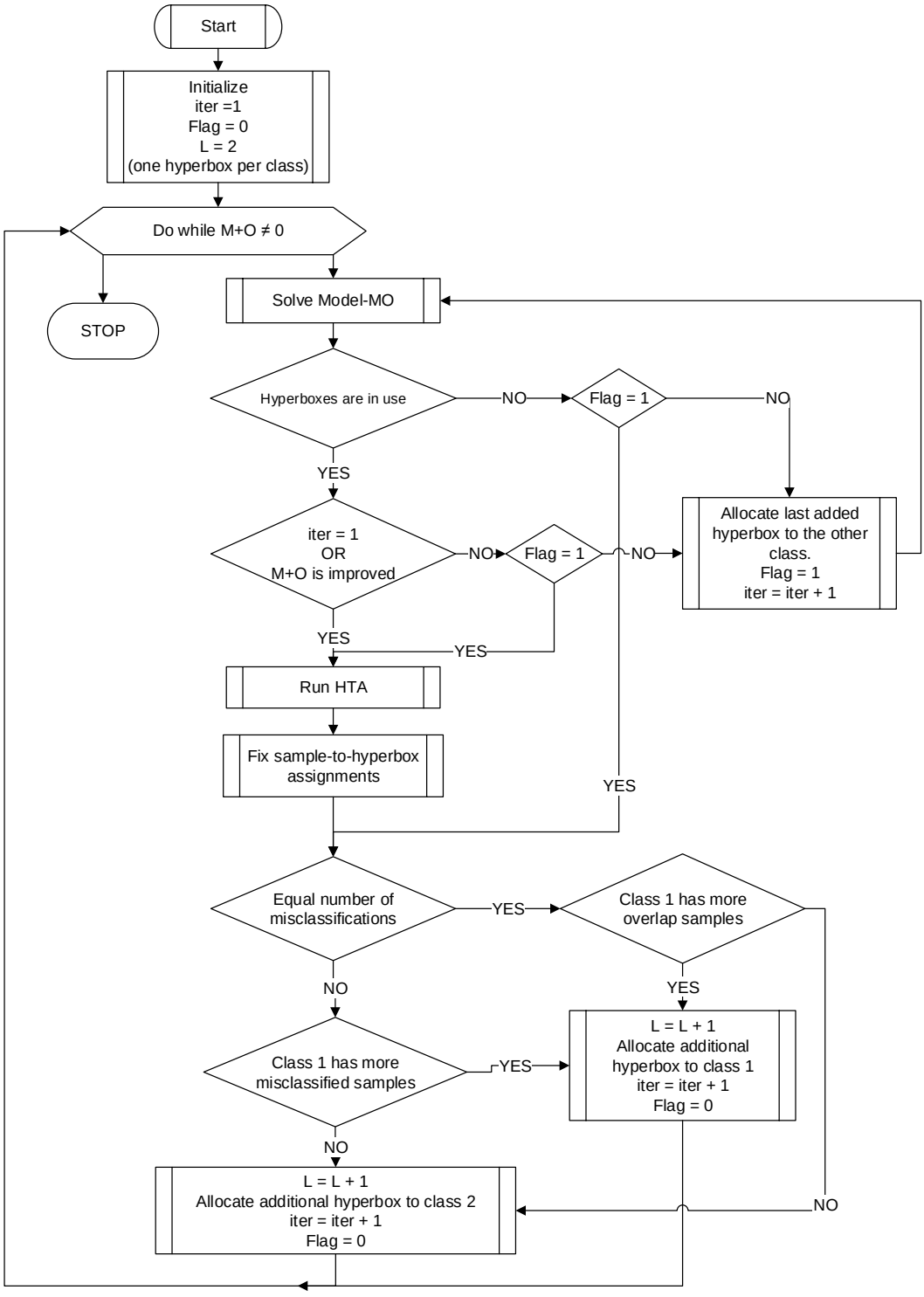


Figure 5.7. Flowchart of ICB Algorithm

However, if an improvement is not observed on a node, or the hyperbox turns out to be unused, then the last allocated hyperbox is transferred to the other class. This transfer causes a horizontal move between nodes at the same level, which is not

included in Figure 5.2. Allowing horizontal moves between nodes is important since it helps exploring the search space. Note that two horizontal moves cannot be performed in a row, since it causes a cycle. In the flowchart, *Flag* = 1 indicates that a horizontal move is performed in the last iteration, and prevents cycling between nodes.

For the cases where a horizontal branch is taken in the current iteration, application of HTA differs with respect to hyperbox usage. If there is any unused hyperbox, the subsequent branching is performed without applying HTA. However, for the non-improving nodes where all hyperboxes are in use, HTA is applied before branching.

At each iteration, as a result of applying HTA and fixing sample-to-hyperbox assignments, Model-MO searches for the bounds of only two hyperboxes, and the unclassified sample size is always less than or equal to the sample size of previous iteration. As a result, run time complexity does not increase due to hyperbox usage or sample size throughout the iterations.

The algorithm terminates when all the samples are classified correctly. As it is discussed before, forcing the algorithm to resolve all misclassifications and overlaps may cause overfitting. To prevent this overfitting, it is possible to use some thresholds on accuracy, or a decision maker may decide whether to continue or stop before the next iteration.

5.4.1. Analysis of Experimental Results

ICB algorithm is first applied to simulated datasets. The tree given in Figure 5.8 illustrates the iterations of the algorithm for Simulated 1, where no misclassified or overlap samples remain at node (4,4). According to Figure 5.2, the 100 percent accuracy was achieved at node (4,3) when none of the hyperboxes or samples are fixed. However, there exists one misclassified sample of class 1 for the same node when ICB is applied.

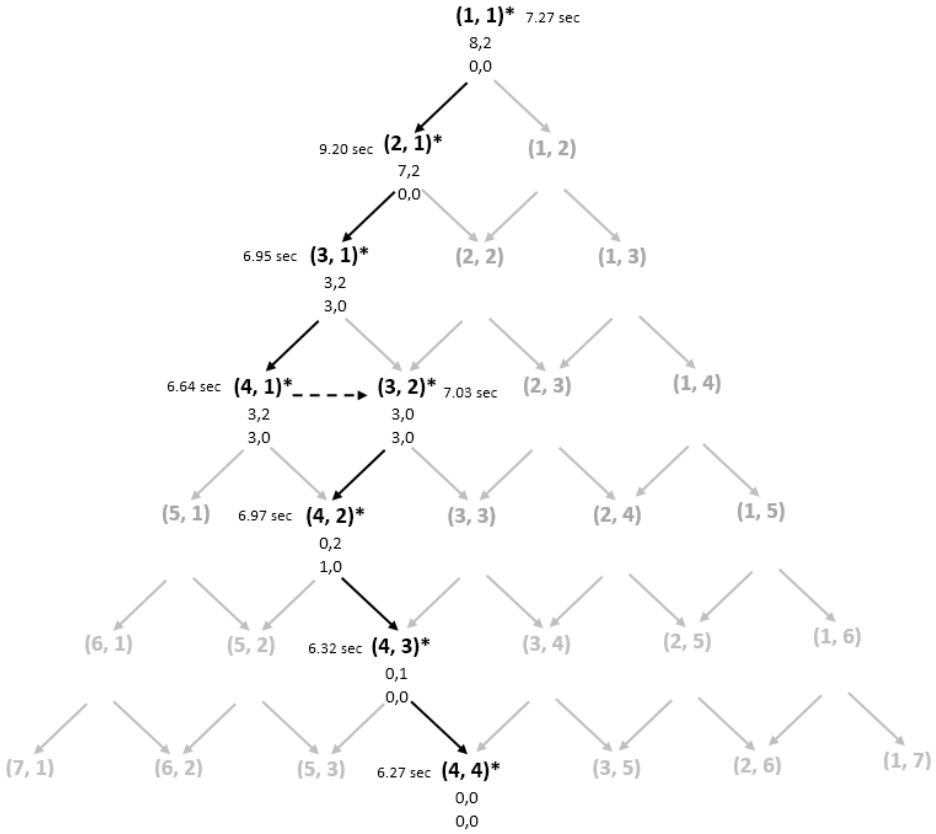


Figure 5.8. Illustration of ICB for Simulated1

The reason for this is the use of HTA. Since some hyperboxes and the samples enclosed within those boxes are fixed, the search space changes and reaching the global optimum for the original search space may become impossible. However, it is still possible to obtain the same results with the use of additional hyperboxes. Since the main aim is minimizing the classification error, the increment on the number of hyperboxes is not very critical, as long as the complexity is controlled by reducing the dataset size. For the given case, due to the hyperbox and sample fixes, 100% accuracy is achieved when there are 4 hyperboxes for each class.

The total number of hyperboxes, allocation of these boxes to classes, resulting misclassified and overlap samples, and the CPU times are shown in Figure 5.8 for *Simulated 1* dataset. Additional information about the trimmed hyperboxes and sample-to-hyperbox assignments are given in Table 5.3.

Table 5.3. Result of ICB with Model-MO for Simulated 1 dataset

Iter.	L	Box allocation	Unused hyperbox	M	O	Trimmed hyperbox	# of samples to be fixed	Model Accuracy %	Model-MO CPU time (sec)
1	2	(1,1)	-	8,2	0,0	class 1	18	83.33	7.267
2	3	(2,1)	-	7,2	0,0	class 1	5	85.00	9.202
3	4	(3,1)	-	3,2	3,0	class 1	4	86.67	6.984
4	5	(4,1)	-	3,2	3,0	-	-	86.67	6.642
5	5	(3,2)	-	0,1	1,0	class 2	8	96.67	7.029
6	6	(3,3)	-	1,0	0,0	class 2	4	98.33	6.971
7	7	(4,3)	-	0,1	0,0	class 1	3	98.33	6.324
8	8	(4,4)	-	0,0	0,0	-	-	100.00	6.273

According to Figure 5.8 and Table 5.3, additional hyperboxes are allocated to class 1 for the first four iterations, since class 1 has more misclassified samples. Iteration 4 does not improve the previous objective. Thus, a horizontal branching is performed from node (4,1) to (3,2) without applying HTA. At the end of 8 iterations, ICB terminates with 100% accuracy at node (4,4). The run time of Model-MO is less than 10 seconds for each iteration. The total run time of the model is 56.69 seconds. Run time of HTA is not reported, since it has a negligible time compared to the run time of the model. The total run times of all nodes in Figure 5.2 was approximately 3,620 seconds, which is far from the performance of ICB.

The results of ICB are given in Tables 5.4, 5.5, and 5.6 for simulated datasets 2, 3, and 4 respectively. The datasets are classified with 100% accuracy in the final iteration. Without the use of ICB, choice of the *L* value and hyperbox allocations is an open question. Since the run time complexity increases with *L*, and reaching to a proven optimal solution becomes impossible, finding the ideal hyperbox allocation scheme with trial and error is a time consuming effort. So far, the ICB algorithm provides a reasonable procedure to handle the problem.

Table 5.4. Result of ICB with Model-MO for Simulated 2 dataset

Iter.	L	Box allocation	Unused hyperbox	M	O	Class of trimmed hyperbox	# of samples to be fixed	Model Accuracy %	Model-MO CPU time (sec)
1	2	(1,1)	-	9,6	0,0	class 1	12	91.67	10.89
2	3	(2,1)	-	9,6	0,0	-	-	91.67	6.757
3	3	(1,2)	-	16,0	0,0	class 2	12	90.00	7.721
4	4	(2,2)	-	7,10	0,9	-	-	53.33	8.464
5	4	(1,3)	class 2	16,0	0,0	-	-	90.00	9.176
6	5	(2,3)	-	9,3	0,9	class 1	6	65.00	9.567
7	6	(3,3)	-	1,10	0,9	class 2	3	66.66	10.365
8	7	(3,4)	-	9,0	0,9	class 1	1	70.00	7.109
9	8	(4,4)	-	6,3	0,0	class 2	5	85.00	8.489
10	9	(5,4)	-	0,5	0,3	class 1	2	86.67	8.554
11	10	(5,5)	-	4,3	0,0	class 1	2	88.33	6.401
12	11	(6,5)	-	4,3	0,0	-	-	88.33	7.119
13	11	(5,6)	-	5,0	0,2	class 2	2	88.33	6.124
14	12	(6,6)	-	4,2	0,0	class 1	1	90.00	7.319
15	13	(7,6)	-	0,2	3,0	class 2	1	91.67	5.943
16	14	(7,7)	-	0,1	0,2	class 1	2	95.00	8.123
17	15	(7,8)	-	2,0	0,0	class 1	1	96.67	4.021
18	16	(8,8)	-	0,0	0,3	-	-	95.00	5.722
19	16	(7,9)	-	2,0	0,0	class 2	1	96.67	4.526
20	17	(8,9)	-	1,0	0,0	Class 2	1	98.33	5.468
21	18	(9,9)	-	0,0	0,0	-	-	100.00	4.006

Table 5.5. Results of ICB for Simulated 3 dataset

Iter.	L	Box allocation	Unused hyperbox	M	O	Class of trimmed hyperbox	# of samples to be fixed	Model Accuracy %	Model-MO CPU time (sec)
1	2	(1,1)	-	31,0	0,0	class 1	23	69.00	11.315
2	3	(2,1)	-	11,0	1,1	class 1	17	86.00	9.123
3	4	(3,1)	-	8,0	2,0	class 1	4	90.00	7.015
4	5	(4,1)	Class 1	9,0	1,0	-	-	90.00	10.106
5	5	(3,2)	-	0,1	1,0	class 2	24	98.00	10.273
6	6	(3,3)	-	1,0	0,0	class 2	25	99.00	4.847
7	7	(4,3)	-	0,0	0,1	class 1	3	99.00	6.125
8	8	(4,4)	-	0,0	0,0	-	-	100.00	5.982

Table 5.6. Result of ICB with Model-MO for Simulated 4 dataset

Iter.	L	Box llocation	Unused hyperbox	M	O	Class of trimmed hyperbox	# of samples to be fixed	Model Accuracy %	Model-MO CPU time (sec)
1	2	(1,1)	-	27,0	0,0	class 1	23	73.00	12.173
2	3	(2,1)	-	17,2	0,0	class 1	7	81.00	11.271
3	4	(3,1)	-	15,1	0,0	class 1	4	84.00	13.159
4	5	(4,1)	-	15,1	0,0	-	-	84.00	11.213
5	5	(3,2)	-	16,0	0,0	class 2	12	84.00	6.897
6	6	(4,2)	-	15,1	0,12	-	-	72.00	12.21
7	6	(3,3)	-	16,0	0,10	class 2	6	83.00	8.772
8	7	(4,3)	-	8,1	3,6	class 1	3	82.00	12.785
9	8	(5,3)	-	7,0	4,3	class 1	4	86.00	25.606
10	9	(6,3)	-	9,1	0,0	class 1	1	90.00	39.951
11	10	(7,3)	class 1	9,1	0,0	-	-	90.00	48.106
12	10	(6,4)	class 2	9,1	0,0	-	-	90.00	38.332
13	11	(7,4)	-	0,2	0,0	class 2	9	98.00	283.25
14	12	(7,5)	-	0,1	0,0	class 2	12	99.00	98.015
15	13	(7,6)	class 2	0,0	0,1	-	-	99.00	46.391
16	13	(6,7)	-	1,0	0,0	class 2	7	99.00	51.443
17	14	(7,7)	class 2	1,0	0,0	-	-	99.00	44.681
18	14	(8,6)	-	0,0	0,0	-	-	100.00	62.123

As mentioned before, ICB is an initial version of HCB and used to analyze the behavior of model throughout the iterations as hyperboxes are added, and sample-to-hyperbox assignments are fixed. Although the results are promising, there are some drawbacks of the ICB algorithm. As shown in Figure 5.9, for the datasets with highly overlapping classes, such as *Simulated 2* and *Simulated 4*, obtaining a smooth decrease in the (M + O) value is not possible throughout the iterations. Application of HTA may cause fluctuations in the (M + O) value, especially on the overlap side.

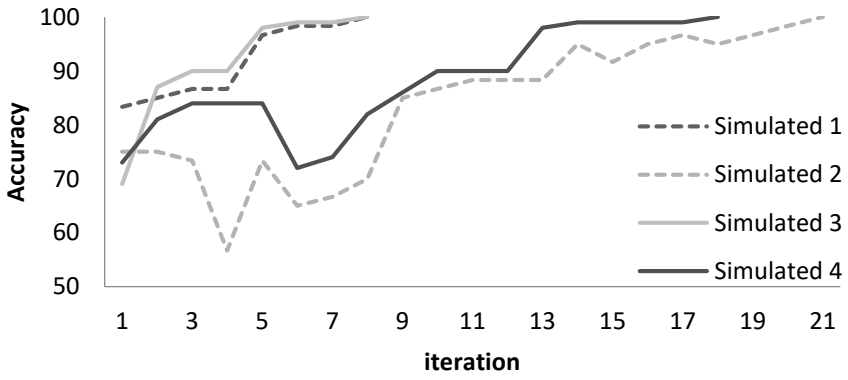


Figure 5.9. Accuracy throughout the iterations of ICB

Figure 5.10 illustrates the resulting hyperboxes of Model-MO at the 3rd and 4th iterations of ICB for *Simulated 2* dataset. Hyperbox A of class1 was obtained by HTA in the 1st iteration, and no hyperboxes and samples were fixed in iteration 2. In the 3rd iteration, HTA is applied to hyperbox B of class 2 through the dashed lines. Because the model is constrained to assign all samples to a hyperbox, the additional hyperbox C allocated to class 1 in iteration 4 highly overlaps with the trimmed hyperbox B' of the previous iteration. This makes hyperbox C useless. Finally in iteration 21, fixing the two large hyperboxes results in many small hyperboxes to resolve the overlaps.

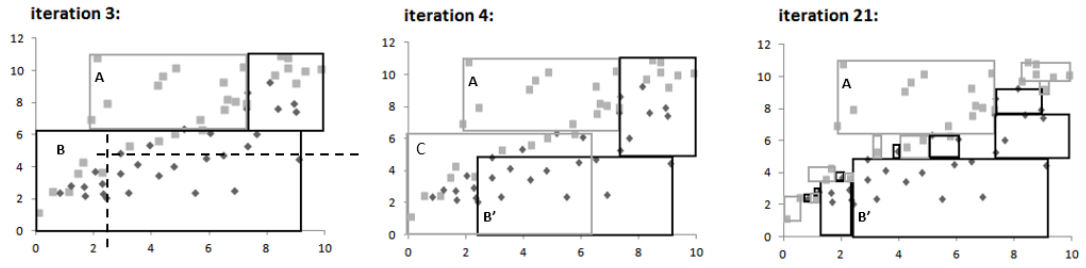


Figure 5.10. Effect of sample-to-box assignment constraint on HTA and accuracy of Model-MO

Fluctuations in accuracy increase the number of required iterations for ICB to reach the optimal solution. Even for *Simulated 2*, which has only 60 samples with two features, 21 iterations are performed until the optimal solution is found. However, applying HTA reduces the problem size and these iterations are not time consuming. In addition, it is possible for the decision maker to terminate the algorithm after any iteration, if the solution is satisfying enough.

For some larger sized datasets, this drawback of the ICB algorithm can be more crucial. As an example, the first iteration of ICB for *Breast cancer* dataset results in 39 misclassified samples and 92.86% accuracy within 1403 seconds. However, in the following iterations of ICB, Model-MO cannot be solved optimally within 20 minutes and terminates with lower accuracies despite the use of additional hyperboxes. Before trying to compensate this complexity issue with feature elimination, Model-MO is modified to alleviate this drawback by relaxing the sample assignment constraint.

5.5. Relaxing the Assignment Constraint: Model-MOU

In the light of the analysis reported in Section 5.4.1, it is observed that enforcing the model to assign every sample to a hyperbox can cause a large number of overlap samples in some iterations. In order to avoid such situations, Model-MO is revised as follows to relax the sample assignment constraint.

Model-MOU

$$\min\ z = \sum_{i=1}^{N}\sum_{k=1}^{K} E_{ik} + \sum_{i=1}^{N} t_i + \left(N - \sum_{i=1}^{N}\sum_{k=l}^{L} Y_{il} \right) \tag{45}$$

Subject to

$$\sum_{l=1}^{L} Y_{il} \le 1 \qquad \forall i \tag{46}$$

$$t_i + \sum_{k=1}^{K} E_{ik} \le \sum_{l=1}^{L} Y_{il} \qquad \forall i \tag{47}$$

(35), (36), (37), (38), (39), (40), (41), (42), (44).

Objective function given in equation (33) of Model-MO is replaced by equation (45). The new objective function minimizes the sum of the numbers of misclassified, overlap and unassigned samples. Constraint set (34) is relaxed as in (46); the samples are not forced to be assigned to a hyperbox anymore. Instead, the number of unassigned samples is to be minimized in the objective function. Finally, constraint set (43) is replaced by (47) to force the sample to be assigned to a hyperbox, if it falls into an overlap or misclassification region. The rest of Model-MOU is the same as Model-MO.

The ICB algorithm is tested with Model-MOU for *Simulated 2*, *Simulated 4*, and *Breast cancer* datasets. Results are given in Tables 5.7 through 5.9. As an additional update in the ICB algorithm, the number of unassigned samples (given in column U) is used as the first tie breaker when the number of misclassified samples is the same for both classes.

Table 5.7. Result of ICB with Model-MOU for Simulated 2 dataset

Iter.	Box alloc.	Model-MOU Results				Class of trimmed box	Fixed box alloc.	# of samples to be fixed	A2	CPU times (sec.)	
		M	O	U	A1					Model MOU	HTA
1	(1,1)	12,2	0,0	1,0	75.00	1	(1,0)	11	18.33	11.48	0.11
2	(2,1)	7,6	0,0	2,0	75.00	-	-	-	18.33	11.2	-
3	(1,2)	19,0	0,0	0,0	68.33	1	(1,1)	14	41.67	7.12	0.09
4	(2,2)	5,10	0,0	1,0	73.33	2	(2,1)	9	56.67	10.79	0.09
5	(2,3)	5,0	0,0	5,3	78.33	2	(2,2)	6	66.67	10.17	0.09
6	(3,3)	3,3	0,0	4,1	81.67	2	(2,3)	4	73.33	10.8	0.09
7	(4,3)	0,4	0,1	2,2	85.00	1	(3,3)	2	76.67	9.17	0.07
8	(4,4)	2,1	0,0	3,2	86.67	1	(4,3)	2	80.00	9.04	0.09
9	(5,4)	0,1	0,0	3,3	88.33	2	(4,4)	2	83.33	10.95	1.36
10	(5,5)	0,1	0,1	1,3	90.00	2	(4,5)	1	85.00	11.2	3.01
11	(5,6)	0,1	0,0	3,1	91.67	2	(4,6)	1	86.67	10.6	1.32
12	(5,7)	0,1	0,0	3,0	93.33	2	(4,7)	1	88.33	9.8	1.22
13	(5,8)	0,0	0,0	4,0	93.33	-	-		88.33	8.51	-
14	(6,7)	0,1	0,0	2,0	95.00	1	(5,7)	1	90.00	11.15	1.29
15	(6,8)	0,0	0,1	2,0	95.00	-	-	-	90.00	10.55	-
16	(7,7)	0,1	0,0	1,0	96.67	1	(6,7)	1	91.67	8.55	1.37
17	(7,8)	0,1	0,0	1,0	96.67	-	-	-	91.67	8.3	-
18	(8,7)	0,1	0,0	0,0	98.33	1	(7,7)	1	93.33	8.18	1.13
19	(8,8)	0,0	0,1	0,0	98.33	-	-	-	93.33	8.14	-
20	(9,7)	0,1	0,0	0,0	98.33	2	(8,7)	2	96.67	8.11	0.19
21	(9,8)	0,0	0,0	0,0	100.0				100.0	7.89	

A1: Accuracy of Model-MOU (%), **A2**: Accuracy of fixed samples (%)

Table 5.8. Result of ICB with Model-MOU for Simulated 4 dataset

Iter.	Box alloc.	Model-MOU Results				Class of trimmed box	Fixed box alloc.	# of samples to be fixed	A2	CPU times (sec.)	
		M	O	U	A1					Model MOU	HTA
1	(1,1)	17,0	0,0	10,0	73	1	(1,0)	23	23	17.34	4.77
2	(2,1)	14,3	0,0	2,0	81	1	(2,0)	7	30	13.47	0.10
3	(3,1)	19,0	0,0	3,2	76	-	-	-	30	6.50	-
4	(2,2)	17,0	0,0	3,0	80	2	(2,1)	12	42	6.58	0.13
5	(3,2)	12,1	0,0	5,0	82	1	(3,1)	4	46	14.94	0.10
6	(4,2)	11,0	0,0	5,0	84	1	(4,1)	4	50	11.97	1.75
7	(5,2)	11,1	0,0	3,0	85	2	(4,2)	11	61	14.73	0.13
8	(6,2)	0,11	0,0	7,10	72	-	-	-	61	13.69	-
9	(5,3)	1,2	0,0	7,10	80	2	(4,3)	11	72	10.54	0.12
10	(5,4)	0,0	0,0	4,5	91	1	(5,3)	8	80	9.11	1.80
11	(5,5)	0,0	0,0	4,2	94	2	(5,4)	11	91	8.10	1.46
12	(6,5)	0,0	0,0	2,2	96	2	(5,5)	3	94	11.25	1.15
13	(6,6)	0,0	0,0	2,0	98	1	(6,5)	2	96	7.35	2.48
14	(7,6)	0,0	0,0	1,0	99	2	(6,6)	2	98	7.69	0.22
15	(8,6)	0,0	0,0	0,0	100				100	6.90	

A1: Accuracy of Model-MOU (%), **A2**: Accuracy of fixed samples (%)

Table 5.9. Result of ICB with Model-MOU for Breast Cancer dataset

Iter.	Box alloc.	Model-MOU Results				Class of trimmed box	Fixed box alloc.	# of samples to be fixed	A2	CPU times (sec.)	
		M	O	U	A1					Model MOU	HTA
1	(1,1)	11,3	5,7	8,7	93	1	(1,0)	224	41	1200	0.27
2	(2,1)	9,2	5,3	16,22	90	1	(2,0)	105	60	1200	0.13
3	(1,2)	15,7	6,3	14,55	82	-	-	-	60	268.27	-
4	(3,1)	8,1	0,2	8,20	93	2	(2,1)	12	62	1200	3.90
5	(4,1)	8,1	10,0	7,6	94	1	(3,1)	75	76	1200	0.33
6	(5,1)	6,0	0,0	55,67	78	-	-	-	76	671.98	-
7	(4,2)	5,1	0,9	20,26	87	2	(3,2)	20	80	1200	0.23
8	(5,2)	2,0	0,0	51,43	82	-	-	-	80	1200	-
9	(4,3)	0,0	0,0	55,31	84	2	(3,3)	8	81	1200	2.82
10	(5,3)	2,0	0,0	53,46	82	-	-	-	81	1200	-

A1: Accuracy of Model-MOU (%), **A2**: Accuracy of fixed samples (%)

Compared with the results of Model-MO embedded ICB given in Table 5.4, results in Table 5.7 shows that ICB with Model-MOU solves *Simulated 2* using fewer hyperboxes in the same number of iterations. On the other hand, *Simulated 4* is solved using the same number of hyperboxes in fewer iterations as shown in Table 5.8. Figures 5.11 and 5.12 illustrate the accuracies at each iteration of ICB using different models. On the x-axis, iterations numbers for Model-MO and Model-MOU do not necessarily correspond to the same hyperbox allocation scheme. For *Simulated 2* dataset, it is seen in Figure 5.11 that accuracy increases more smoothly with Model-MOU compared to Model-MO, and ICB ends up with 100% accuracy. However, it is still not possible to state that the accuracy is non-decreasing through the iterations.

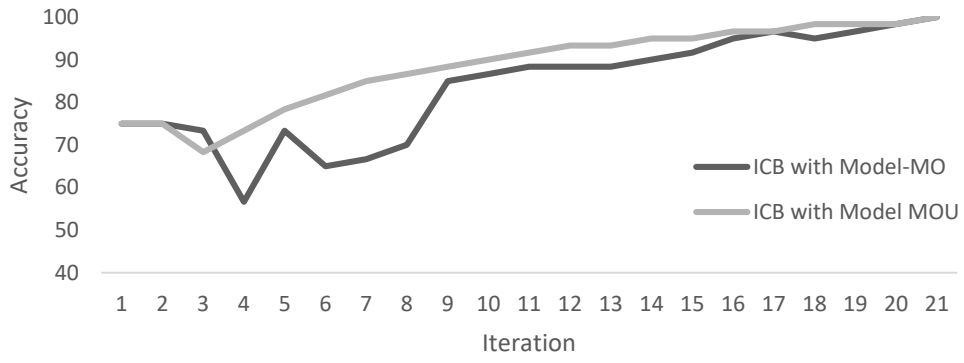


Figure 5.11. Simulated 2 – accuracy vs. iterations of ICB with different models

For *Simulated 4* dataset, accuracy also fluctuates with Model-MOU, as seen in in Figure 5.12. However, 100% accuracy is achieved earlier.

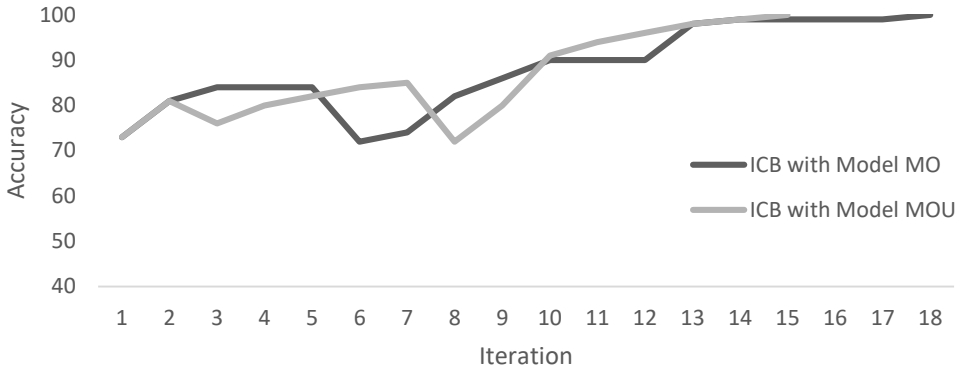


Figure 5.12. Simulated 4 – accuracy vs. iterations of ICB with different models

Under the use of Model-MOU, performance of the ICB algorithm is still not promising to reach a 100% accuracy for larger datasets. As seen in Figure 5.13 for *Breast Cancer* dataset, at the end of iteration 10, the accuracy gets worse compared to the first iteration. Highest accuracy is observed at iteration 5 with 93.96%, but it is only slightly higher than the initial accuracy. Considering the computing time of over four hours for those iterations, the experiment for *Breast cancer* dataset is interrupted at iteration 10.

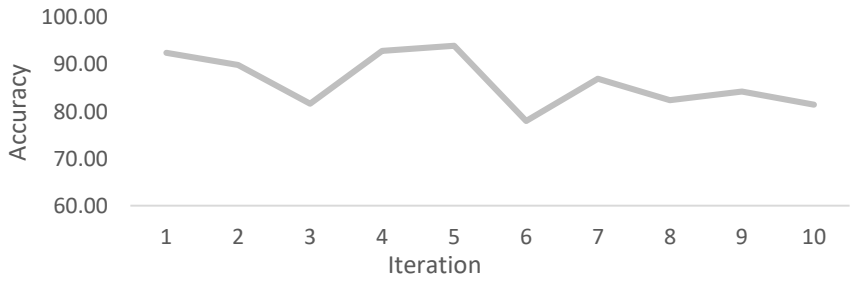


Figure 5.13. Breast Cancer – accuracy vs. iterations of ICB with Model-MOU

Beside the plotted model accuracies (column A1) above, accuracy of fixed samples is also reported in Tables 5.7 through 5.9. Accuracies given in column 'A1', and plotted in the figures above stands for the direct result of mathematical model run, before applying the HTA. When HTA is applied, some correctly classified samples are fixed in a hyperbox. Accuracy of fixed samples reflects this correctly classified portion of

the samples. In other words, this accuracy corresponds to the solved portion of the problem and given in the column 'A2' of these tables. If the model terminates with optimal solution in one iteration, this figure constitutes a lower bound on the model accuracy of the succeeding iteration as it is formulated below.

$$f_i^m = f_{i-1} + \hat{f}_i \qquad (47)$$

f_i is non-decreasing, and $\hat{f}_i \geq 0$

where

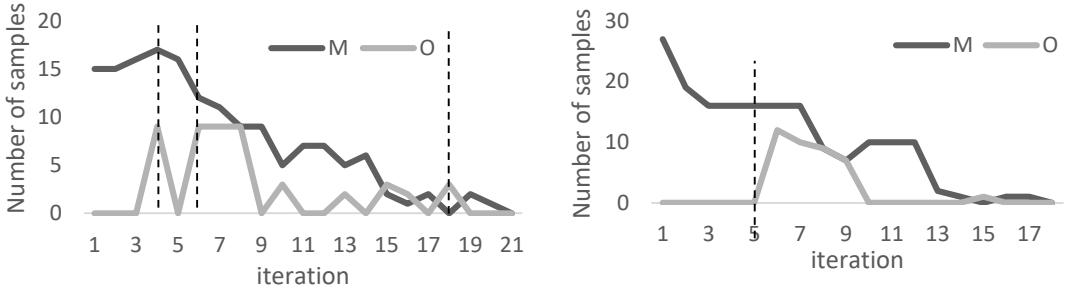
f_i^m : Model - MOU accuracy at iteration i

f_i : accuracy of fixed samples at the end of iteration i

$\hat{f}_i$: percent of correctly classified non - fixed samples by the model at iteration i

Depending on the previously fixed samples and hyperboxes, $\hat{f}_i$ can be either smaller than, equal to, or larger than $\hat{f}_{i-1}$. This causes obtaining $f_i^m \leq f_{i-1}^m$ for some instances, and this is the main reason of the fluctuation in model accuracy.

The accuracy decreases in some iterations in terms of (M + O) for Model-MO, and (M + O + U) for Model-MOU. When the downturn points of the accuracy plots are analyzed, a difference is observed between Model-MO and Model-MOU. The plots given in Figure 5.14 illustrate the number of misclassified and overlap samples when Model-MO is used in ICB. The vertical dashed lines show the iterations at which the accuracy decreases. As it is seen, the source of the accuracy decrease is mainly the overlap samples for both *Simulated 2* and *Simulated 4* datasets.

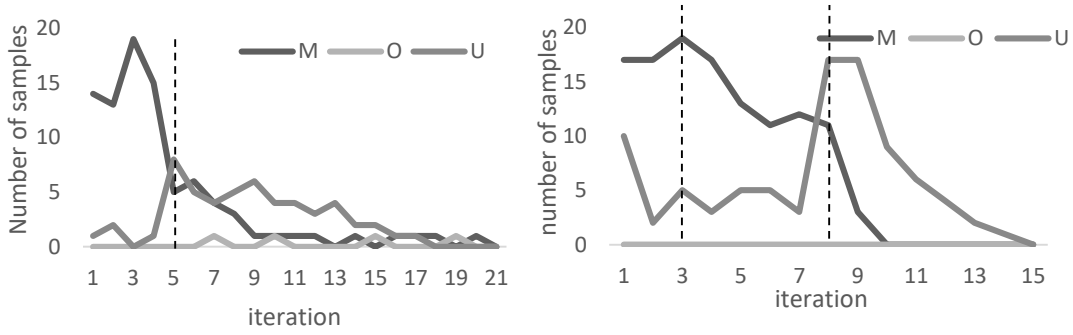


a. Simulated 2 *b. Simulated 4*

Figure 5.14. Misclassified and overlap samples in ICB with Model-MO

Model-MOU is proposed to eliminate decreases in accuracy by relaxing the sample-to-hyperbox assignment constraint. Implicitly, it aims to smooth out the accuracy fluctuations and reduce the required number of iterations for ICB. As illustrated in Figure 5.15, the main reason for accuracy decrease is the unassigned samples as opposed to overlap samples when Model-MOU is used. This brings flexibility to the procedure because the unassigned samples can be assigned to new hyperboxes generated at later iterations.

These findings indicate the benefit of Model-MOU in reducing the loss of accuracy due to overlap samples. However, use of Model-MOU alone does not have a substantial impact on reducing the accuracy fluctuations or the number of iterations. Hyperbox allocation and trimming rules also need to be revised to improve the performance.



a. Simulated 2 *b. Simulated 4*

Figure 5.15. Misclassified, overlap, and unassigned samples, ICB with Model-MOU

5.6. Matheuristic for Hyperbox Classification for Binary Classes

In order to support and enhance the positive impact of Model-MOU, the reasons behind the increases in the number of overlap and unassigned samples are further analyzed throughout the iterations of ICB both with Model-MO and Model-MOU. According to these analyses, there are three main causes for those increases.

1. The main purpose of trimming a hyperbox in HTA is to reduce the sample size and the run time complexity. However, it is observed that selecting the

hyperbox with the smallest misclassification ratio for trimming does not guarantee the largest sample size reduction, because the amount of reduction attained by fixing a hyperbox is not known until after applying HTA. As there are always only two candidate hyperboxes to be trimmed in each iteration and the HTA runtime is short, it is better to apply HTA to both hyperboxes, and then trim and fix the one that covers more samples.

2. In each iteration, hyperbox addition decision is made based on the number of misclassified samples. However, Model-MOU also generates unassigned samples. Increase in the number of unassigned samples may be caused by the lack of a hyperbox for the class of those samples. Thus it is more reasonable to add a hyperbox to the class with a higher (M + U) value.
3. Finally, the most obvious cause is the allocation of a new (free) hyperbox to a class in an iteration. If the algorithm chooses to fix the hyperbox of one class but adds a new hyperbox to the other class, the former class will have no free allocated hyperbox whereas the latter class will have two free hyperboxes in the subsequent iteration. As a result, all samples of the former class with no free hyperbox remain unassigned or misclassified. An example situation can be seen in Table 5.8 for the 2^{nd} and 7^{th} iterations of *Simulated 4* dataset.

In the light of the analysis on experimental results, ICB is revised to alleviate the drawbacks that are explained above. The modifications made on the ICB algorithm to obtain HCB matheuristic are summarized below.

- The HTA is applied to trim both hyperboxes in each iteration, and the sample-to-hyperbox assignments are fixed for the box of class *i* that covers more samples after the trimming.
- A new hyperbox is added to class *j* that has more misclassified and unassigned (M + U) samples instead of the class that has only more misclassified samples.
- If $i \neq j$, then the sample-to-hyperbox assignments are also fixed for the box of class *j*, and a new (free) hyperbox is also added for class *i*.
- Model-MOU is used for the classification instead of Model-MO.

After these modifications, it is guaranteed to have two free hyperboxes at each iteration. In addition, there will always be a free hyperbox for each class in each iteration of HCB. In this way, HCB has additional branches in the search tree as shown in Figure 5.16. It is important to note that the model accuracy may still fluctuate due to $\hat{f}_i$. However, these fluctuations are expected to decrease by the use of HCB.

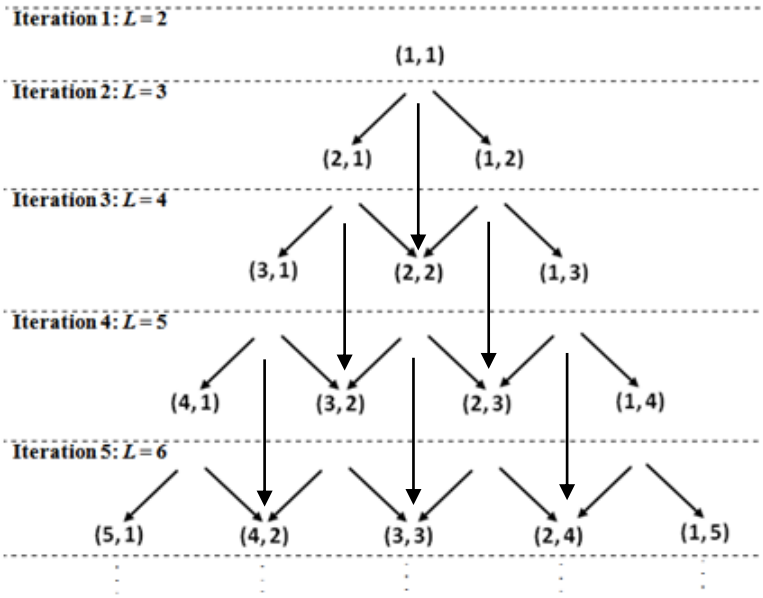


Figure 5.16. Hyperbox allocation possibilities to (class 1, class2) throughout the iterations of HCB

The finalized algorithm for HCB is given in Figure 5.17. Some set and variable definitions used within the algorithm is given below.

D: Set of all samples.

P: Set of samples whose sample-to-hyperbox assignments are fixed.

P_k: Set of candidate samples that can be fixed for class k.

H^t: Set of hyperboxes for iteration t.

l_k: Index for candidate hyperboxes of class k added to H^t after HTA.

e_k: Number of misclassified and unassigned samples for class k in objective function z^* of Model-MOU.

o_k: Number of overlap samples for class k in objective function z^* of Model-MOU.

HCB Algorithm

```
1     t = 1
2     H^1: one hyperbox for each class
3     P = { }
4     z* = a large value
5     While z* ≠ 0
6           SOLVE Model - MOU for D fixing hyperbox assignments of samples in P using
            hyperboxes in H^t, → S, z*
7           z_t = z*
8           If t ≠ 1
9                 If l ≠ 0 and (z_t ≥ z_t-1 or hyperbox l is unused)
10                      H^t = H^t-1
11                      Fix boundaries and sample-to-box assignments for hyperbox l_k'
12                      Update H^t by adding a free hyperbox for class k'
13                      P = P - P_k ;  P = P ⋃ P_k'
14                      SOLVE Model - MOU for D fixing hyperbox assignments of
                        samples in P using hyperboxes in H^t, → S, z*
15                      z_t = z*
16                End if
17          End if
18          CALL HTA for solution S → l1, l2, P1, P2
19          If |P1| > |P2| and  e1 ≥ e2
20                l = l1; fix boundaries and sample-to-box assignments for hyperbox l1
21                k = 1; update H^t+1 by adding a free hyperbox for class 1 to H^t
22                P = P ⋃ P1
23                k' = 2
24          Else If |P2| > |P1| and e2 ≥ e1
25                l = l2; fix boundaries and sample-to-box assignments for hyperbox l2
26                k = 2; update H^t+1 by adding a free hyperbox for class 2 to H^t
27                P = P ⋃ P2
28                k' = 1
29          Else If |P1| = |P2| and e1 = e2
30                If  o1 ≥ o2
31                    l = l1; fix boundaries and sample-to-box assignments for hyperbox l1
32                    k = 1; update H^t+1 by adding a free hyperbox for class 1 to H^t
33                    P = P ⋃ P1
34                    k' = 2
35                Else
36                    l = l2; fix boundaries and sample-to-box assignments for hyperbox l2
37                    k = 2; update H^t+1 by adding a free hyperbox for class 2 to H^t
38                    P = P ⋃ P2
39                    k'= 1
40                End if
41          Else
42                l = 0; fix boundaries and sample-to-box assignments for hyperboxes l1, l2
43                Update H^t+1 by adding a free hyperbox for each class to H^t
44                P = P ⋃ P1 ⋃ P2
45          End if
46          t = t+1
47    END WHILE
```

Figure 5.17. HCB Algorithm

In the algorithm, lines 1 to 4 comprise the initialization. The main loop between lines 5 and 47 is repeated until 100% accuracy is achieved with $z^* = 0$. In line 6, after fixing hyperbox assignments for the samples in set P, Model-MOU is solved for dataset D using hyperboxes in H^t. Set H^t includes one free hyperbox for each class (l_1, l_2) whereas the boundary variables for all remaining hyperboxes in the set are fixed. Solution S of Model-MOU includes all decision variable values required for applying HTA at line 18. HTA is applied to both free hyperboxes and resulting sample size values of trimmed hyperboxes ($|P_1|$, $|P_2|$) are reported. e_1, e_2, o_1, and o_2 values are obtained from objective function z* of solution S. Based on these values, a new hyperbox allocation decision is made by setting k along with fixing boundaries and sample-to-hyperbox assignments for selected hyperbox(es) between the lines 19 and 45. Note that the number of overlap samples is used as a tie breaker when the two classes are the same in terms of $|P_k|$ and e_k values. Between lines 8 and 17, the selected class for a new hyperbox allocation is changed from k to k' if an unused hyperbox or an unimproved objective function value is observed in the new Model-MOU solution. Beside this change performed in lines 10 through 12, set P is also updated as in line 13 and Model-MOU is solved again.

Once the hyperbox to be trimmed is decided, boundary variables of this hyperbox and respective sample-to-hyperbox assignment variables are fixed, and new decision variables are included in the model for new free hyperbox(es). In the following iteration, Model-MOU is solved with this newly constrained search space and new variables. Since the new variables are defined only for two free hyperboxes at each iteration, increase in the problem size throughout the iterations remain under control.

CHAPTER 6

EXPERIMENTAL RESULTS OF HCB

In this section, the performance of HCB matheuristic is first analyzed in comparison with ICB which utilizes Model-MOU. Due to the satisfactory results of these analyses, HCB is used as the classification algorithm in the rest of the experiments, and the results are reported. Finally, performance of HCB is compared with that of the CART algorithm (Classification and regression tree).

6.1. Preliminary Results and Comparison of HCB with ICB

As preliminary experiments, HCB algorithm is applied to *Simulated 2*, *Simulated 4*, and *Breast Cancer* datasets to see the effect of the modifications in the algorithm over ICB. For all datasets, 100% training accuracy is achieved. Details of the iterations are given in Appendix E. The training accuracies (refer to Model-MOU accuracy for the rest of this thesis) for iterations of ICB and HCB are compared in Figures 6.1-6.3 for these datasets.

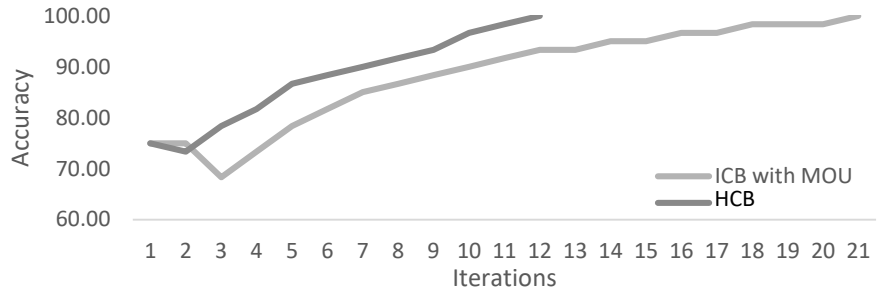


Figure 6.1. Comparison of model accuracies for ICB and HCB – Simulated 2

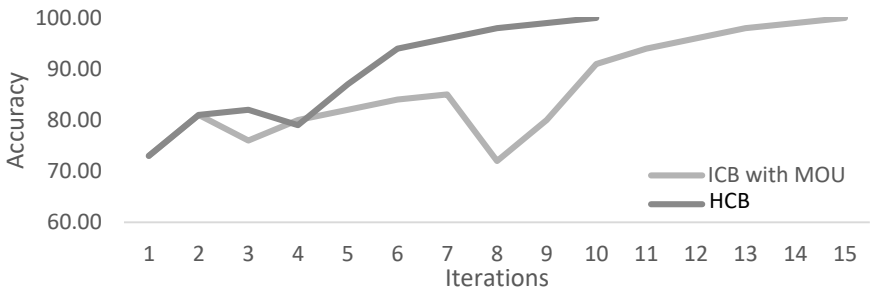


Figure 6.2. Comparison of model accuracies for ICB and HCB – Simulated 4

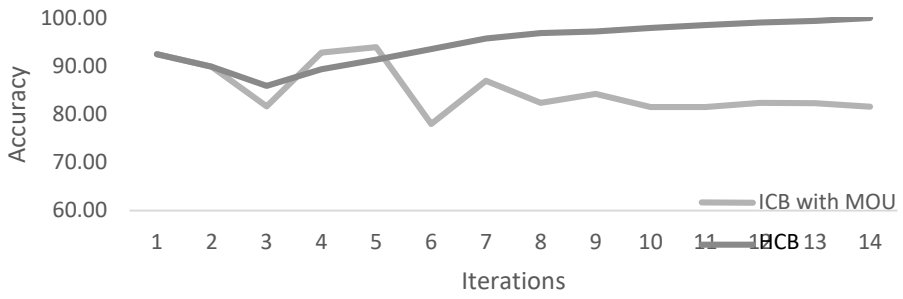


Figure 6.3. Comparison of model accuracies for ICB and HCB – Breast Cancer

As seen in the figures, HCB is more stable and performs fewer iterations to reach the result. The reason HCB makes fewer iterations is the possibility of directly branching from node (n_1, n_2) to node (n_1+1, n_2+1) as in Figure 5.16 by adding two hyperboxes in the same iteration. Moreover, unused hyperboxes are rarely observed with the HCB algorithm. Thus, horizontal actions are also reduced. According to the results, HCB outperforms ICB and ICB with Model-MOU in the training phase.

Table 6.1. Training and test phase accuracies for HCB – Simulated 2, Simulated 4, Breast Cancer

Iter.	Simulated 2		Simulated 4		Breast Cancer	
	Training accuracy	Test accuracy	Training accuracy	Test accuracy	Training accuracy	Test accuracy
1	75.00	70.00	73.00	64.00	92.49	94.89
2	73.33	**90.00**	81.00	76.00	89.93	94.89
3	78.33	86.67	82.00	64.00	85.90	93.43
4	81.67	83.33	79.00	72.00	89.38	95.62
5	86.67	86.67	87.00	68.00	91.39	95.62
6	88.33	86.67	94.00	72.00	93.59	95.62
7	90.00	86.67	96.00	**80.00**	95.79	**96.35**
8	91.67	83.33	98.00	80.00	96.89	95.62
9	93.33	83.33	99.00	80.00	97.99	95.62
10	96.67	83.33	100.00	80.00	98.53	95.62
11	98.33	83.33	-	-	99.08	95.62
12	100.00	83.33	-	-	99.45	95.62
13	-	-	-	-	100.00	95.62

In order to evaluate the classification performance of HCB, test phase accuracies are also analyzed at each iteration. Test accuracies for all datasets are given with the training accuracies in Table 6.1. The maximum test phase accuracies reached are shown in bold, and all of them are obtained in earlier iterations.

As a common observation, test accuracies are fixed at a certain iteration and remain the same for the rest of the iterations. It is an expected behavior due to the characteristics of the generated hyperboxes. First of all, when a test sample falls into a fixed hyperbox of the wrong class, that sample will be misclassified for the rest of the iterations. In addition, hyperboxes generated in early iterations are larger, and their sizes get smaller throughout the algorithm. For example, after the 7th iteration of *Simulated 4* dataset, three additional hyperboxes are used to classify only six remaining samples as it is seen in Table 6.2. One of those hyperboxes covers only one sample, thus its effect in the test phase is insignificant.

Table 6.2. Iteration details of HCB for Simulated 4 dataset

Iter.	Allocation	M	O	U	Fixed box allocation	# of samples to be fixed
1	(1,1)	17,0	0,0	10,0	(1,0)	23
2	(2,1)	14,3	0,0	2,0	(2,1)	21
3	(3,2)	14,1	0,0	1,2	(3,2)	15
4	(4,3)	9,0	0,1	4,7	(3,3)	11
5	(4,4)	0,0	0,0	8,5	(4,4)	17
6	(5,5)	0,0	0,0	4,2	(5,4)	4
7	(6,5)	0,0	0,0	2,2	(5,5)	3
8	(6,6)	0,0	0,0	2,0	(6,5)	2
9	(7,6)	0,0	0,0	1,0	(7,6)	1
10	(8,6)	0,0	0,0	0,0		

6.2. Contribution of Hyperboxes to Classification

In order to see the marginal contribution of each hyperbox to classification, two hyperbox measures namely ‘purity’ and ‘power’ are developed are computed. Purity is about the classification accuracy of a hyperbox, whereas power is related with the amount of data classified by a hyperbox. These measures can be defined as follows.

$$purity_{kl} = \frac{N_{kl} - M_{kl}}{N_{kl}} \tag{48}$$

$$power_{kl} = \frac{N_{kl}}{N_k} \tag{49}$$

where

N_k : number of samples in class k ,

N_{kl} : number of samples from class k in hyperbox l of class k ,

M_{kl} : number of misclassified samples in hyperbox l of class k .

Using Equations (48) and (49), purity and power values are computed for the hyperboxes throughout the iterations of HCB for each dataset. Training phase results for *Simulated 2* are given in Table 6.3. Since the algorithm terminates with 100% training accuracy, there are no misclassified samples at the end, and all of the hyperboxes have a purity score of 1.0. The power decreases as the additional hyperboxes get smaller in later iterations. The last seven hyperboxes are used to classify only one sample each, starting with the 11th iteration. As seen in Table 6.1, these small hyperboxes are indeed ineffective in the test phase.

Table 6.3. Training phase purity and power of hyperboxes obtained with HCB – Simulated 2 dataset

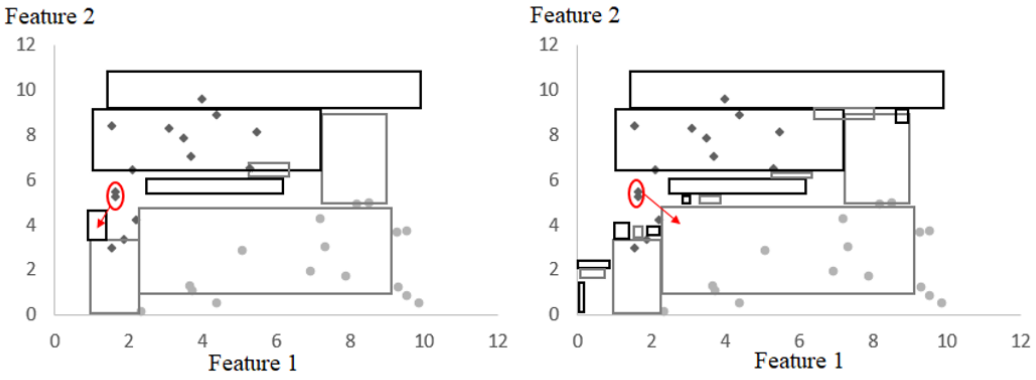
l	N_{kl}	M_{kl}	**Iteration**	**Class**	**Purity**	**Power**
1	11	0	1	1	1.0000	0.3667
2	14	0	1	2	1.0000	0.4667
3	6	0	3	2	1.0000	0.2000
4	9	0	2	1	1.0000	0.3000
5	4	0	4	2	1.0000	0.1333
6	2	0	4	1	1.0000	0.0667
7	2	0	6	1	1.0000	0.0667
8	2	0	5	2	1.0000	0.0667
9	1	0	6	2	1.0000	0.0333
10	2	0	7	1	1.0000	0.0667
11	1	0	9	2	1.0000	0.0333
12	1	0	8	1	1.0000	0.0333
13	1	0	9	1	1.0000	0.0333
14	1	0	11	1	1.0000	0.0333
15	1	0	10	2	1.0000	0.0333
16	1	0	12	2	1.0000	0.0333
17	1	0	12	1	1.0000	0.0333

The power, and purity of the hyperboxes are also computed for the test phase. Table 6.4 gives the results for the first four iterations and five hyperboxes of *Simulated 2* dataset. The remaining twelve hyperboxes do not have any contribution to the classification of the test samples, including the 6th hyperbox that was finalized at iteration 4 in the training phase. Although the power of these hyperboxes for the rest is zero, better test accuracies are observed using the hyperboxes generated at iterations 5, 6, and 7 as seen in Table 6.1.

Table 6.4. Test phase purity and power of hyperboxes obtained with HCB – Simulated 2 dataset

l	N_{kl}	M_{kl}	Iteration	Class	Purity	Power
1	1	0	1	1	1.0000	0.0333
2	15	3	1	2	0.8000	0.5000
3	8	0	3	2	1.0000	0.2667
4	0	0	2	1	-	0.0000
5	1	2	4	2	1.0000	0.0333

The fluctuation in the test accuracy is caused by the assignment of unknown and overlap samples using the COUS algorithm. Even some slight changes in the hyperbox coordinates may alternate the class of an unknown or overlap sample, as it is illustrated in Figure 6.4. As seen in Figure 6.4a, two unknown samples marked in red circle is assigned to the shown hyperbox of correct class, given the resulting hyperboxes of iteration 5. However, this hyperbox is not fixed in that iteration yet and replaced by the smaller hyperboxes in the further iterations. As seen in Figure 6.4b, resulting small hyperboxes of the correct class are relatively farther, and COUS assigns these samples to the closer but incorrect class. The reason for not obtaining the maximum test accuracy with the boxes obtained at the final iteration of HCB is an indicator of overfitting caused by resolving all overlaps and forcing the samples for assignment.



a. 5th iteration, accuracy: 86.67%

b. Final iteration, accuracy: 83.33%

Figure 6.4. Illustration of test phase results at different iterations of HCB – Simulated 2

Purity and power values for the hyperboxes of *Simulated 4* and *Breast Cancer* dataset are given in Tables 6.5 and 6.6, respectively. As for *Simulated 2*, power of the hyperboxes has a decreasing trend throughout the iterations, as the training accuracy increases.

Table 6.5. Purity and power of hyperboxes obtained with HCB – Simulated 4 dataset

			Training phase				Test phase			
l	**Iter.**	**Class**	***Nkl***	***Mkl***	**Purity**	**Power**	***Nkl***	***Mkl***	**Purity**	**Power**
1	1	1	23	0	1.0000	0.4600	3	0	1.0000	0.2500
2	2	2	12	0	1.0000	0.2400	2	0	1.0000	0.1538
3	2	1	7	0	1.0000	0.1400	2	0	1.0000	0.1667
4	3	1	4	0	1.0000	0.0800	2	4	0.0000	0.1667
5	3	2	11	0	1.0000	0.2200	2	0	1.0000	0.1538
6	4	1	8	0	1.0000	0.1600	3	0	1.0000	0.2500
7	5	2	11	0	1.0000	0.2200	1	0	1.0000	0.0769
8	4	2	11	0	1.0000	0.2200	2	1	0.5000	0.1538
9	6	1	4	0	1.0000	0.0800	1	0	1.0000	0.0833
10	7	2	3	0	1.0000	0.0600	0	0	-	0.0000
11	8	1	1	0	1.0000	0.0200	0	0	-	0.0000
12	9	2	2	0	1.0000	0.0400	0	0	-	0.0000
13	10	1	1	0	1.0000	0.0200	0	0	-	0.0000
14	10	1	2	0	1.0000	0.0400	0	0	-	0.0000

Table 6.6. Purity and power of hyperboxes obtained with HCB - Breast Cancer dataset

			Training phase				Test phase			
l	**Iter.**	**Class**	***Nkl***	***Mkl***	**Purity**	**Power**	***Nkl***	***Mkl***	**Purity**	**Power**
1	1	1	224	0	1.0000	0.6257	55	0	1.0000	0.6395
2	2	2	109	0	1.0000	0.5798	34	1	0.9706	0.6667
3	2	1	105	0	1.0000	0.2933	23	0	1.0000	0.2674
4	4	1	12	0	1.0000	0.0335	2	0	1.0000	0.0233
5	3	2	21	0	1.0000	0.1117	5	1	0.8000	0.0980
6	4	2	15	0	1.0000	0.0798	3	1	0.6667	0.0588
7	5	1	5	0	1.0000	0.0140	1	0	1.0000	0.0116
8	5	2	8	0	1.0000	0.0426	2	1	0.5000	0.0392
9	6	1	4	0	1.0000	0.0112	0	0	-	0.0000
10	6	2	11	0	1.0000	0.0585	2	0	1.0000	0.0392
11	8	1	2	0	1.0000	0.0056	0	0	-	0.0000
12	7	2	9	0	1.0000	0.0479	1	2	1.0000	0.0196
13	8	2	4	0	1.0000	0.0213	1	0	1.0000	0.0196
14	12	1	2	0	1.0000	0.0056	0	0	-	0.0000
15	9	2	4	0	1.0000	0.0213	1	0	1.0000	0.0196
16	10	2	3	0	1.0000	0.0160	0	0	-	0.0000
17	11	2	3	0	1.0000	0.0160	1	0	1.0000	0.0196
18	12	2	2	0	1.0000	0.0106	0	0	-	0.0000
19	13	1	1	0	1.0000	0.0028	0	0	-	0.0000
20	13	2	2	0	1.0000	0.0106	0	0	-	0.0000

Figure 6.5 illustrates the power of hyperboxes and accuracies for the hyperboxes generated in HCB iterations of *Simulated 4* dataset. As it is seen, patterns of power and accuracy are almost symmetrical around the dashed line.

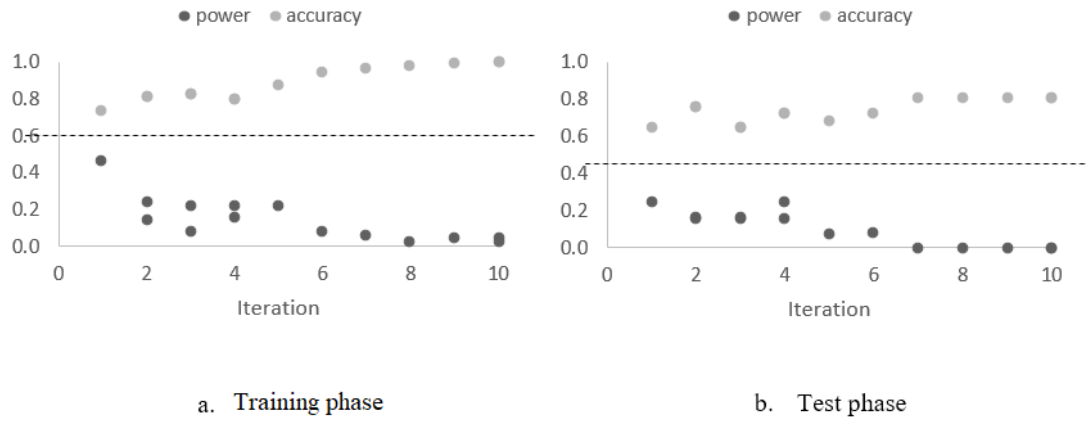


Figure 6.5. Power and accuracy for Simulated 4

To summarize, for these three datasets, HCB generates the most powerful hyperboxes in the earlier iterations. The later iterations of the algorithm have only incremental contributions to the training phase accuracy, and in some cases they even reduce the test phase accuracy due to overfitting. This indicates that selecting the hyperboxes of an earlier iteration may be more beneficial on the classification performance, as the later iterations result in overfitting. The main motivation to develop the HCB matheuristic is to deal with the complexity issues. However, it also provides a side benefit of assessing contribution of additional hyperboxes for different levels of accuracy, instead of reporting only a final configuration with 100% training accuracy but somewhat overfitted results. Based on these preliminary results obtained so far, HCB seems promising, therefore further experiments are conducted with HCB.

6.3. Computational Results of HCB

In order to evaluate the performance of HCB, experiments are extended. Among the previously introduced data sets, *Simulated 2* and *Simulated 4* are used with their five folds, since they are more complex versions of *Simulated 1* and *Simulated 3*, respectively. Five folds of *Breast Cancer* dataset are also used due to its multi-feature structure. As an example case for larger sized dataset, *Skin Segmentation* dataset from

UCI Machine Learning Repository (Due & Karra, 2017) is selected. As given in Table 6.7, the source of runtime complexity with *Skin Segmentation* is the number of samples since there are only three features representing the data. HCB matheuristic is capable of dealing with this complexity by the use of HTA and fixing sample-to-hyperbox assignments, but it may not find even an initial solution for some datasets as the number of features increases. Therefore, experiments on datasets with larger number of features is postponed until after addition of feature selection.

Table 6.7. Skin Segmentation Dataset

Dataset	**Number of samples**	**Number of classes**	**Number of features**	**Data type**
Skin Segmentation	194,198	2	3	Integer

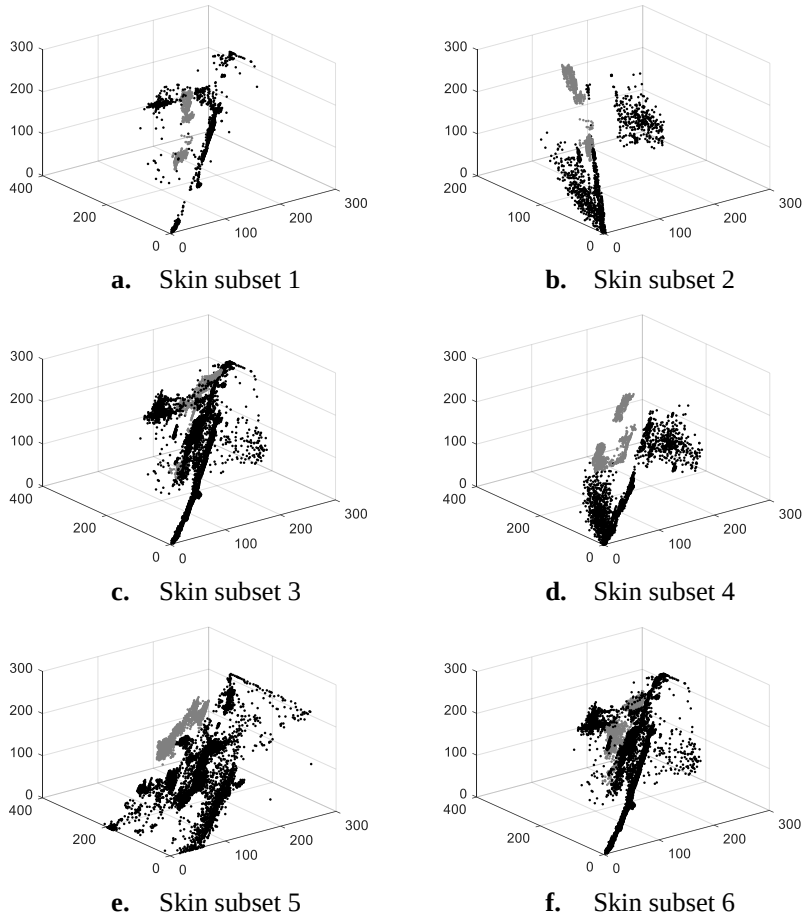
Skin Segmentation has three features that represent the color codes of RGB (Red, Green, Blue) each takes values between 0 and 255. The two classes denote human skin and non-human skin, which correspond to class 1 and class 2, respectively. Due to the variety in human skin colors, samples of class 1 take values in different zones of RGB color ranges, which can be resolved using hyperboxes at different locations, and this can be an informative exercise for HCB. Smaller subsets of different complexities in terms of class separability are generated from *Skin Segmentation* dataset to see the performance of HCB.

Before generating these subsets, the duplicate samples are removed. Among the remaining 51,444 unique samples, 14,654 are class 1 and 36,790 are class 2 samples. Subsets are generated in different sample sizes as 1750, 2500, and 7000. Ratio of class 1 and class 2 samples are taken as 1:2.5 in order to maintain the original scheme. Then, samples of class 1 and class 2 are divided into non-overlapping bins of respective sizes. For example, bins of 500 samples for class 1 and bins of 1250 samples for class 2 are used for the 1750 sample case. By cross matching the bins, different subsets are generated. In each subset size, two subsets are selected for the experiment and named as easy and hard. The terms easy and hard refer to the difficulty level to separate the classes of samples due to the interwoven structure of the samples.

Table 6.8. Subsets of Skin Segmentation dataset

Subset	Size	Number of samples from class 1	Number of samples from class 2	Property
Skin Subset 1	1,750	500	1,250	hard
Skin Subset 2	1,750	500	1,250	easy
Skin Subset 3	3,500	1,000	2,500	hard
Skin Subset 4	3,500	1,000	2,500	easy
Skin Subset 5	7,000	2,000	5,000	easy
Skin Subset 6	7,000	2,000	5,000	hard

The properties of the selected subsets are summarized in Table 6.8 and illustrated in Figure 6.6. As seen in the figure, the two classes overlap (samples are interwoven) more in hard subsets and less in easy subsets.



a. Skin subset 1 **b.** Skin subset 2

c. Skin subset 3 **d.** Skin subset 4

e. Skin subset 5 **f.** Skin subset 6

Figure 6.6. Visualization of Skin Segmentation subsets

Five folds of the six subsets are generated where 20% is dedicated as the test data. However, it is observed that the sample size still causes complexity issues. In the first iteration of HCB, a 20-minute run of Model MOU results in so many overlap and unassigned samples that it is not computationally feasible to continue with further iterations of the algorithm. When the overlaps are analyzed in detail, it is realized that they mostly occur among the hyperboxes of the same class. In Model MOU, constraint set (41) enforces separation of all hyperboxes even though they belong to same class, which makes the formulation intractable as the number of samples increases. Thus, the constraint is modified as in (50) to prevent the overlaps only for the hyperboxes of different classes. Here $\sum_{k} d_{ik}k$ and $\sum_{k} Box_{lk}k$ give the classes of sample *i* and hyperbox *l*, respectively.

$$\sum_{m=1}^{M} O_{ilm} - Y_{il} + 1 \leq M + tl_{il} \qquad \forall i,l \quad \text{where} \quad \sum_{k} d_{ik}k \neq \sum_{k} Box_{lk}k \qquad (50)$$

However, this modification is not sufficient and the model still terminates in the first iteration with many unassigned samples such that an efficient hyperbox containing a large number of samples cannot be obtained for HTA. In order to alleviate this unassigned samples problem, in the first iteration of HCB, a single hyperbox is created by the model such that all samples of the larger class are forced to be assigned to this hyperbox.

In Table 6.9 results of the first iterations for training datasets of five folds with and without this assignment constraining are given. In the table, *N* denotes the total number of samples, and N_1 and N_2 denote the numbers of samples in *class 1* and *class 2*, respectively. Also, z_1 and z_2 denote the objective function values without and with the assignment constraining. It is seen that, constraining the model gives better results for datasets that are classified as hard, with an exception of subset 1 which yields not better but an equal objective function value as given in Table 6.9. For the last two subsets, which have 5600 samples in the training dataset, forcing the assignment yields better results both for easy and hard cases. During the experiments, this type of assignment to a single hyperbox is enforced when it is not possible to obtain an efficient solution of the model in the first iteration.

Table 6.9. Model results for the first iterations of Skin Segmentation subsets without and with single hyperbox assignments

Dataset	N_1	N_2	N	z_1	z_2
skin-s1c1	400	1000	1400	400	**400**
skin-s1c2	400	1000	1400	400	**400**
skin-s1c3	400	1000	1400	400	**400**
skin-s1c4	400	1000	1400	400	**400**
skin-s1c5	400	1000	1400	400	**400**
skin-s2c1	400	1000	1400	207	400
skin-s2c2	400	1000	1400	190	400
skin-s2c3	400	1000	1400	201	400
skin-s2c4	400	1000	1400	191	400
skin-s2c5	400	1000	1400	197	400
skin-s3c1	800	2000	2800	801	**800**
skin-s3c2	800	2000	2800	1706	**800**
skin-s3c3	800	2000	2800	800	**800**
skin-s3c4	800	2000	2800	800	**800**
skin-s3c5	800	2000	2800	803	**800**
skin-s4c1	800	2000	2800	800	**800**
skin-s4c2	800	2000	2800	719	800
skin-s4c3	800	2000	2800	412	800
skin-s4c4	800	2000	2800	393	800
skin-s4c5	800	2000	2800	385	800
skin-s5c1	1600	4000	5600	1834	**1600**
skin-s5c2	1600	4000	5600	1598	**1600**
skin-s5c3	1600	4000	5600	1629	**1600**
skin-s5c4	1600	4000	5600	5120	**1600**
skin-s5c5	1600	4000	5600	5600	**1600**
skin-s6c1	1600	4000	5600	2787	**1600**
skin-s6c2	1600	4000	5600	1600	**1600**
skin-s6c3	1600	4000	5600	3657	**1600**
skin-s6c4	1600	4000	5600	5199	**1600**
skin-s6c5	1600	4000	5600	3762	**1600**

Results for Simulated 2 Dataset

During the preliminary experiments on *Simulated 2*, 60 training samples and 30 test samples were taken and random sub-sampling was used to obtain five replications. In this section, 60 training and 15 test samples are used within the 5-fold scheme. The results are given in Table 6.10 with the details of each iteration, including the hyperbox allocation, model results, the number of samples fixed by HTA, and the classification accuracies for training and test phases.

Table 6.10. Iteration details of HCB for five folds of Simulated 2 dataset

Training phase													Test phase	
				Model-MOU Results				Class of trimmed hyperbox	# of samples to be fixed		CPU times (sec.)			
Fold	Iter.	L	Box alloc.	M	O	U	A_M			A_{TB}	Model MOU	HTA	M	A
1	1	2	(1,1)	12,2	0,0	1,0	75.00	1 - 2	25	41.67	11.481	0.171	6	66.67
	2	4	(2,2)	5,10	0,0	1,0	73.33	2	9	56.67	6.747	0.047	3	**80.00**
	3	5	(2,3)	5,0	0,0	5,3	78.33	1 - 2	6	66.67	5.885	0.102	3	80.00
	4	6	(3,3)	3,3	0,0	4,1	81.67	1 - 2	6	76.67	6.043	0.149	3	80.00
	5	8	(4,4)	0,1	0,1	3,3	86.67	2	2	80.00	5.524	1.719	3	80.00
	6	9	(4,5)	0,1	0,1	3,2	88.33	1 - 2	3	85.00	5.12	1.813	3	80.00
	7	11	(5,6)	0,0	0,1	3,2	90.00	1	2	88.33	5.248	2.317	3	80.00
	8	12	(6,6)	0,0	0,0	3,2	91.67	1	1	90.00	5.565	2.375	3	80.00
	9	13	(7,6)	0,0	0,0	2,2	93.33	1 - 2	2	93.33	4.794	1.703	3	80.00
	10	15	(8,7)	0,0	0,1	0,1	96.67	2	1	95.00	5.562	1.43	3	80.00
	11	16	(8,8)	1,0	0,0	0,0	98.33	1	1	96.67	3.216	0.216	3	80.00
	12	17	(9,8)	0,0	0,0	0,0	100.00		2	100.00	3.685		3	80.00
2	1	2	(1,1)	6,8	0,0	1,0	75.00	2	19	31.67	6.251	0.011	3	**80.00**
	2	3	(1,2)	0,9	2,6	4,0	65.00	1 - 2	18	61.67	2.405	1.828	3	80.00
	3	5	(2,3)	1,2	0,1	7,4	78.33	1 - 2	9	76.67	3.658	0.164	3	80.00
	4	7	(3,4)	0,1	0,0	6,2	85.00	1	5	85.00	6.903	1.406	4	73.33
	5	8	(4,4)	0,0	0,0	4,1	91.67	1	3	90.00	3.528	0.664	4	73.33
	6	9	(5,4)	0,1	0,0	0,1	96.67	1 - 2	3	95.00	3.231	0.078	4	73.33
	7	11	(6,5)	0,1	0,0	0,0	98.33	2	1	96.67	3.488	1.367	4	73.33
	8	12	(6,6)	0,0	0,0	0,0	100.00		2	100.00	3.529		4	73.33
3	1	2	(1,1)	6,8	0,0	1,1	73.33	2	17	28.33	7.194	0.168	2	86.67
	2	3	(1,2)	7,8	0,0	1,0	73.33	1	16	55.00	4.811	0.132	2	86.67
	3	4	(2,2)	1,4	0,0	5,2	80.00	1	5	63.33	8.892	0.332	4	73.33
	4	5	(3,2)	0,0	1,1	6,6	78.33	2	5	71.67	9.822	1.239	4	73.33
	5	6	(3,3)	2,1	0,0	4,2	85.00	1	4	78.33	4.49	0.191	3	80.00
	6	7	(4,3)	2,2	0,0	2,1	88.33	1 - 2	4	85.00	4.56	0.207	2	86.67
	7	9	(5,4)	1,1	0,0	1,1	93.33	1	2	88.33	3.362	0.211	3	80.00
	8	10	(6,4)	1,0	0,0	1,2	93.33	1 - 2	2	91.67	4.125	2.057	3	80.00
	9	12	(7,5)	0,0	0,0	2,1	95.00	1	1	93.33	19.643	0.135	3	80.00
	10	13	(8,5)	0,0	0,0	0,2	96.67	2	1	95.00	6.213	0.035	1	**93.33**
	11	14	(8,6)	0,0	0,0	0,1	98.33	2	1	96.67	5.175	1.013	1	93.33
	12	15	(8,7)	0,0	0,0	0,0	100.00		2	100.00	18.3		1	93.33
4	1	2	(1,1)	6,4	0,0	2,0	80.00	1 - 2	34	56.67	8.116	0.172	3	**80.00**
	2	4	(2,2)	0,5	0,0	7,2	76.67	2	5	65.00	7.508	1.086	3	80.00
	3	5	(2,3)	3,0	0,0	5,2	83.33	1	6	75.00	4.722	1.336	4	73.33
	4	6	(3,3)	1,1	0,0	2,2	90.00	2	4	81.67	7.851	0.195	3	80.00
	5	7	(3,4)	0,1	0,0	3,0	93.33	1	3	86.67	4.207	1.270	4	73.33
	6	8	(4,4)	0,0	0,0	3,1	93.33	1	2	90.00	3.178	1.527	4	73.33
	7	9	(5,4)	0,0	0,0	1,1	96.67	1 - 2	4	96.67	2.504	1.172	4	73.33
	8	11	(6,5)	0,0	0,0	0,0	100.00		2	100.00	2.666		4	73.33
5	1	2	(1,1)	9,3	0,0	1,0	78.33	1 - 2	32	53.33	6.764	0.171	4	**73.33**
	2	4	(2,2)	3,0	0,0	7,7	71.67	1 - 2	6	63.33	5.692	1.398	4	73.33
	3	5	(3,2)	0,3	1,0	2,5	81.67	2	4	70.00	3.991	1.067	4	73.33
	4	6	(3,3)	3,1	0,0	1,2	88.33	1 - 2	7	81.67	3.501	0.187	4	73.33
	5	8	(4,4)	2,0	0,2	0,3	88.33	1 - 2	3	86.67	3.87	0.156	4	73.33
	6	10	(5,5)	1,0	0,0	1,3	91.67	1 - 2	3	91.67	3.511	1.563	4	73.33
	7	12	(6,6)	0,0	0,0	0,2	96.67	2	1	93.33	4.17	1.016	4	73.33
	8	13	(6,7)	0,0	0,0	0,0	100.00		4	100.00	3.056		4	73.33

M, O, U : Number of misclassified, overlap, and unassigned samples, respectively

A_M: Model-MOU accuracy, **A_{TB}**: accuracy of fixed samples, **A**: Test phase accuracy, all in percentages

On the average, it takes 9.6 iterations and 13.6 hyperboxes to reach 100% training accuracy within 61.87 seconds. Maximum hyperbox usage is observed in the first fold with 17 hyperboxes throughout 12 iterations. Maximum CPU time is approximately 103 seconds. Best test accuracy throughout the iterations of all folds is obtained as 93.33% for the 3rd fold where the average is 78.67%. Consistent with the preliminary experimental results, it is again observed that the highest test accuracy is obtained before the last iteration for all fold as shown in bold.

Purity and power values of the generated hyperboxes and their test phase performances are summarized in Table 6.11. As described in Section 6.1, purity is about the classification accuracy of a hyperbox, whereas power is related with the amount of data classified by a hyperbox.

Table 6.11. Purity and power scores of hyperboxes for five folds of Simulated 2 dataset

			Training Phase									**Test Phase**				
			Model MOU					**After HTA**				**After HTA**				
Fold	**Iter.**	**L**	**N_{kl}**	**M_{kl}**	**Purity**	**Power**	**A_M**	**N_{kl}**	**M_{kl}**	**Power**	**A_{TB}**	**N_{kl}**	**M_{kl}**	**Purity**	**Power**	**A**
	1	1	17	2	0.88	0.57		11	0	0.37		1	0	1.00	0.12	
	1	2	28	12	0.57	0.93	75.0	14	0	0.47	0.42	3	0	1.00	0.43	66.7
	2	4	13	10	0.23	0.43	73.3	9	0	0.30	0.57	4	0	1.00	0.50	80.0
	3	3	7	1	0.86	0.23	78.3	6	0	0.20	0.67					
	4	5	9	0	1.00	0.30		4	0	0.13		0	1	-	0.00	
	4	6	3	3	0.00	0.10	81.7	2	0	0.07	0.77					**80.0**
	5	8	2	0	1.00	0.07	86.7	2	0	0.07	0.80					80.0
	6	7	5	1	0.80	0.17		2	0	0.07						
1	6	9	1	0	1.00	0.03	88.3	1	0	0.03	0.85					80.0
	7	10	2	1	0.50	0.07	90.0	2	0	0.07	0.88					80.0
	8	12	1	0	1.00	0.03	91.7	1	0	0.03	0.90					80.0
	9	11	1	0	1.00	0.03		1	0	0.03						
	9	13	1	0	1.00	0.03	93.3	1	0	0.03	0.93					80.0
	10	15	1	0	1.00	0.03	96.7	1	0	0.03	0.95					80.0
	11	14	1	0	1.00	0.03	98.3	1	0	0.03	0.97					80.0
	12	16	1	0	1.00	0.03		1	0	0.03						
	12	17	1	0	1.00	0.03	100	1	0	0.03	1.00					80.0
	1	2	21	6	0.71	0.68	75.0	19	0	0.61	0.37	3	0	1.00	0.38	80.0
	2	1	21	15	0.29	0.72		16	0	0.55		4	0	1.00	0.57	
	2	3	2	0	1.00	0.07	65.0	2	0	0.07	0.67	1	0	1.00	0.13	80.0
	3	4	5	2	0.60	0.17	0.0	4	0	0.14		3	1	0.67	0.38	
	3	5	6	1	0.83	0.19	78.3	5	0	0.16	0.77					80.0
	4	6	5	0	1.00	0.17	91.7	5	0	0.00	0.86					73.3
2	5	8	3	0	1.00	0.10	85.0	3	0	0.10	0.90	0	1	-	0.00	**73.3**
	6	7	1	0	1.00	0.03		1	0	0.03						
	6	9	2	0	1.00	0.07	96.7	2	0	0.07	0.95					73.3
	7	11	1	0	1.00	0.03	98.3	1	0	0.03	0.97					73.3
	8	10	1	0	1.00	0.04		1	0	0.04						
	8	12	1	0	1.00	0.03	100	1	0	0.03	1.00					73.3

Table 6.11 (continued)

			Training Phase									Test Phase				
			Model MOU					After HTA				After HTA				
Fold	**Iter.**	**L**	**N_{kl}**	**M_{kl}**	**Purity**	**Power**	**A_M**	**N_{kl}**	**M_{kl}**	**Power**	**A_{TB}**	**N_{kl}**	**M_{kl}**	**Purity**	**Power**	**A**
	1	2	21	6	0.71	0.68	73.0	17	0	0.55	0.28	3	0	1.00	0.38	87.0
	2	1	23	8	0.65	0.79	73.0	16	0	0.55	0.55	3	0	1.00	0.43	87.0
	3	4	9	4	0.56	0.31	80.0	5	0	0.17	0.63	2	0	1.00	0.25	73.0
	4	3	5	1	0.80	0.16	78.0	5	0	0.16	0.72	0	1	N/A	0.00	73.0
	5	5	4	1	0.75	0.14	85.0	4	0	0.14	0.78	1	0	1.00	0.13	80.0
	6	7	2	2	0.00	0.07		1	0	0.03						
	6	6	4	2	0.50	0.13	88.0	3	0	0.10	0.85	1	0	1.00	0.13	87.0
3	7	8	3	1	0.67	0.10	93.0	2	0	0.07	0.88					80.0
	8	9	2	1	0.50	0.07		1	0	0.03						
	8	10	1	0	1.00	0.03	93.0	1	0	0.03	0.92					80.0
	9	11	1	0	1.00	0.03	95.0	1	0	0.03	0.93					80.0
	10	13	1	0	1.00	0.03	97.0	1	0	0.03	0.95	1	0	1.00	0.13	**93.0**
	11	12	1	0	1.00	0.03	98.0	1	0	0.03	0.97					93.0
	12	14	1	0	1.00	0.03		1	0	0.03						93.0
	12	15	1	0	1.00	0.03	100.0	1	0	0.03	1.00					93.0
	1	1	22	4	0.82	0.73		16	0	0.53		3	0	1.00	0.38	
	1	2	26	6	0.77	0.87	80.0	18	0	0.600	0.57	4	1	0.75	0.57	80.0
	2	4	5	0	1.00	0.17	77.0	5	0	0.17	0.65					80.0
	3	3	6	0	1.00	0.20	83.0	6	0	0.20	0.75					73.0
	4	5	5	1	0.80	0.17	90.0	4	0	0.13	0.82	0	1	-	0.00	80.0
4	5	6	5	1	0.80	0.17	93.0	3	0	0.10	0.87					73.0
	6	8	2	0	1.00	0.07	93.0	2	0	0.07	0.90					73.0
	7	7	2	0	1.00	0.07		2	0	0.07		0	1	-	0.00	73.0
	7	9	2	0	1.00	0.07	97.0	2	0	0.07	0.97	0	1	-	0.00	**73.0**
	8	10	1	0	1.00	0.03		1	0	0.03						
	8	11	1	0	1.00	0.03	100.0	1	0	0.03	1.00					73.0
	1	1	20	3	0.85	0.67		14	0	0.47		4	1	0.75	0.50	73.0
	1	2	27	9	0.67	0.90	78.0	19	0	0.63	0.55	3	1	0.67	0.43	
	2	3	6	0	1.00	0.20	72.0	6	0	0.20	0.65	1	0	1.00	0.13	73.0
	3	4	4	0	1.00	0.13	82.0	4	0	0.13	0.72	1	1	0.00	0.13	**73.0**
	4	5	6	1	0.83	0.20		4	0	0.13						
	4	6	5	3	0.40	0.17	88.0	3	0	0.10	0.83					73.0
5	5	7	2	2	0.00	0.07		2	0	0.07						
	5	8	2	2	0.00	0.07	88.0	1	0	0.03	0.88					73.0
	6	9	2	0	1.00	0.07		2	0	0.07						
	6	10	1	1	0.00	0.03	92.0	1	0	0.03	0.93					73.0
	7	12	1	0	1.00	0.03		1	0	0.03	0.95					73.0
	8	11	2	0	1.00	0.07		2	0	0.07						
	8	13	1	0	1.00	0.03	100.0	1	0	0.03	1.00					73.0

M, O, U : Number of misclassified, overlap, and unassigned samples, respectively.
A_M: Model-MOU accuracy, **A_{TB}**: accuracy of fixed samples, **A**: Test phase accuracy, all in percentages

The hyperboxes are first generated by Model-MOU and then trimmed by HTA. In the test phase trimmed hyperboxes with a purity score of 1 is used. Thus, purity metric is not directly relevant in the test phase. However, when HTA is applied to a hyperbox, power of that box decreases, and purity is an indicator for the amount of this decrease. A box with a higher purity score is expected to have smaller power loss after HTA.

In preliminary experiments, a decreasing trend was observed on the power of the generated hyperboxes as the number of iterations increases. The results given above also support that observation. Moreover, only the hyperboxes of earlier iterations are used for classification in the test phase. In Table 6.11, the test accuracies written in bold correspond to the last iteration where a test sample falls into a generated hyperbox. These test accuracies are equal to the test accuracy at the final iteration. It is not always guaranteed to have such kind of an equality, but it is expected to have close values since subsequent hyperboxes have decreasing classification power.

Iteration details and purity and power scores of hyperboxes generated by HCB for Simulated 4, Breast Cancer, and Skin Segmentation datasets are given in Appendix F and Appendix G, respectively. The results are summarized and discussed below.

Results for Simulated 4 Dataset

The first of *Simulated 4* dataset was already tested in Section 6.1. HCB algorithm is applied for the remaining 4 folds of the dataset. Training and test phase accuracies for the dataset are reported in Table 6.12.

Table 6.12. Training and test phase accuracies throughout the iterations of HCB – Simulated 4 dataset

	c1		**c2**		**c3**		**c4**		**c5**	
Iter.	**Training**	**Test**	**Training**	**Test**	**Training**	**Test**	**Training**	**Test**	**Training**	**Test**
1	73.00	64.00	71.00	76.00	78.00	80.00	75.00	56.00	73.00	68.00
2	81.00	76.00	76.00	72.00	79.00	80.00	82.00	72.00	75.00	80.00
3	82.00	64.00	76.00	76.00	82.00	72.00	82.00	64.00	79.00	84.00
4	79.00	72.00	81.00	72.00	88.00	76.00	85.00	72.00	83.00	84.00
5	87.00	68.00	82.00	72.00	91.00	84.00	86.00	76.00	89.00	88.00
6	94.00	72.00	84.00	80.00	95.00	**88.00**	92.00	76.00	92.00	84.00
7	96.00	**80.00**	89.00	84.00	97.00	**88.00**	94.00	76.00	95.00	**92.00**
8	98.00	**80.00**	89.00	84.00	99.00	**88.00**	99.00	**84.00**	97.00	88.00
9	99.00	**80.00**	94.00	**96.00**	100.0	**88.00**	100.0	**84.00**	99.00	**92.00**
10	100.0	**80.00**	98.00	88.00	-	-	-	-	100.0	**92.00**
11	-	-	100.0	**96.00**	-	-	-	-	-	-

On the average, 9.8 iterations are performed and 13.8 hyperboxes are used to achieve 100% training phase accuracy. The best test accuracy is obtained as 96% for the 2nd fold and Figure 6.7 illustrates the test phase with generated hyperbox classifier in this fold. There are 3 overlap samples and 2 unknown samples in the test phase, and only the sample (41,11) is misclassified after applying COUS.

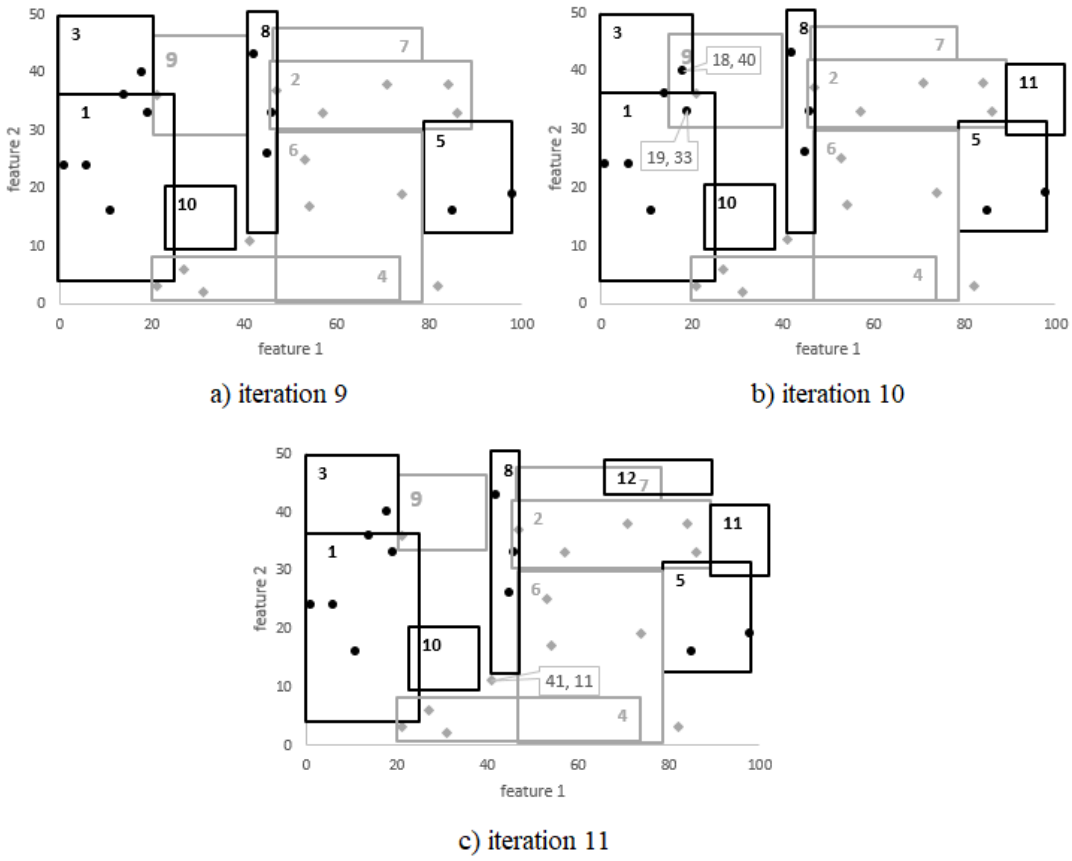


Figure 6.7. Test phase illustration for the second fold of Simulated 4 dataset

In Figure 6.7, hyperboxes are numbered in generation order. The last three hyperboxes 10, 11, and 12 are not used in the test phase, and the highest power scores are obtained with the first two hyperboxes. 96% test accuracy is first obtained when there are 10 hyperboxes in iteration 9, after trimming hyperbox 10. In iteration 10, 11th hyperbox is added to Model-MOU and selected to be trimmed. New boundaries of free hyperbox 9 cover sample (18, 40) and sample (19, 33). It decreases the test accuracy to 88% for this iteration. Before the final iteration, hyperbox 11 is trimmed and hyperbox 12 is added to the model. Finally, boundaries of hyperbox 9 is again changes yielding 96% test accuracy in the last iteration.

Results for Breast Cancer Dataset

Breast cancer dataset was already tested in Section 6.1 for its first fold. HCB algorithm is applied for the remaining folds of the dataset. On the average, it takes 12.4 iterations and 16.8 hyperboxes to reach 100% training accuracy. In most of the iterations, Model MOU cannot reach the optimal solution and terminates due to the 20-minute time limit. Although, this situation is not an obstacle to reach the 100% classification accuracy at the end of algorithm, it is an indicator that Model-MOU can resolve complexity due to the sample size, but it is not very efficient when the number of features is considered. On the average, it takes 12.4 iterations and 16.6 hyperboxes are generated to reach 100% training accuracy.

Similar to the results of simulated datasets, power of the hyperboxes decreases throughout the iterations as it is illustrated in Figure 6.8. Approximately after the 10^{th} hyperbox (according to the generation order) the power of the hyperboxes converges to zero, and they are also inefficient classifiers for the test phase.

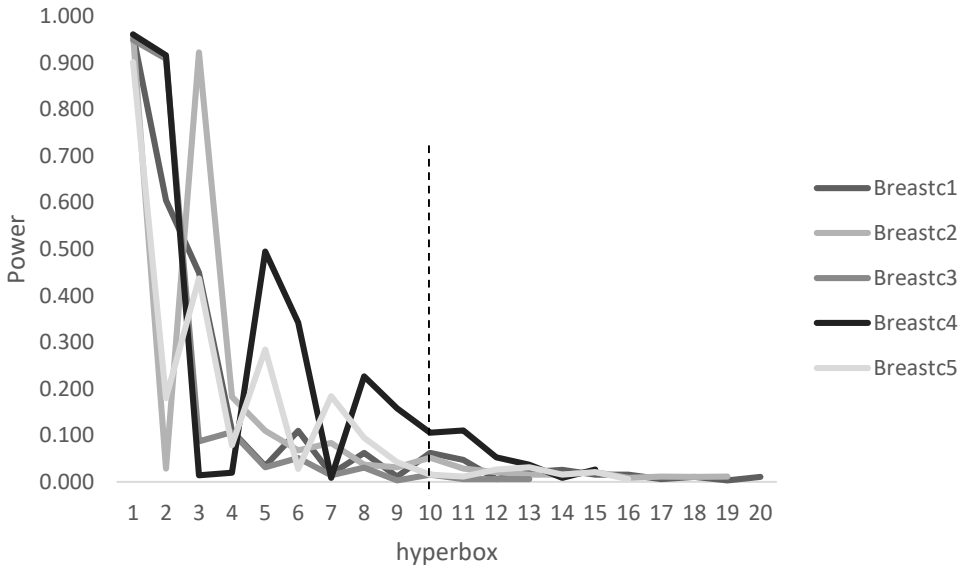


Figure 6.8. Power of hyperboxes for five folds of Breast Cancer dataset

In Table 6.13, training and test accuracies for the five folds of *Breast Cancer* dataset is given. The test accuracies shown in bold corresponds to the iterations where 10 hyperboxes are generated. When the final test accuracies are compared with the test accuracies with 10 hyperboxes, inefficiency of those hyperboxes is obviously seen.

Table 6.13. Training and test phase accuracies throughout the iterations of HCB – Breast Cancer dataset

	c1		**c2**		**c3**		**c4**		**c5**	
Iter.	**Training**	**Test**	**Training**	**Test**	**Training**	**Test**	**Training**	**Test**	**Training**	**Test**
1	92.49	93.42	94.87	93.33	93.41	92.75	93.96	93.48	93.77	95.62
2	89.93	94.89	93.96	92.59	90.48	94.20	94.32	93.48	93.41	95.62
3	85.9	93.43	84.80	94.81	94.87	92.75	94.69	94.20	95.42	94.89
4	89.38	95.62	87.73	94.81	96.52	94.20	94.87	95.65	95.60	95.62
5	91.39	95.62	89.56	95.56	97.07	94.20	94.87	94.20	97.07	94.89
6	93.59	**95.62**	91.39	94.07	97.80	93.48	94.87	**94.93**	96.34	94.16
7	95.79	95.62	92.31	94.81	98.53	**93.48**	95.24	95.65	97.25	**94.89**
8	96.89	96.35	93.22	**94.81**	99.63	93.48	96.52	95.65	97.80	94.89
9	97.99	95.62	95.05	94.81	100.00	93.48	98.17	95.65	98.17	94.89
10	98.53	95.62	95.05	94.81	-	-	99.08	95.65	99.08	94.89
11	99.08	95.62	97.44	94.81	-	-	100.00	95.65	99.45	94.89
12	99.45	95.62	97.44	94.81	-	-	-	-	99.82	94.89
13	100	95.62	97.99	94.81	-	-	-	-	100.00	94.89
14	-	-	99.08	94.81	-	-	-	-	-	-
15	-	-	99.63	94.81	-	-	-	-	-	-
16	-	-	100.00	94.81	-	-	-	-	-	-

Results for Skin Segmentation Dataset

In the experiments with five folds of *Skin Segmentation* subsets, as expected, the easy sets consume less CPU time compared to the hard sets. Moreover, the easy/hard distinction becomes more effective on the computational time than the sample size. In Table 6.14, total CPU times for Model-MOU are given for all folds of subset 1 (hard) and subset 4 (easy). Although the sample size of subset 4 is twice as large as that of subset 1, it requires less time on the average. In fact, when the CPU times per iteration are analyzed, it is seen that in general an iteration takes longer time when there are more samples in the dataset. However, the easiness of the dataset results in reaching the 100% training accuracy in fewer iterations, yielding shorter completion times in total.

Table 6.14. CPU times (sec) for Skin subset 1 and Skin subset 4 datasets

	subset 1	**subset 4**
c1	9290.63	3611.40
c2	3645.03	1705.70
c3	3503.73	3277.05
c4	1674.60	2577.90
c5	2317.30	4519.60
average	4086.26	3138.33

For *skin subset 1*, training and test accuracies throughout the iterations of the algorithm are summarized in Table 6.15. On the average, 13.4 hyperboxes are generated in 9.2 iterations. For all folds, the best test accuracy is obtained before hitting the last iterations. Moreover, the best test accuracy among all folds is 100%, and it is not obtained as a final iteration result.

Table 6.15. Training and test phase accuracies throughout the iterations of HCB - Skin subset 1

	c1		**c2**		**c3**		**c4**		**c5**	
Iter.	**Training**	**Test**	**Training**	**Test**	**Training**	**Test**	**Training**	**Test**	**Training**	**Test**
1	71.43	65.43	71.43	76.29	71.43	82.57	71.43	71.43	28.57	68.38
2	83.50	88.86	82.36	80.86	82.14	93.14	82.29	85.43	82.29	86.61
3	92.21	92.29	83.50	86.57	94.00	90.29	83.21	91.14	86.43	82.91
4	94.00	96.57	87.64	94.86	97.36	95.14	92.93	93.14	96.86	84.90
5	94.57	96.86	92.64	93.43	82.14	98.00	97.93	98.57	97.71	98.29
6	95.07	96.57	97.93	97.43	99.14	97.71	98.86	**99.43**	97.86	**99.72**
7	96.43	98.00	98.43	98.57	99.29	**98.86**	99.64	**99.43**	98.86	**99.72**
8	97.00	98.57	98.64	99.43	100.00	**98.86**	100.00	**99.43**	100.00	**99.72**
9	97.71	**98.86**	99.14	**100.00**	-	-	-	-	-	-
10	98.50	98.57	100.00	99.72	-	-	-	-	-	-
11	99.71	**98.86**	-	-	-	-	-	-	-	-
12	100.00	**98.86**	-	-	-	-	-	-	-	-

For *skin subset 2*, it takes 4.6 iterations and 8.2 hyperboxes are generated to reach 100% training accuracy, on the average. For some folds, the best test accuracy is obtained only in final iteration as it is seen in Table 6.16. It can be said that the best performance it obtained on the 3rd fold, which results in 100% test accuracy within only 3 iterations and 6 hyperboxes.

Table 6.16. Training and test phase accuracies throughout the iterations of HCB - Skin subset 2

	c1		c2		c3		c4		c5	
Iter	**Training**	**Test**	**Training**	**Test**	**Training**	**Test**	**Training**	**Test**	**Training**	**Test**
1	85.21	90.31	86.43	83.76	73.33	90.86	86.36	87.14	85.93	90.29
2	97.36	98.29	97.71	96.87	99.71	99.43	97.86	97.43	98.43	97.43
3	98.43	98.86	98.79	96.87	100.00	**100.00**	98.50	98.57	99.57	**100.00**
4	99.64	**99.72**	99.64	96.58	-	-	99.00	98.86	100.00	**100.00**
5	100.00	**99.72**	100.00	**99.43**	-	-	99.57	98.86	-	-
6	-	-	-	-	-	-	100.00	**100.00**	-	-

Results for *Skin subset 3* are summarized in Table 6.17. It takes 19.6 iterations and 28.8 hyperboxes on the average to reach 100% training phase accuracy. For the first fold of the dataset a test accuracy above 99% is achieved.

Table 6.17. Training and test phase accuracies throughout the iterations of HCB - Skin subset 3

	c1		c2		c3		c4		c5	
Iter	**Training**	**Test**	**Training**	**Test**	**Training**	**Test**	**Training**	**Test**	**Training**	**Test**
1	71.43	19.57	71.43	27.57	71.43	24.28	71.43	22.86	71.43	28.57
2	70.68	24.71	55.79	26.00	60.57	20.00	70.86	20.29	64.86	22.86
3	60.57	30.43	62.71	27.00	64.96	21.43	62.71	28.57	67.93	24.57
4	54.57	35.43	74.46	55.57	74.79	69.29	64.96	35.43	74.92	32.29
5	64.96	69.29	81.46	75.00	80.25	85.43	74.46	52.14	77.00	58.29
6	74.79	73.00	85.43	76.86	81.46	84.86	80.25	51.43	82.29	66.86
7	78.93	85.43	88.21	84.86	83.50	89.71	81.46	66.43	87.68	75.00
8	80.25	89.71	88.86	84.86	82.50	91.00	83.64	72.85	85.71	83.50
9	83.50	89.71	97.25	91.00	89.71	93.00	88.86	80.15	89.95	88.86
10	81.43	89.71	94.71	93.29	93.00	95.71	85.71	83.71	88.86	85.43
11	82.50	92.00	97.00	97.29	94.14	97.57	86.14	83.24	89.61	89.71
12	83.64	95.71	**98.86**	**98.43**	95.71	97.57	89.61	86.00	91.39	89.71
13	93.00	97.57	98.96	97.57	96.36	97.57	93.36	87.71	93.29	89.61
14	94.25	97.57	98.96	97.86	98.96	97.86	94.96	91.00	94.11	91.00
15	94.14	98.00	99.36	97.86	97.89	95.29	96.46	92.14	96.00	93.29
16	96.36	97.57	99.50	98.43	**99.50**	**98.86**	97.21	94.54	96.21	95.71
17	97.89	97.57	99.68	98.43	99.68	95.29	98.86	97.00	98.43	97.57
18	98.71	97.57	**100.00**	**98.43**	**100.00**	**95.29**	98.61	98.43	98.86	**98.71**
19	99.00	98.14	-	-	-	-	**99.64**	**99.43**	98.92	98.71
20	**100.00**	**99.14**	-	-	-	-	**100.00**	**98.43**	99.00	98.71
21	-	-	-	-	-	-	-	-	99.21	98.71
22	-	-	-	-	-	-	-	-	100.00	**98.71**

Figure 6.9 illustrates the relationship between power of hyperboxes and test accuracies for the first folds of *Skin subset 3*. As usual, test accuracy becomes fairly steady after the iteration where power values converge to zero. Here, power of a hyperbox obtained by Model-MOU and power after HTA differ as it seen in the figure, but they both have a common decreasing trend. The gap between the powers before and after HTA is mainly related with the purities of the hyperboxes. If purity is close to 1.00 then the gap is smaller, and vice versa.

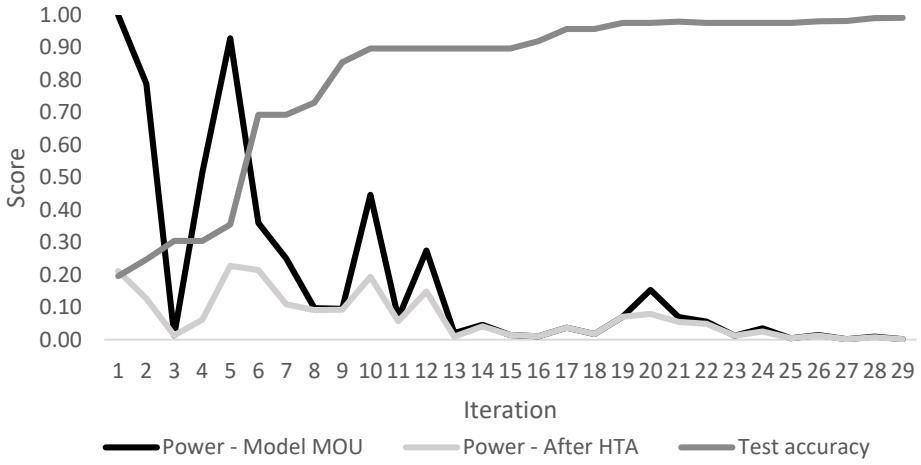


Figure 6.9. Power vs. test accuracy throughout the iterations of HCB for fold 1 of Skin subset 3

Skin subset 4 terminates in 4 iterations for all folds with an average of 5.8 hyperboxes. The accuracies on each iteration are given in Table 6.18. The best result is obtained by last fold having 100% test accuracy with only 5 hyperboxes.

Table 6.18. Training and test phase accuracies throughout the iterations of HCB - Skin subset 4

c1		**c2**		**c3**		**c4**		**c5**	
Training	**Test**	**Training**	**Test**	**Training**	**Test**	**Training**	**Test**	**Training**	**Test**
71.43	85.43	71.43	88.00	85.34	89.57	85.96	92.00	86.57	70.00
98.04	96.29	98.57	95.57	98.57	97.86	98.36	96.00	98.71	96.43
99.25	99.86	98.75	99.43	99.43	**100.00**	95.82	98.29	98.96	99.86
100.00	**100.00**	100.00	**99.86**	100.00	**100.00**	100.00	**100.00**	100.00	**100.00**

On the average, it takes 6.4 iterations and 9.2 hyperboxes to reach 100% training accuracy for *Skin subset* 5. Except the last few iterations, Model-MOU cannot reach the optimal solution and terminates due to the 20-minute time limitation. Training and test accuracies are given in Table 6.19.

Table 6.19. Training and test phase accuracies throughout the iterations of HCB - Skin subset 5

c1		c2		c3		c4		c5	
Training	**Test**	**Training**	**Test**	**Training**	**Test**	**Training**	**Test**	**Training**	**Test**
71.43	73.50	82.14	82.43	71.43	85.57	71.43	69.79	71.43	84.64
91.16	97.21	88.93	98.14	91.39	93.50	94.23	95.43	90.30	96.43
95.71	97.43	75.00	98.64	96.16	98.00	94.93	91.29	92.39	97.79
98.89	98.64	97.96	99.14	99.14	99.29	98.25	98.71	97.59	99.79
98.89	99.57	99.09	99.00	99.98	**99.93**	99.46	99.64	99.32	99.86
99.61	97.21	99.79	**99.79**	100.00	**99.93**	100.00	**99.79**	100.00	**99.93**
100.00	**100.00**	100.00	**99.79**	-	-	-	-	-	-

Finally, when HCB is used on S*kin subset 6* It takes 13.8 iterations and 19 hyperboxes are needed on the average to reach 100% accuracy. Except the last three iterations, 20 minute of time limit is reached in each iteration of all folds. Accuracy of the iterations are given in Table 6.20 The best test accuracy is 100% using the generated 15 hyperboxes in the first fold.

Table 6.20. Training and test phase accuracies throughout the iterations of HCB - Skin subset 6

	c1		c2		c3		c4		c5	
Iter	**Training**	**Test**	**Training**	**Test**	**Training**	**Test**	**Training**	**Test**	**Training**	**Test**
1	71.43	64.43	71.43	63.43	71.43	67.38	71.43	62.43	71.43	59.57
2	76.86	87.21	76.79	74.51	78.13	72.55	78.63	65.57	76.79	63.71
3	77.70	81.86	78.50	72.28	87.43	75.00	80.35	61.17	77.70	67.00
4	78.91	89.14	79.04	79.57	89.34	74.14	83.93	76.14	79.04	73.00
5	87.46	89.71	82.20	81.00	92.85	79.14	85.71	82.29	82.20	79.71
6	97.66	98.00	84.36	82.14	93.07	81.14	88.43	82.71	87.43	77.43
7	98.09	98.29	86.11	90.48	94.02	81.14	91.02	85.43	85.71	81.29
8	98.70	98.50	90.48	97.29	94.75	82.29	92.86	86.57	87.46	89.71
9	99.64	99.29	91.54	98.50	96.71	89.14	96.54	90.14	90.48	91.00
10	99.86	99.64	96.43	99.64	98.21	89.14	98.43	96.29	92.86	88.29
11	**100.00**	**100.00**	98.83	99.64	98.92	90.48	98.75	98.83	94.75	93.29
12	-	-	**100.00**	**99.93**	99.02	97.29	98.83	98.83	96.64	97.29
13	-	-	-	-	99.15	98.83	98.87	98.83	98.92	97.29
14	-	-	-	-	99.86	89.83	99.64	**99.57**	99.64	98.83
15	-	-	-	-	**100.00**	**99.36**	**100.00**	**99.57**	99.84	**99.57**
16	-	-	-	-	-	-	-	-	**100.00**	**99.57**

6.4. Comparison with CART

Tree learning is a widely used method for classification. The classifier rules obtained by a classification tree are similar to the rules inherently defined by a hyperbox classifier. A classification tree basically starts with a root node and splits the samples into two groups of different classes, based on a bounding value of a selected feature associated with that node. Splitting of samples into subgroups continues with branching from the root node and generation of additional nodes. When the complete tree is constructed, each child node corresponds to a classifier rule, giving the bounds on the feature values. These bounds can be a lower bound, an upper bound, or an interval with both lower and upper bounds on the feature. On the other hand, a hyperbox classifier always gives both lower and upper bounds on all features. In addition, HCB aims to have a minimal set of classifier rules by the use of Model MOU.

In this section, a well-known tree learning method CART (Breimen et al., 1984) is applied to the selected datasets, and the results are compared with HCB results. As for the implementation of CART, Salford Predictive Modeler 8.2 is used during the experiments. It should be noted that comparisons are not performed under a time limitation basis as HCB takes an optimization approach and requires solving a mixed integer program.

Table 6.21 illustrates the test results of CART and HCB where 100% training accuracy is achieved for both methods. The names of the datasets and fold numbers are given in the first column. For CART, the number of classifier nodes are reported together with the test accuracies. 100% training accuracy for HCB is the result of the final iteration, and the number of hyperboxes are reported along with the test accuracies. As seen in the table, for all datasets, HCB uses fewer or the same number of hyperboxes compared to the number of nodes used by CART, except the 4^{th} fold of *Skin subset* 2. Higher of the two respective test accuracies obtained by CART and HCB are shown in bold in the table. Based on these results, it can be said that final iterations of HCB in general outperform the complete tree of CART by generating higher test accuracies using fewer classifier rules.

Table 6.21. CART vs HCB, Comparison of test accuracies for 100% training phase accuracy

	CART		HCB	
Data set	**Nodes**	**Test Accuracy**	**Hyperboxes**	**Test Accuracy**
simulated2c1	19	**0.8667**	17	0.8000
simulated2c2	12	0.6667	12	**0.7333**
simulated2c3	15	0.8667	15	**0.9333**
simulated2c4	11	0.6667	11	**0.7333**
simulated2c5	12	**0.7333**	12	**0.7333**
simulated4c1	15	0.7200	14	**0.8000**
simulated4c2	13	0.8000	12	**0.9600**
simulated4c3	17	**1.0000**	12	0.8800
simulated4c4	16	**0.8800**	17	0.8400
simulated4c5	16	**0.9800**	14	0.9200
breastc1	34	0.9489	20	**0.9562**
breastc2	30	0.9407	19	**0.9481**
breastc3	27	0.9270	13	**0.9348**
breastc4	28	**0.9565**	15	**0.9565**
breastc5	27	0.9343	17	**0.9489**
skin-s1c1	28	0.9829	18	**0.9886**
skin-s1c2	21	0.9829	10	**0.9971**
skin-s1c3	27	0.9743	8	**0.9886**
skin-s1c4	31	**0.9971**	11	0.9943
skin-s1c5	32	**0.9971**	13	**0.9971**
skin-s2c1	13	**1.0000**	9	0.9971
skin-s2c2	13	0.9886	8	**0.9943**
skin-s2c3	11	0.9943	6	**1.0000**
skin-s2c4	**7**	**1.0000**	10	**1.0000**
skin-s2c5	11	**1.0000**	8	**1.0000**
skin-s3c1	38	0.9886	29	**0.9914**
skin-s3c2	34	0.9529	25	**0.9843**
skin-s3c3	46	**0.9871**	26	0.9529
skin-s3c4	43	**0.9929**	30	0.9843
skin-s3c5	44	**0.9914**	34	0.9871
skin-s4c1	14	0.9957	6	**1.0000**
skin-s4c2	11	0.9971	5	**0.9986**
skin-s4c3	11	**1.0000**	7	**1.0000**
skin-s4c4	11	0.9986	5	**1.0000**
skin-s4c5	11	0.9971	6	**1.0000**
skin-s5c1	24	0.9993	9	**1.0000**
skin-s5c2	16	0.9957	10	**0.9979**
skin-s5c3	22	**0.9993**	9	**0.9993**
skin-s5c4	21	**0.9993**	9	0.9979
skin-s5c5	17	**0.9993**	9	**0.9993**
skin-s6c1	46	0.9964	15	**1.0000**
skin-s6c2	49	0.9979	17	**0.9993**
skin-s6c3	49	**0.9936**	21	**0.9936**
skin-s6c4	52	**0.9957**	18	**0.9957**
skin-s6c5	54	**0.9993**	20	0.9957

Table 6.22. CART vs. HCB, Comparison of test accuracies for best test phase accuracy

	CART			HCB		
Data set	**Nodes**	**Training Accuracy**	**Best Test Accuracy**	**hyperboxes**	**Train Accuracy**	**Best Test Accuracy**
simulated2c1	19	1.0000	**0.8667**	4	0.7333	0.8000
simulated2c2	2	0.7500	0.7333	2	0.7500	**0.8000**
simulated2c3	2	0.7333	0.8667	13	0.9667	**0.9333**
simulated2c4	7	0.9667	0.6667	2	0.8000	**0.8000**
simulated2c5	8	0.9167	**0.7333**	2	0.7833	**0.7333**
Simulated4c1	7	0.9400	**0.8000**	11	0.9600	**0.8000**
Simulated4c2	8	0.9700	0.8400	10	0.9400	**0.9600**
Simulated4c3	17	1.0000	**1.0000**	9	0.9500	0.8800
Simulated4c4	16	1.0000	**0.8800**	15	0.9900	0.8400
Simulated4c5	10	0.8300	**0.9800**	10	0.9500	0.9200
breastc1	6	0.8768	**0.9562**	6	0.9139	**0.9562**
breastc2	19	0.9638	**0.9556**	7	0.8956	**0.9556**
breastc3	11	0.9348	**0.9420**	4	0.9048	**0.9420**
breastc4	17	0.9638	**0.9710**	7	0.9487	0.9565
breastc5	6	0.8986	**0.9638**	2	0.9377	0.9562
skin-s1c1	15	0.9950	**0.9886**	12	0.9771	**0.9886**
skin-s1c2	**9**	0.9900	0.9857	11	0.9914	**1.0000**
skin-s1c3	10	0.9929	0.9743	10	0.9929	**0.9886**
skin-s1c4	31	1.0000	**0.9971**	8	0.9886	0.9943
skin-s1c5	21	0.9964	**1.0000**	8	0.9786	0.9972
skin-s2c1	13	1.0000	**1.0000**	7	0.9964	0.9972
skin-s2c2	13	1.0000	0.9886	8	1.0000	**0.9943**
skin-s2c3	9	0.9986	0.9943	6	1.0000	**1.0000**
skin-s2c4	6	0.9993	**1.0000**	10	1.0000	**1.0000**
skin-s2c5	11	1.0000	**1.0000**	8	1.0000	**1.0000**
skin-s3c1	30	0.9986	0.9900	29	1.0000	**0.9914**
skin-s3c2	34	1.0000	0.9529	21	0.9886	**0.9843**
skin-s3c3	26	0.9950	**0.9886**	22	0.9950	**0.9886**
skin-s3c4	29	0.9968	**0.9943**	29	0.9968	**0.9943**
skin-s3c5	29	0.9939	**0.9914**	30	0.9886	0.9871
skin-s4c1	6	0.9971	0.9957	6	1.0000	**1.0000**
skin-s4c2	7	0.9989	0.9971	5	1.0000	**0.9986**
skin-s4c3	7	0.9993	**1.0000**	5	0.9994	**1.0000**
skin-s4c4	7	0.9993	0.9986	5	1.0000	**1.0000**
skin-s4c5	10	0.9996	0.9971	6	1.0000	**1.0000**
skin-s5c1	24	1.0000	0.9993	9	1.0000	**1.0000**
skin-s5c2	16	1.0000	0.9957	9	0.9979	**0.9979**
skin-s5c3	22	1.0000	**0.9993**	7	0.9998	**0.9993**
skin-s5c4	21	1.0000	**0.9993**	9	1.0000	0.9979
skin-s5c5	17	1.0000	**0.9993**	9	1.0000	**0.9993**
skin-s6c1	46	1.0000	0.9964	15	1.0000	**1.0000**
skin-s6c2	31	0.9986	0.9979	17	1.0000	**0.9993**
skin-s6c3	49	1.0000	**0.9936**	21	1.0000	**0.9936**
skin-s6c4	30	0.9980	**0.9964**	18	0.9964	0.9957
skin-s6c5	34	0.9984	**0.9993**	20	0.9984	0.9957

As mentioned before, it is possible to have higher test accuracies in earlier iterations with HCB. On the other hand, CART can also obtain better test results with trees having lower training accuracies. CART follows a forward growing tree to reach 100% training accuracy, but it can also perform backward pruning on the developed full tree. To decide on pruning node, it checks the resulting test accuracy. Enumerating all possible pruning operations, it ends up with a smaller tree which has lower training accuracy but higher test accuracy. In Salford Predictive Modeler, the pruned tree having the best test accuracy is named as optimal tree. The best test accuracies throughout the iterations for CART and HCB are also compared along with the corresponding training accuracies. This best case comparison is given in Table 6.22. As it is seen in the table, best possible test accuracy obtained by HCB outperforms CART for most of the instances. In addition, number of nodes are less than number of generated hyperboxes only for six instances.

CHAPTER 7

FEATURE SELECTION

Experiments conducted so far highlight the challenge of obtaining optimal results due to the computational complexity. The HCB algorithm reduces the number of samples and keeps the number of hyperboxes under control. However, the number of features is still a significate source of problem complexity.

It is possible to determine the relevant features and an importance order of the features based on the results of the mathematical models, as it will be described in Section 7.1. However, there is a paradoxical situation such that all features must be used in the model to derive the importance order, but it is not possible to obtain the optimal results when all features are used for most of the datasets. Thus, the HCB algorithm is revised such that the features are included in the model iteratively in a systematic manner as it will be explained in Section 7.2. This new version of HCB, which is capable of selecting features, is called as HCB-f and facilitates obtaining solutions for classification problems in larger sizes.

7.1. Feature Importance Order Based on the Model Results

The proposed mathematical models Model-MOB, Model-MO, and Model-MOU allow overlapping of hyperboxes with a penalty in the objective function. The main reason for this is to handle the tradeoff involving misclassification, overfitting, and time complexity as mentioned in Section 4.1. Furthermore, when the solutions of the problem instances are analyzed, it is observed that the decision variables O_{ilm} related with the overlap samples are indicators of the irrelevant features.

An example with two features and three boxes is illustrated in Table 7.1. There are five samples, the first four of them are assigned to hyperbox 2, and the last one is assigned to hyperbox 3. $O_{112} = 1$, which means that sample 1 also falls into hyperbox 1 in terms of feature 2. However, $O_{111} = 0$ indicates that sample 1 is not within the

boundaries of hyperbox 1 in terms of feature 1, therefore there really is not an overlap. In order to detect that there is an overlap, both O_{il1} and O_{il2} must be equal to 1. In the example, samples 2 and 3 are both in the overlapping region of hyperboxes 1 and 2. Although $O_{il2} = 1$ for samples 1, 4, and 5, they are not in the overlapping region since $O_{il1} = 0$. In other words, feature 2 is irrelevant or ineffective for the classification of samples 1, 4, and 5. This property can be used to rank the features with respect to their effectiveness on classification.

Table 7.1. Example for overlap decision variables

Sample	Hyperbox	O_{il1}	O_{il2}	Actual hyperbox assignment
1	**1**	0	1	2
	3	0	0	
2	**1**	1	1	2
	3	0	0	
3	**1**	1	1	2
	3	0	1	
4	**1**	0	1	2
	3	0	0	
5	**1**	0	1	3
	2	0	1	

For this purpose, the algorithm given in Figure 7.1 is executed for all features starting with the one which has the highest overlap summation rank, since the largest summation indicates the highest overlap in terms of that feature. $O_{ilm} = 1$ shows that feature m does not have an effect on classification of sample i. In case of $O_{ilm} = 0$, it is checked whether any other feature j is also effective on classification of sample i. If $O_{ilm} = 0$ for any of $j \neq m$ and, then feature m is not effective in classifying sample i. If a feature is effective for a number of samples (represented by *'test'* in the algorithm) less than a threshold, then that feature can be removed from the dataset. On the other hand, it is possible to rank the features with respect to their test values to have an idea about the relative importance of the features.

```
1   Step 1. Sum_m = Σ_{i=1}^{N} Σ_{l=1}^{L} O_ilm      for m = 1,…,M

2   Step 2. Sort Sum_m in descending order.
3           rank_m = order of m
4           iter = 1
5           treshold = 0
6   Step 3. Select feature m with rank_m = iter.
7           test = 0
8           effective_i = 0
9           For i = 1 to N
10                  For l = 1 to L
11                      If O_ilm = 0
12                          effective_i = 1
13                          Exit For l
14                      End If
15                  Next l
16                  If effective_i = 1
17                      For j = 1 to M
18                          If j ≠ m
19                              For l = 1 to L
20                                  If O_ilj = 0
21                                      effective_i = 0
22                                      Exit For j
23                                  End If
24                              Next l
25                          End If
26                      Next j
27                  End If
28                  test = test + effective_i
29          Next i
30          If test < threshold
31              Feature m is removed
32          End if
33          iter = iter +1
34          repeat Step 3 If iter < M
```

Figure 7.1. Pseudo code for feature importance algorithm

Some experiments are performed to see the performance of the algorithm. First of all, a redundant feature *z* is included in the *Simulated 1* dataset as the third feature. This feature involves binaries, and both of the classes 1 and 2 include the same number of samples with zeros and ones. The resulting dataset is illustrated in the Figure 7.2. As seen in the figure, the classes cannot be differentiated on the 3rd axis. When Model-MOB is run with all three features, it is seen that $O_{il3} = 1$ for all samples and boxes which verifies that feature z is irrelevant.

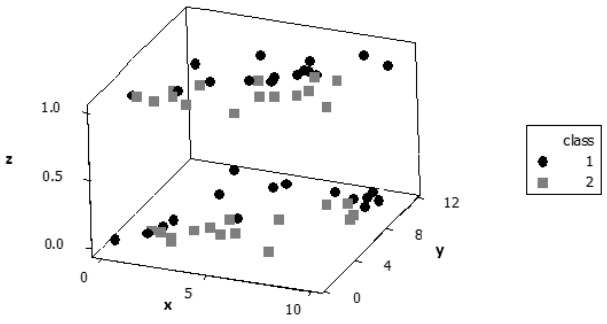


Figure 7.2. Simulated 1 dataset with redundant feature z

As another experiment, a new binary feature is introduced to *simulated 2* dataset. This time value of feature *z* is equal to 1 for all class 1 samples, and it is equal to 0 for all class 2 samples. Figure 7.3 illustrates this modified dataset.

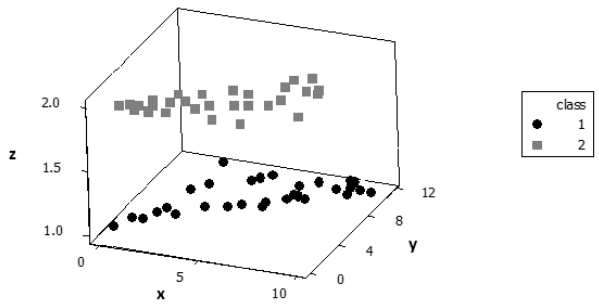


Figure 7.3. Simulated 1 dataset with new feature z

As seen in the figure, the classes are well separated along the *z*-axis. When the model is run with these three features, it converged to optimality within seconds and $O_{ilm} = 1$ for all samples and boxes for both $m = 1$ and $m = 2$. This shows that the two original features become irrelevant when a discriminative third feature is added.

These two extreme examples show the ability of the algorithm to determine when the features are absolutely irrelevant. However, for most datasets, it may not be possible to find an absolutely irrelevant feature. For those cases, a positive threshold value may be required to declare a feature is redundant.

Another experiment is conducted to see whether or not the algorithm ranks the features in the order of their relevance. Using the solution for the Wine dataset (from section 4.3), the 13 features are ranked based on the '*test*' values of the algorithm given in Figure 7.1. The lowest ranked feature is the least important, so it is removed from the dataset and the model is run with the remaining 12 features. Then, the next lowest ranked feature is also removed and the model is run with 11 features. This process is repeated until a single feature having the highest rank remains.

Table 7.2. Order of feature importance of Wine dataset in training phase

# of features	**# of boxes**	**# of misclassification**	**# of overlaps**	**Accuracy**	**Last removed feature no**
1	6	31	0	0.78	4
2	3	24	0	0.83	10
3	4	14	4	0.87	3
4	3	13	10	0.84	13
5	3	8	15	0.84	6
6	4	8	2	0.93	2
7	3	8	5	0.91	11
8	3	6	5	0.92	7
9	3	1	22	0.84	12
10	4	1	2	0.98	9
11	3	2	8	0.93	5
12	4	0	7	0.95	8
13	3	0	7	0.95	

The results of these runs are given in Table 7.2. According to the results highest accuracy in the training phase is 98% and it can be achieved without the use of features 5, 8 and 9. In addition, a relatively close accuracy of 93% can also be reached by using only six features: 1, 4, 10, 3, 13 and 6. From this point of view, this feature ranking method seems to be working. However, it should maintain the consistency in the test phase of classification. Results of the test phase presented in Table 7.3 show that use of the same six features provides 92% accuracy.

Table 7.3. Test results for feature importance order of Wine dataset

# of features	# of boxes	# of misclassification	# of overlaps	# of uncovered	After COUS	Accuracy (A)
1	6	10	2	0	11	0.69
2	3	8	1	0	8	0.78
3	4	4	3	3	7	0.81
4	3	3	5	4	9	0.75
5	3	1	4	6	3	0.92
6	4	1	1	8	3	0.92
7	3	4	0	7	4	0.89
8	3	1	0	7	2	0.94
9	3	1	3	7	4	0.89
10	4	0	1	11	2	0.94
11	3	0	2	10	2	0.94
12	4	0	1	10	1	0.97
13	3	0	1	12	2	0.94

Figure 7.4 illustrates the fluctuation in the accuracy depending on the number of features that are used in classifying the dataset. When the number of features is less than five, the gap between the accuracies of the training and test phases is higher. When more than five features are used, it is seen that the training and test accuracies settle down. The graphic also highlights the tradeoff between the effect of adding a feature and the run time complexity. For example, when the 9th important feature (originally feature 7) is added, the accuracy significantly decreases. The reason for this is that Model-MOU cannot be solved to optimality due to run time complexity.

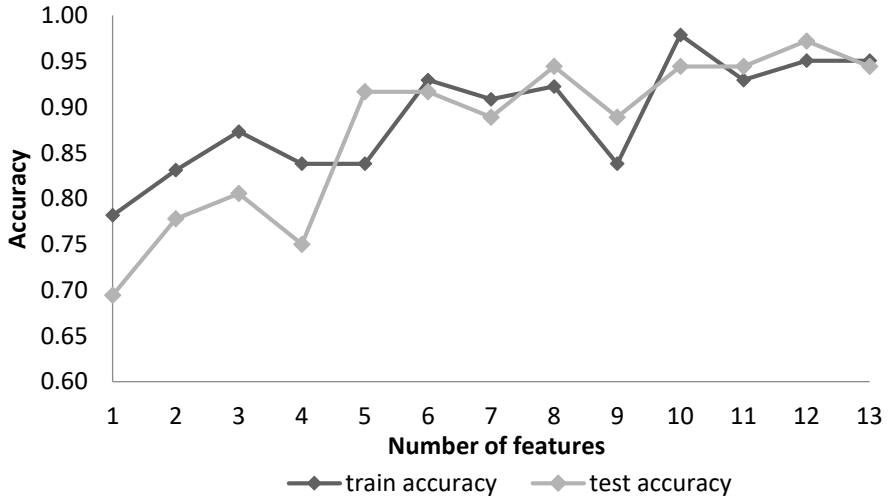


Figure 7.4. Training and test phase accuracies vs. number of features

After observing these promising results concerning the feature selection, five fold cross validation is performed for the *Wine* dataset. Training and test accuracy values of five folds of cross validation are presented as box plots in Figures 7.5 - 7.6 for each of the 13 runs. In the figure, x-axis stands for the number of features used in the solution, and the ID of the lastly added feature is given above the boxes. For both training and test phases, eliminating the least important three features 5, 8, and 9 does not significantly decrease the classification accuracy. This stability continues even after elimination of five more features. However, it is important to provide a consistent feature set for both training and test phases. Thus, considering the Figure 7.5 and 7.6, use of six features (1, 4, 10, 3, 13, 6) may be adequate for the *wine* dataset.

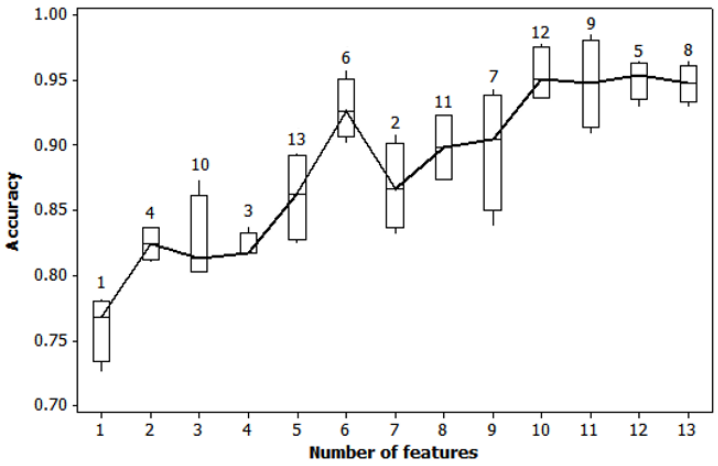


Figure 7.5. 5-fold results for comparison of runs of Wine dataset – training phase

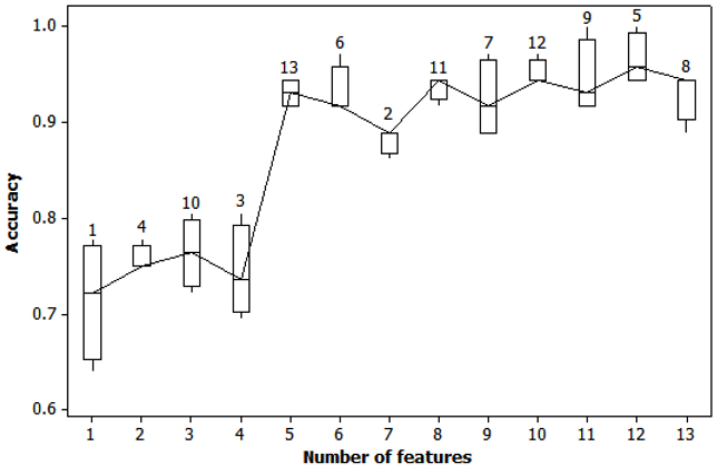


Figure 7.6. 5-fold results for comparison of runs of Wine dataset – test phase

This approach to feature ranking seems to be useful. It is also consistent between the training and test phases. However, it requires solving the model starting with all features, which makes the approach intractable. Therefore, we turn our attention to revising HCB for solving the feature selection problem.

7.2. HCB-f: Hyperbox Classification for Binary Datasets with Feature Selection

Since it is not possible to obtain even a feasible solution for large sized problems, it is difficult to use the procedure in Section 7.1 to obtain a feature importance order. Because of this reason, a different approach is taken to order the features in terms of their explanatory power for classification. Model-MOU is solved with one feature at a time using one hyperbox for each class for all features of the dataset. The features are then ordered according to objective function values where the one that has the minimum value is the most explanatory one. This feature ranking can be considered as an initialization phase of the algorithm described below, and it is repeated for training datasets of all folds in the experiments. In this way, none of the test samples can affect the feature selection or the classifier, as proposed as a non-contaminated protocol by Kuncheva and Rodrigues (2018).

The Hyperbox Classification for Binary class datasets with Feature selection (HCB-f) algorithm starts with the solution obtained by using the most important feature only by allocating one hyperbox for each class. In the rest of the algorithm, the main idea is making a decision on either adding new hyperbox(es) or adding a new feature. HCB-f checks both of these decisions only for the startup iteration. Figure 7.7 illustrates an example for the startup iteration of HCB-f. In the root node, '1F' represents that only one feature is used in the model, and z_1^* is the respective objective function value.

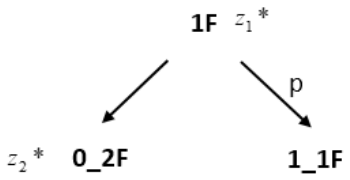


Figure 7.7. Startup iterations for HCB-f

In the tree representation, moving down on the right hand side branch corresponds to applying the HTA algorithm for a solution *S* by adding hyperboxes as in HCB. In the figure, *p* samples are fixed for the trimmed hyperbox in solution *S*. The node '1_1F', means that a trimming operation has been performed and still one feature is in use. However, it is not required to solve Model-MOU for this node yet. If we move down on the left hand side branch of the tree, node '0_2F' represents that hyperbox allocation scheme is not changed (0 trimming operation has been performed), but the two most important features are used in Model-MOU with respective objective function value of z_2^*. Based on the solutions, contributions of adding a feature or trimming a hyperbox are compared. If the number of correctly classified samples within trimmed hyperboxes (reduction in sample size as a result of fixing hyperbox assignments) is greater than or equal to the increase in the number of correctly classified samples after adding a new feature, i.e. $p \geq z_1^* - z_2^*$, then the algorithm moves in the direction of hyperbox addition. Otherwise it chooses to move in the direction of feature addition.

The pseudo code for HCB-f is given in Figure 7.8. The initialization procedure between lines 1 and 12 stands for the feature ranking and defining sets. The startup procedure between lines 13 and 22 represents the first branching described above. The main procedure between lines 23 and 43 is repeated until 100% training accuracy is achieved. Here, the number of features included is controlled at line 24 in the second while loop since the algorithm turns into HCB after all features are included in Model-MOU. Once the algorithm decides to move in the direction of adding hyperboxes, the loop between the lines 25 and 32 is executed. $\Delta z = z_1^* - z_2^*$ is recorded at that level, and only the number of samples to be fixed (*P*) is considered and compared with the recorded Δz value for the upcoming iterations. If the number of the samples that can be fixed is beaten by Δz at any iteration, then the direction changes, and the algorithm starts the loop between the lines 33 and 40. The *p* value at that level is recorded, and the algorithm starts adding a new feature in each new iteration. This time, the Δz value is updated throughout the iterations. If Δz value is smaller than the recorded *p* value at any iteration, then the direction changes again and the algorithm goes back to adding hyperboxes.

Initialization Procedure

For m = 1 to M
    SOLVE Model-MOU with m^{th} feature only and one hyperbox per class
    Let the objective function value be o_m^*
Next m
F = {f1,f2,…,fM}; list of features in ascending order of o_m^*
F' = {f1}
S : Solution of Model-MOU
D : Set of samples
P : Set of fixed samples; $P = \{\}$, $P \subset D$
P' : Set of candidate samples to be fixed
H : Set of hyperboxes, 1 hyperbox per class initially
H' : Set of candidate hyperboxes to add after HTA

Startup Procedure

$F = F - \{f_1\}$
SOLVE Model-MOU with feature set $F' \rightarrow S, z^*$
$z_1^* = z^*$
CALL HTA for solution $S \rightarrow P'$, H'
$p = |P'|$
SOLVE Model-MOU with $F' \cup \{f_2\}$ and $H \rightarrow z^*$
$z_2^* = z^*$
$\Delta z = z_1^* - z_2^*$
$z_1^* = z_2^*$
$m = 2$

Main Procedure

While $z^* \neq 0$
    While $m \leq M$
        While $p \geq \Delta z$ and $p > 2$
            $P = P \cup P'$
            $H = H \cup H'$
            SOLVE Model-MOU for D with F' and H, fixing hyperbox assignments of samples in $P \rightarrow S, z^*$
            $z_1^* = z^*$
            CALL HTA for solution $S \rightarrow P'$
            $p = |P'|$
        End While
        While $\Delta z \geq p$ or $p \leq 2$
            $F' \cup \{f_m\}$
            SOLVE Model-MOU for D with F' and H, fixing hyperbox assignments of samples in $P \rightarrow S, z^*$
            $z_2^* = z^*$
            $\Delta z = z_1^* - z_2^*$
            $z_1^* = z_2^*$
            $m = m+1$
        End While
    End While
    CALL HCB
End While

Figure 7.8. Pseudo code of HCB-f algorithm

Here, p and Δz account for the absolute improvement of the accuracy of fixed samples and the model accuracy, respectively. As p gets smaller, the power of the generated hyperboxes decreases. As it is observed in the experiments with HCB, hyperboxes with smaller power values are ineffective in the test phase. Thus, for $p \leq 2$, algorithm does not fix the hyperbox but adds a new feature instead. Note that it is also possible to set different treshold values for p, to ensure obtaining more powerful hyperboxes with the algorithm. When the algorithm terminates with 100% accuracy, the main outcome is the hyperboxes to be used as classifiers as in HCB. However, for HCB-f, generated hyperboxes are in different dimensions since set F' changes throughout the iterations. The final F' gives the features selected at the end of the algorithm. Hence, the main idea is to alternate between adding new hyperboxes and adding new features, considering the tradeoff between contributions of these two actions.

7.3. Experimental Results for HCB-f

Experiments are conducted to test the performance of HCB-f in terms of classification and feature selection. *Breast cancer*, *Hepatitis*, *Firm*, *Voting* and *Sonar* datasets from UCI machine learning repository (Dua and Karra, 2017), and microarray datasets *Colon* (Alon et al., 1999), *Leukemia* (Golub et al., 1999), and *Lung Cancer* (Gordon et al., 2002) are used as binary class, multi-feature datasets. *Wine* dataset, which has three classes, is transformed into a two class dataset by merging the 1st and 3rd classes in a single class, and then used in the experiments. Five fold cross validation is performed for all datasets, and feature ranking is conducted separately using the training dataset of each fold. However, for *Colon*, *Leukemia* and *Lung Cancer* datasets, which have extremely large number of features, the initialization step for feature ranking is performed only for the top 1,000 features selected according to the Fisher score. The number of features and the sample sizes of the datasets are given in Table 7.4.

A personal computer with Intel Core i7 2.8 GHz processor and 16.00 GB RAM is used for the runs, and the run time for each solution of Model-MOU is limited with 10 minutes.

Table 7.4. Datasets used in experimenting with the HCB-f algorithm

Dataset	Number of training samples	Number of test samples	Number of features
Breast Cancer (Wisc.)	546	137	9
Wine (2 class)	142	36	13
Hepatitis	124	31	19
Firm	65	18	13
Voting	348	87	16
Sonar	166	42	60
Colon	49	13	2,000
Leukemia	58	14	7,129
Lung Cancer	145	36	12,533

Table 7.5 summarizes the iterations of HCB-f for the first fold of *Breast Cancer* dataset. In the table, A_M is the accuracy of the Model-MOU based on the objective function value z^*, and A_{TB} is the accuracy reached by the samples in the trimmed hyperboxes, after HTA is applied.

Table 7.5. Iteration details of HCB-f for Breast Cancer dataset – fold 1

Iteration	# of Samples to be classified	Set of features used	# of hyperboxes used	z^*	A_M	# of samples fixed, p	A_{TB}
1	546	{2}	2	39	0.9286	42	0.0769
2	504	{2}	3	39	0.9286	51	0.1703
3	453	{2}	4	39	0.9286	22	0.2106
4	431	{2}	5	39	0.9286	0	0.2106
5	431	{2,3}	5	34	0.9377	0	0.2106
6	431	{2,3,6}	7	32	0.9414	87	0.3700
7	344	{2,3,6}	8	32	0.9414	19	0.4048
8	325	{2,3,6}	10	29	0.9469	0	0.4048
9	325	{2,3,6,7}	10	29	0.9469	166	0.7088
10	159	{2,3,6,7}	11	29	0.9469	10	0.7271
11	149	{2,3,6,7}	13	28	0.9487	0	0.7271
12	149	{2,3,6,7,9}	13	28	0.9487	8	0.7418
13	141	{2,3,6,7,9}	15	27	0.9505	6	0.7527
14	135	{2,3,6,7,9}	16	24	0.9560	8	0.7674
15	127	{2,3,6,7,9}	17	21	0.9615	68	0.8919
16	59	{2,3,6,7,9}	18	23	0.9579	6	0.9029
17	53	{2,3,6,7,9}	19	21	0.9615	0	0.9029
18	53	{2,3,6,7,9,5}	19	21	0.9615	9	0.9194
19	44	{2,3,6,7,9,5}	21	15	0.9725	4	0.9267
20	40	{2,3,6,7,9,5}	22	12	0.9780	20	0.9451
21	30	{2,3,6,7,9,5}	24	8	0.9853	10	0.9817
22	10	{2,3,6,7,9,5}	26	0	1.0000	10	1.0000

The algorithm terminates at the 22th iteration, with 26 generated hyperboxes and using the feature set {2,3,6,7,9,5}. When the same dataset was classified using the HCB algorithm (see appendix E, Table E.3), there were 20 generated hyperboxes for all nine features. HCB-f algorithm increases the total number of hyperboxes compared to HCB, however it provides a guidance to eliminate irrelevant features. In addition, the set of features that defines a hyperbox in HCB-f is not the same for all hyperboxes. For example, the first five hyperboxes are only one dimensional and use feature 2. Only the last five hyperboxes make use of all six features that are selected.

Figure 7.9 illustrates the changes in the number of features, number of hyperboxes, and accuracy of fixed samples (A_{TB}) throughout the iterations for the first fold of the *Breast Cancer* dataset. In the figure, the left hand side vertical axis stands for the number of features and number of hyperboxes.

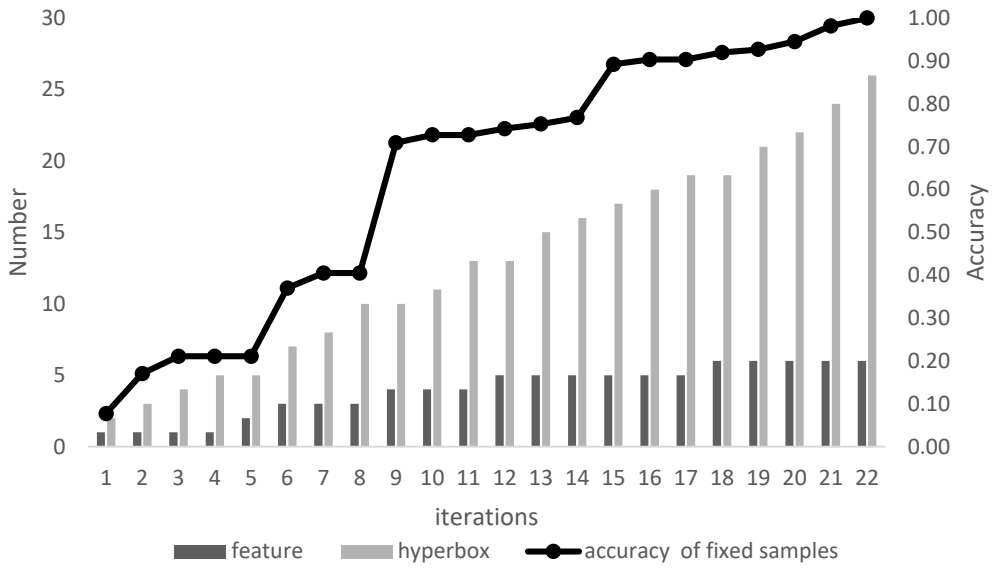


Figure 7.9. Progress of HCB-f for Breast Cancer dataset – fold 1

For the HCB algorithm, power of the generated hyperbox has a decreasing trend throughout the iterations in most of the cases. A similar trend is observed for HCB-f iterations where the number of features remains unchanged. For example, only one feature is used in the first four iterations. During these iterations, increase in the accuracy of fixed samples, which is directly related with the power of trimmed hyperbox, shows a decreasing trend. However, after adding two new features in the

5th and 6th iterations, a jump is observed in the accuracy of fixed samples. The same behavior is observed in the 9th iteration where a fourth feature is added. Although this is an expected situation, the effect of adding a new feature may not always be that strong as the number of samples to be classified is reduced. Also, adding a new hyperbox can still cause generation of hyperboxes with higher powers for some datasets.

Compared to the classification algorithm HCB, HCB-f provides feature selection as well as classification. Moreover, the iterative manner of feature addition provides a control over complexity as a side benefit. Thus, in addition to the number of hyperboxes and number of samples to be classified, the number of features, which is the final source or factor of complexity, becomes controllable. Figure 7.10 illustrates the changes in these three factors of complexity. In the figure, the left hand side axis stands for the number of samples to be classified, and the right hand side axis is used for the numbers of features and hyperboxes. As the numbers of features and hyperboxes increases, the number of samples to be classified decreases, reducing the problem size and providing a balance in terms of complexity. The effect of this balance is also observed in the experiments.

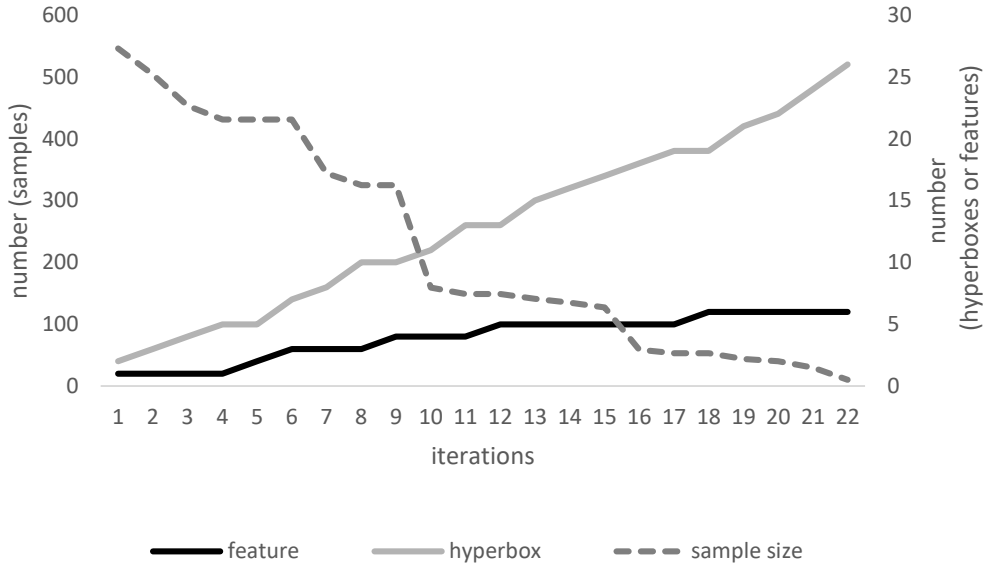


Figure 7.10. Evolution of complexity factors in HCB-f for Breast Cancer dataset - fold 1

As it is reported in Table F.2 of Appendix F for *Breast Cancer* dataset, Model-MOU cannot be solved to optimality in most of the iterations of HCB, and the best result reached in 20 minutes was reported. Using HCB-f, optimal results are obtained for all iterations of *Breast Cancer* dataset in less than 120 seconds in total.

Table 7.6. Cross fold validation results of HCB-f for Breast cancer dataset

	Training Phase		**Test Phase**
Fold	**Feature set**	**# of hyperboxes**	**Accuracy**
1	{2,3,6,7,9,5}	26	0.9630
2	{2,3,6,7,9,5}	26	0.9333
3	{2,3,6,5,7,8}	25	0.9562
4	{2,3,7,6,5,8}	24	0.9489
5	{2,3,6,7,5,8}	24	0.9635

When the five fold cross validation results of *Breast Cancer* dataset are analyzed, it is seen that six features are selected in all folds as shown in Table 7.6. While features 2,3,5,6, and 7 are selected in common, features 1 and 4 are not selected in any of the folds. Moreover, the features in the set are given in their importance order, and the two most important features are the same for all folds. The number of generated hyperboxes also varies in a narrow range between 24 and 26. Since the training accuracy is 100% for all folds, cross validation test accuracies of a classifier on test data of other folds are also 100%. Thus, using the hyperboxes generated in the 5^{th} fold yields the best results in terms of test accuracies. Note that, this accuracy is better than the best test accuracy reported with HCB.

Table 7.7 summarizes the results of numerical experiments for the datasets with relatively smaller number of features. Features that are selected are listed in their importance order. When the results are examined, it is seen that the feature selections are fairly consistent among the folds. The detailed results for the folds having the best test accuracies are to be discussed further in this section. Details on iteration progress for the remaining folds can be seen in Appendix H.

Table 7.7. HCB-f cross validation results for Wine, Hepatitis, Firm, Voting, and Sonar datasets

		Training Phase			Test Phase
Dataset	**Fold**	**F**	**Feature set**	**L**	**Accuracy**
wine (2 class)	1	5	{10,1,2,3,5}	10	0.9722
	2	4	{10,1,2,3}	10	0.8889
	3	4	{10,1,2,3}	8	0.9167
	4	4	{10,1,2,3}	10	0.9722
	5	4	{10,1,3,4}	8	0.8889
Hepatitis	1	6	{17,12,13,14,18,15}	14	0.9032
	2	4	{17,18,14,12}	14	0.8387
	3	5	{17,18,14,12,15}	13	0.8710
	4	6	{17,12,14,18,8,9}	14	0.9032
	5	5	{17,12,18,14,13}	14	0.8065
Firm	1	2	{13,7}	6	0.8889
	2	3	{13,6,9}	5	0.9375
	3	1	{13}	2	0.8824
	4	3	{13,7,6}	7	0.8750
	5	4	{13,7,6,9}	5	0.9412
Voting	1	11	{4,3,12,5,8,9,7,13,14,15,16}	13	0.9080
	2	12	{4,3,12,5,8,9,13,14,7,15,16,1}	12	0.9540
	3	11	{4,3,12,5,8,9,14,13,7,16,15}	15	0.9655
	4	11	{4,3,5,12,8,9,14,7,13,15,16}	16	0.9770
	5	12	{4,12,3,5,8,9,14,13,7,15,16,11}	16	0.9425
Sonar	1	9	{11,9,12,48,10,13,49,51,52}	31	0.8810
	2	9	{11,12,13,10,48,49,9,52,46}	24	0.7805
	3	8	{12,11,13,10,21,49,48,9}	22	0.6829
	4	9	{9,11,51,12,49,48,10,36,52}	22	0.7619
	5	10	{11,9,12,10,14,21,13,49,48,20}	23	0.6905

L: Number of hyperboxes, **F**: Number of features

For *Wine* (*2 class*) dataset, features 1, 2, 3, and 10 are commonly selected in all folds. When the method described in section 7.1 was used, features 1, 3, and 10 were also found as the important features for *Wine* dataset. Although the *Wine* dataset was analyzed in its original form consisting of three classes in that experiment, it can be said that the results are still consistent for those three features. According to Table 7.7 the best test accuracies are obtained in the 1st and the 4th folds, each with 10 hyperboxes. However, using the hyperboxes generated in the 4th fold would be better in terms of interpretability, since these hyperboxes are defined with fewer features, and they provide simpler classifier rules. Progress of iterations for the 4th fold of *Wine* dataset is illustrated in Figure 7.11 and Table 7.8.

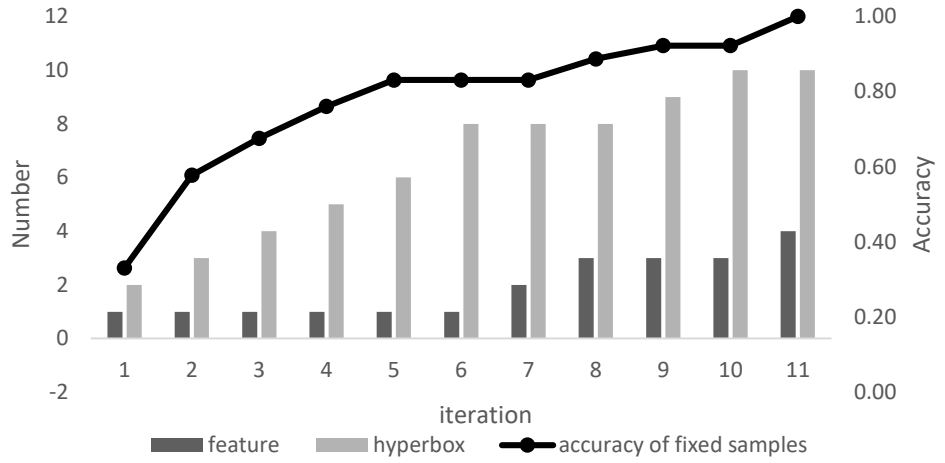


Figure 7.11. Progress of HCB-f for Wine (2 class) dataset - fold 4

In Table 7.8, the first six hyperboxes are defined by one feature, and only the last hyperbox is defined in terms of four features. In addition, it is possible to reach 90% training accuracy only by using feature 10.

Table 7.8. Iteration details of HCB-f for Wine (2 class dataset) - fold 4

Iteration	# of samples to be classified	Set of features used	# of hyperboxes used	z*	A_M	# of samples fixed, p	A_{TB}
1	141	{10}	2	15	0.8944	47	0.3310
2	95	{10}	3	15	0.8944	35	0.5775
3	60	{10}	4	15	0.8944	14	0.6761
4	46	{10}	5	15	0.8944	12	0.7606
5	34	{10}	6	14	0.9014	10	0.8310
6	24	{10}	8	13	0.9085	0	0.8310
7	24	{10,1}	8	8	0.9437	0	0.8310
8	24	{10,1,2}	8	6	0.9577	8	0.8873
9	16	{10,1,2}	9	2	0.9859	5	0.9225
10	11	{10,1,2}	10	2	0.9859	0	0.9225
11	11	{10,1,2,3}	10	0	1.0000	11	1.0000

A_M: Model-MOU accuracy, **A_{TB}**: accuracy of fixed samples

The number of generated hyperboxes are almost the same for all folds of *Hepatitis* dataset as seen in Table 7.7, and the number of selected features is between four and six. The best test accuracy of 90% is obtained in the 1st and 4th folds. In both folds, 14 hyperboxes are generated using at most six features. Details for the 4th fold are given in Figure 7.12 and Table 7.9.

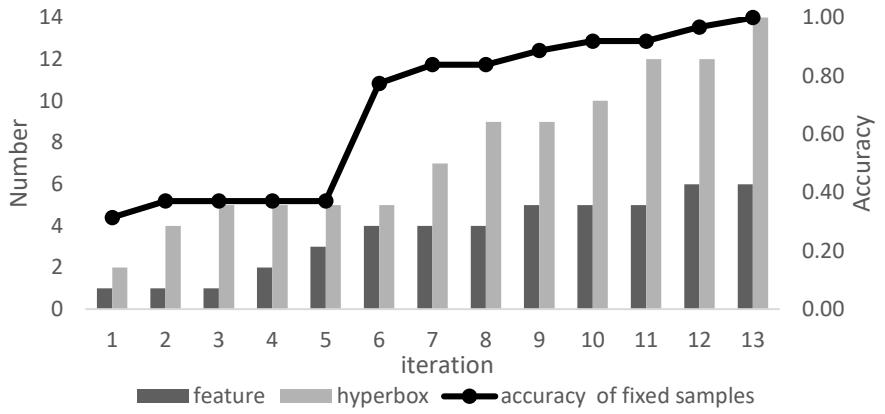


Figure 7.12. Progress of HCB-f for Hepatitis dataset - fold 4

After the 3rd iteration in Figure 7.12, three features are included in the model one after another. This led to a major jump in accuracy in iteration 6. Eventually, last 10 hyperboxes are defined by 4 or more features as seen in Table 7.9.

Table 7.9. Iteration details of HCB-f for Hepatitis dataset – fold 4

Iteration	# of samples to be classified	Set of features used	# of hyperboxes used	z^*	A_M	# of samples fixed, p	A_{TB}
1	124	{17}	2	14	0.8871	39	0.315
2	85	{17}	4	14	0.8871	7	0.371
3	78	{17}	5	14	0.8871	0	0.371
4	78	{17,12}	5	12	0.9032	0	0.371
5	78	{17,12,14}	5	11	0.9113	0	0.371
6	78	{17,12,14,18}	5	8	0.9355	50	0.774
7	28	{17,12,14,18}	7	7	0.9435	8	0.839
8	20	{17,12,14,18}	9	6	0.9516	0	0.839
9	20	{17,12,14,18,8}	9	6	0.9516	6	0.887
10	14	{17,12,14,18,8}	10	6	0.9516	4	0.919
11	10	{17,12,14,18,8}	12	2	0.9839	0	0.919
12	10	{17,12,14,18,8,9}	12	1	0.9919	6	0.968
13	4	{17,12,14,18,8,9}	14	0	1.0000	4	1.000

A_M: Model-MOU accuracy, **A_{TB}**: accuracy of fixed samples

The HCB-f algorithm terminates in at most seven iterations for the *Firm* dataset. For the 3rd fold, 100% train accuracy is obtained at the startup iteration, using just one feature and two hyperboxes. However, the corresponding test accuracy for this fold is below 90%. The best test accuracy is obtained using five hyperboxes generated in the 5th fold and four features selected. The main reason of variability in number of hyperboxes and feature selection is the limited size of the training dataset. The details of iterations for the 5th fold are summarized in Figure 7.13 and Table 7.10.

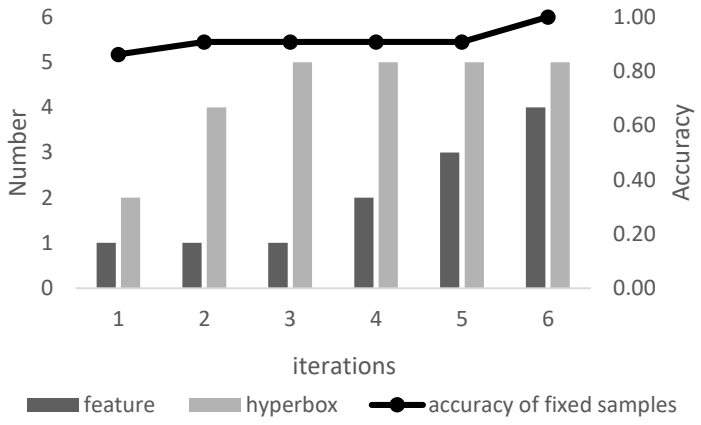


Figure 7.13. Progress of HCB-f for Firm dataset - fold 5

Table 7.10. Iteration details of HCB-f for Firm dataset - fold 5

Iteration	# of samples to be classified	Set of features used	# of hyperboxes used	z^*	A_M	# of samples fixed, p	A_{TB}
1	65	{13}	2	3	0.9538	56	0.8615
2	9	{13}	4	2	0.9692	3	0.9077
3	6	{13}	5	2	0.9692	0	0.9077
4	6	{13,7}	5	2	0.9692	0	0.9077
5	6	{13,7,6}	5	1	0.9846	0	0.9077
6	6	{13,7,6,9}	5	0	1.0000	6	1.0000

A_M: Model-MOU accuracy, **A_{TB}**: accuracy of fixed samples

For the 5th fold, 94% test accuracy is obtained using five hyperboxes. First three of these hyperboxes are defined with only one feature, and 59 out of 65 samples are correctly classified by these boxes in the training phase. The last two hyperboxes are defined with four features and they classify just six samples. However, these last two hyperboxes have significant classifying power in the test phase.

Voting is a categorical dataset of questionnaire results. Each feature corresponds to answering a yes/no question and the answers are coded as 1/0. Due to the binary valued features, required rule variation to classify the samples can be obtained using 11-12 features out of 16 as seen in Table 7.7. The best test accuracy is obtained as 97.7% in the 4th fold with 16 hyperboxes. The details for the iterations of this fold are given in Figure 7.14.

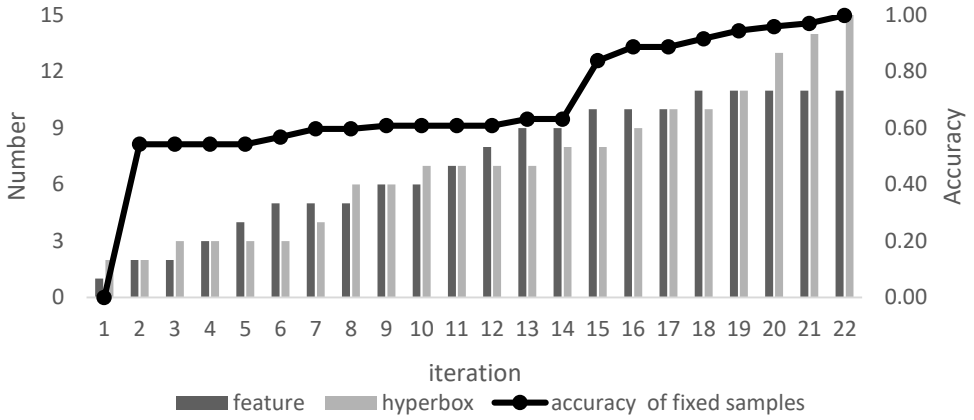


Figure 7.14. Progress of HCB-f for Voting dataset - fold 4

As seen in the figure, HCB-f adds the second feature immediately, since the classes are not separable with just one feature. More than half of the samples can be classified using the two-feature hyperboxes generated in iteration 2. Then, eight more features are included in the model by iteration 15 where a jump in accuracy is observed. After the 15th iteration, the classes are better separable and HCB-f mostly generates additional hyperboxes to classify the samples. Details are given in Table 7.11.

Table 7.11. Iteration details of HCB-f for Voting dataset - fold 4

Iter.	# of samples to be classified	Set of features used	L	z^*	A_M	p	A_{TB}
1	348	{4}	2	14	0.9598	0	0.0000
2	348	{4,3}	2	14	0.9598	189	0.5431
3	159	{4,3}	3	14	0.9598	0	0.5431
4	159	{4,3,5}	3	14	0.9598	0	0.5431
5	159	{4,3,5,12}	3	14	0.9598	0	0.5431
6	159	{4,3,5,12,8}	3	14	0.9598	9	0.5690
7	150	{4,3,5,12,8}	4	14	0.9598	10	0.5977
8	140	{4,3,5,12,8}	6	14	0.9598	0	0.5977
9	140	{4,3,5,12,8,9}	6	14	0.9598	4	0.6092
10	136	{4,3,5,12,8,9}	7	14	0.9598	0	0.6092
11	136	{4,3,5,12,8,9,14}	7	13	0.9626	0	0.6092
12	136	{4,3,5,12,8,9,14,7}	7	12	0.9655	0	0.6092
13	136	{4,3,5,12,8,9,14,7,13}	7	11	0.9684	8	0.6322
14	128	{4,3,5,12,8,9,14,7,13}	8	14	0.9598	0	0.6322
15	128	{4,3,5,12,8,9,14,7,13,15}	8	14	0.9598	72	0.8391
16	56	{4,3,5,12,8,9,14,7,13,15}	9	24	0.9310	17	0.8879
17	39	{4,3,5,12,8,9,14,7,13,15}	10	22	0.9368	0	0.8879
18	39	{4,3,5,12,8,9,14,7,13,15,16}	10	16	0.9540	10	0.9167
19	29	{4,3,5,12,8,9,14,7,13,15,16}	11	14	0.9598	10	0.9454
20	19	{4,3,5,12,8,9,14,7,13,15,16}	13	10	0.9713	5	0.9598
21	14	{4,3,5,12,8,9,14,7,13,15,16}	14	3	0.9914	4	0.9713
22	10	{4,3,5,12,8,9,14,7,13,15,16}	16	0	1.0000	10	1.0000

L: # of hyperboxes used, A_M: Model-MOU accuracy, **p:** # of samples fixed, A_{TB}: accuracy of fixed samples

The worst test results in the experiments are obtained with the *Sonar* dataset. In this dataset, there are 60 decimal features valued in the interval [0,1]. None of the features are capable of distinguishing the classes since the feature values for the two classes are completely interwoven. Thus HCB-f starts with a low accuracy in all folds. In addition, the single feature performances are close to each other, and the feature ranking is not as informative as it is for the other datasets. For example, the 11th feature is ranked first or second in all folds, and values of this feature (for all samples in training and test data) with respect to the classes are plotted in Figure 7.15. As seen in the figure, sample values are very close to each other for the two classes. The cut points that define hyperbox boundaries are data dependent in the training phase. Naturally, this dependency causes instable test phase accuracies.

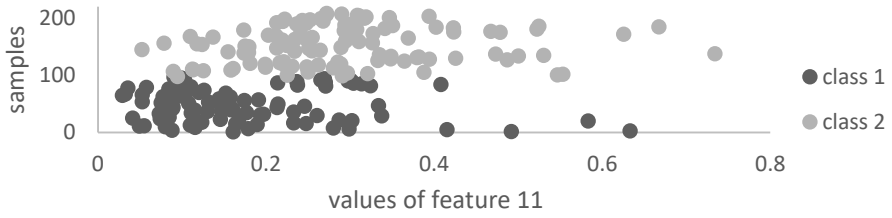


Figure 7.15. Values of feature 11 for Sonar dataset

For the five folds of *Sonar* dataset, 24.2 hyperboxes generated, and nine features are selected on the average. The test phase accuracies are in a wide range between 68% and 88%. The iterations of the first fold, which is the best one in terms of test accuracy, are summarized in Figure 7.15 and Table 7.12.

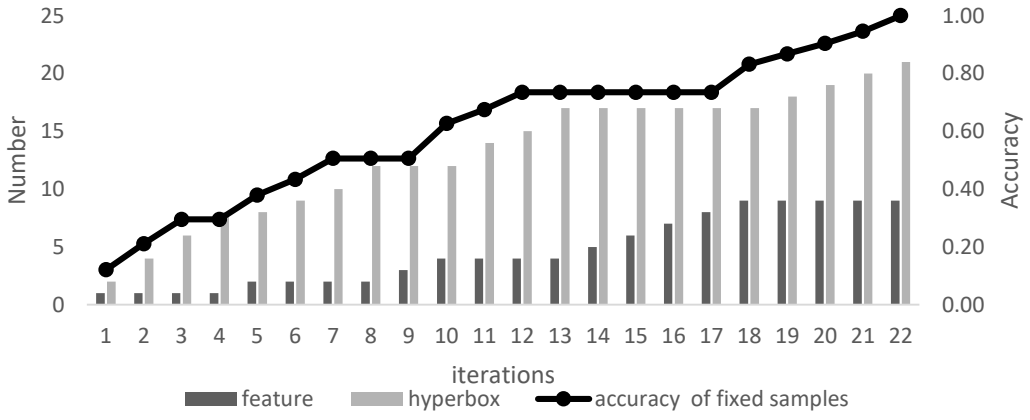


Figure 7.16. Progress of HCB-f for Sonar dataset - fold 1

Table 7.12. Iteration details of HCB-f for Sonar dataset - fold 1

Iter.	# of samples to be classified	Set of features used	L	z^*	A_M	p	A_{TB}
1	166	{11}	2	42	0.7470	20	0.1205
2	146	{11}	4	42	0.7470	15	0.2108
3	131	{11}	6	42	0.7470	14	0.2952
4	117	{11}	8	42	0.7470	0	0.2952
5	117	{11,9}	8	41	0.7530	14	0.3795
6	103	{11,9}	9	39	0.7651	9	0.4337
7	94	{11,9}	10	39	0.7651	12	0.5060
8	82	{11,9}	12	39	0.7651	0	0.5060
9	82	{11,9,12}	12	37	0.7771	0	0.5060
10	82	{11,9,12,48}	12	36	0.7831	20	0.6265
11	62	{11,9,12,48}	14	35	0.7892	8	0.6747
12	54	{11,9,12,48}	15	30	0.8193	10	0.7349
13	44	{11,9,12,48}	17	27	0.8373	0	0.7349
14	44	{11,9,12,48,10}	17	26	0.8434	0	0.7349
15	44	{11,9,12,48,10,13}	17	25	0.8494	0	0.7349
16	44	{11,9,12,48,10,13,49}	17	23	0.8614	0	0.7349
17	44	{11,9,12,48,10,13,49,51}	17	20	0.8795	0	0.7349
18	44	{11,9,12,48,10,13,49,51,52}	17	18	0.8916	16	0.8313
19	28	{11,9,12,48,10,13,49,51,52}	18	13	0.9217	6	0.8675
20	22	{11,9,12,48,10,13,49,51,52}	19	7	0.9578	6	0.9036
21	16	{11,9,12,48,10,13,49,51,52}	20	6	0.9639	7	0.9458
22	9	{11,9,12,48,10,13,49,51,52}	21	0	1.0000	9	1.0000

L: # of hyperboxes used, A_M: Model-MOU accuracy, **p:** # of samples fixed, A_{TB}: accuracy of fixed samples

Applying feature selection techniques becomes a real prerequisite in many studies on classification problems of bioinformatics area, especially for high-dimensional microarray datasets (Saeys et al., 2007). HCB-f is tested on microarray datasets *Colon*, *Leukemia*, and *Lung Cancer*, which have thousands of features and no more than 150 samples. Although the run time to solve the Model-MOU with a single feature is less than 60 seconds, it takes cumulatively a long time to solve the model for all features. Thus, as mentioned before, Fisher scores of the features are computed and the top 1,000 features are considered in the initialization procedure of HCB-f. The results of HCB-f for these datasets are summarized in Table 7.13. Iteration details for the folds with highest test phase accuracy is given in this section (see Appendix I for the remaining folds). As seen in the table, at most five features are selected for *Colon* dataset and four of them are observed in all folds, with the same ranks. *Leukemia* and *Lung Cancer* datasets are originally published with separated training and test data partitions. The results obtained with these partitions is given in the "original" line.

Table 7.13. HCB-f cross fold validation results for Colon, Leukemia, and Lung cancer datasets

		Training Phase			Test Phase
Dataset	**Fold**	**F**	**Feature set**	**L**	**Accuracy**
Colon	1	4	{1671,492,1771,249}	7	0.8333
	2	4	{1671,492,1771,249}	8	0.8333
	3	5	{1671,492,1771,249,512}	9	0.9167
	4	4	{1671,492,1771,249}	10	0.8333
	5	5	{1671,492,1771,249,512}	9	0.9167
Leukemia	1	3	{1882,1834,2288}	4	0.9286
	2	1	{3252}	4	0.8667
	3	2	{1834,1882}	4	1.0000
	4	2	{1834,1882}	5	0.9333
	5	1	{4847}	2	0.7857
Leukemia	original	2	{2020,760}	4	0.7941
Lung Cancer	1	1	{8005}	4	0.9444
	2	2	{7200,8005}	6	0.9444
	3	1	{3333}	4	0.9167
	4	2	{7765,7200}	5	0.9722
	5	2	{3334,3844}	2	0.9459
Lung Cancer	original	1	{1136}	2	0.9060

As another experiment, all training and test samples in these datasets are randomly divided into five equal parts, and five fold cross validation is performed for these parts. The features selected in this experiment are different from the ones that are obtained using the original training data. For *Leukemia* dataset, at most three features are selected in the first fold and this yields a test phase accuracy of approximately 93%. Interestingly, perfect accuracy is achieved in the 3rd fold of *Leukemia* with only two features and four hyperboxes. For *Lung Cancer* dataset, the best test phase accuracy is about 97% in the 4th fold using five hyperboxes and two features. For *Leukemia* and *Lung Cancer* datasets, selected features are not the same across the folds. However, for both datasets, features selected in any of the folds are ranked by Model-MOU among the top 10 for all folds. The training datasets are solved even without requiring all those top 10 features.

Highest test phase accuracy is reported in the 3rd and the 5th folds of the *Colon* dataset, with the same feature space. Details for the iterations of the 3rd fold is given in Figure 7.17 and Table 7.14.

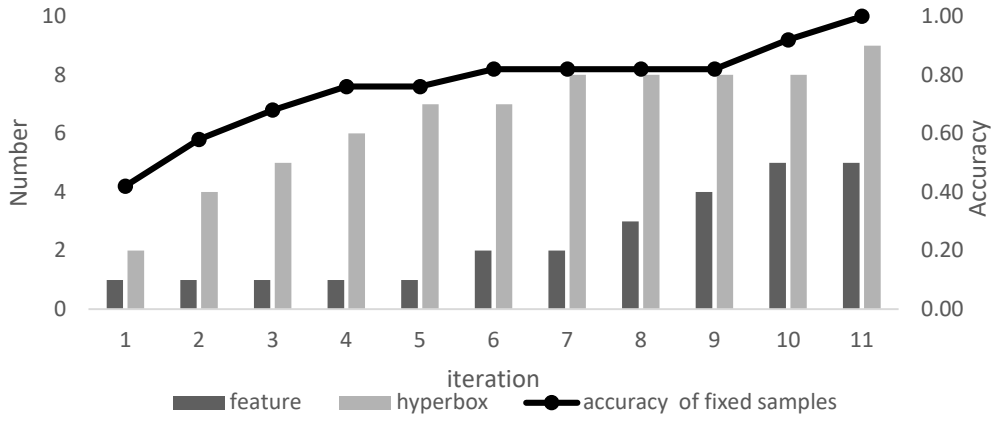


Figure 7.17. Progress of HCB-f for Colon dataset - fold 3

As seen in the table, 94% of the samples are classified with eight hyperboxes defined by only two features in 7th iteration of HCB-f. In order to reach the 100% training accuracy, HCB-f adds three more features. These three features are used for only 8th and 9th hyperboxes.

Table 7.14. Iteration details of HCB-f for Colon dataset – fold 3

Iter.	# of samples to be classified	Set of features used	L	*z**	A_M	*p*	A_{TB}
1	50	{1671}	2	7	0.8600	21	0.4200
2	29	{1671}	4	7	0.8600	8	0.5800
3	21	{1671}	5	7	0.8600	5	0.6800
4	16	{1671}	6	6	0.8800	4	0.7600
5	12	{1671}	7	6	0.8800	0	0.7600
6	12	{1671,492}	7	5	0.9000	3	0.8200
7	9	{1671,492}	8	3	0.9400	0	0.8200
8	9	{1671,492,1771}	8	3	0.9400	0	0.8200
9	9	{1671,492,1771,249}	8	2	0.9600	0	0.8200
10	9	{1671,492,1771,249,512}	8	1	0.9800	5	0.9200
11	4	{1671,492,1771,249,512}	9	0	1.0000	4	1.0000

L: # of hyperboxes used, A_M: Model-MOU accuracy, **p:** # of samples fixed, A_{TB}: accuracy of fixed samples

Using the original training samples of *Leukemia* dataset, HCB-f ends up with two features and four hyperboxes in only three iterations as seen in Table 7.15.

Table 7.15. Iteration details of HCB-f for the original training dataset of Leukemia

Iter.	# of samples to be classified	Set of features used	# of hyperboxes used	z^*	A_M	# of samples fixed, p	A_{TB}
1	38	{2020}	2	1	0.97	26	0.68
2	12	{2020}	3	1	0.97	7	0.87
3	5	{2020,760}	4	0	1.00	5	1.00

Hyperboxes generate interpretable results as mentioned several times before. Selected features 2020 and 760 for *Leukemia* correspond to the genes FAH fumarylacetoacetate and epidermal growth factor receptor kinase substrate (Eps8) mRNA, respectively. As an example, the resulting hyperbox boundaries are reported in Table 7.16.

Table 7.16. Hyperbox boundaries for the original training dataset of Leukemia

Hyperbox	Class	Feature	Lower bound	Upper bound
1	ALL	2020	0	1286
2	AML	2020	1286.1	1821
3	ALL	2020	1811	1822
		760	-10	71
4	AML	2020	1822.1	2633
		760	71.1	9600

When five folds are used for the *Leukemia* dataset, perfect test accuracy is obtained in the 3rd fold as 100% using the hyperboxes given in Table 7.17. The reason of obtaining higher accuracies in this experiment is the larger training dataset size. Originally, training dataset and test dataset have the same size. However, 80% of these data are used for training in the five fold experiment. As the training sample size gets larger, the selected features and generated hyperboxes become more accurate.

Table 7.17. Hyperbox boundaries for Leukemia dataset – fold 3

Hyperbox	Class	Feature	Lower bound	Upper bound
1	ALL	1834	0	274
2	AML	1834	395	2204
3	ALL	1834	275	395
		1882	-200	394
4	AML	1834	261	450
		1882	395	14555

1834: CD33 antigen (differentiation antigen)

1882: CST3 Cystatin C (amyloid angiopathy and cerebral hemorrhage)

Finally, HCB-f is applied to *Lung Cancer* dataset. Among the folds, at most two features are selected, and the test phase accuracy is higher than 90% even with a single feature. The best test accuracy is above 97%, and obtained in the 4th fold using two features, namely 37716_at (7765) and 37157_at (7200) and five hyperboxes as seen in Table 7.18.

Table 7.18. Hyperbox boundaries for the original training dataset of Lung Cancer

Hyperbox	Class	Feature	Lower bound	Upper bound
1	1	7765	635.91	1595.1
2	2	7765	0	511.09
3	1	7765	511.71	576.1
		7200	683.81	1328.9
4	2	7765	512.01	576.09
5	2	7765	512.01	635.9
		7200	0	683.08

As given in Table 7.13, when the original training dataset of 32 samples is used, 90% test accuracy is achieved on the remaining 149 samples. HCB-f selects only one feature and generates two hyperboxes for this setting. The resulting boundaries of the hyperboxes are given in Table 7.19.

Table 7.19. Hyperbox boundaries for Lung Cancer dataset - original

Hyperbox	Class	Feature	Lower bound	Upper bound
1	1	1136	0	1012
2	2	1136	1012.1	1862.8

1136: 2047_s_at

7.4. Comparison of HCB-f Performance with Recent Studies in the Literature

There are many feature selection approaches proposed in the literature and tested on the datasets reported above. However, it is not possible to make a completely fair comparison between the proposed feature selection methods, since they yield different results under different classifier models. In addition, the validation method (folds, generation of training and test datasets) is also effective on the performance.

In this section, performance of HCB-f is compared with selected recent studies, which use common datasets as the ones in Table 7.4, and results in the best test accuracies to the best of our knowledge.

In the study of Shunmugapriya and Kanmani (2017) the swarm based AC-ABC hybrid algorithm combines the characteristics of ant and artificial bee colony optimization for feature selection, where ants use exploitation by the bees to determine the best feature subset, and bees adapt the ants as their food sources. Performance comparison of this method with some other meta-heuristics feature selection methods are available in their study.

Mafarja et al. (2018) have developed the BGOA_EPD meta-heuristic, which combines the Grasshopper optimization algorithm (GOA) with evolutionary population dynamics to design a wrapper-based feature selection method. GOA is a recent population based metaheuristic that mimics the swarming behavior of grasshoppers. In their study, *k*-nearest neighbor (k = 5) with Euclidean metric is used as the classification method. BGOA_EPD_tour is the best performing variant of BGOA_EPD algorithm which utilizes tournament selection among guiding solutions. The performance comparison given in the study indicates that BGOA_EPD_tour outperforms metaheuristics such genetic algorithm and particle swarm optimization, and filter-based methods such as correlation-based feature selection, fast correlation based filter, Fisher score, and information gain.

Table 7.20. Comparison of HCB-f with meta-heuristics

	HCB-f				**AC-ABC hybrid[1]**		**BGOA_EPD_Tour[2]**	
Dataset	**SF_{Best}**	**A_{Best}**	**SF_{avg}**	**A_{avg}**	**SF_{Best}**	**A_{Best}**	**SF_{avg}**	**A_{avg}**
Breast Cancer	6	96.35	6	95.30	3	99.43	5	98.00
Hepatitis	6	90.32	5.2	86.45	11	81.29	-	-
Voting	11	97.70	11.4	94.94	-	-	5.43	96.60
Sonar	9	88.10	9	75.94	13	96.74	36.77	91.12
Colon	5	91.67	4.4	86.67	-	-	1064	87.00
Leukemia	2	100.0	1.8	90.29	-	-	3769	93.10
Lung Cancer	2	97.22	1.6	94.47	27	90.83	-	-

[1] Hybrid algorithm using ant colony and bee colony (Shunmugapriya and Kanmani, 2017)

[2] Binary Grasshopper optimization with evolutionary population dynamics. (Mafarja et al., 2018)

Table 7.20 summarizes the results of HCB-f matheuristic, AC-ABC hybrid (Shunmugapriya and Kanmani, 2017), and BGOA_EPD_ Tour (Mafarja et al., 2018) meta-heuristics. A_{best} stands for to the best test accuracy achieved by the method, and SF_{Best} is the number of selected features that gives the best test accuracy. A_{avg} is the average test accuracy, and SF_{avg} is the average number of features selected. Results of five folds are averaged for HCB-f, whereas 10 and 30 independent runs are used for AC-ABC hybrid and BGOA_EPD_Tour, respectively.

According to results given in Table 7.20 AC-ABC hybrid outperforms HCB-f for *Breast Cancer* and *Sonar* datasets, and HCB-f performs better than AC-ABC hybrid for *Hepatitis* and *Lung Cancer* datasets, in terms of test phase accuracy. On the other hand, except for the *Breast Cancer* dataset, numbers of selected features are fewer with HCB-f. As seen in the results, BGOA_EPD_tour outperforms HCB-f in terms of average test accuracies. However, for *Voting*, *Colon*, and *Leukemia* datasets best test accuracies reported by HCB-f are better. In addition, extremely fewer features are selected for *Colon* and *Leukemia* datasets using HCB-f.

Although identities of the selected features are important information for datasets, many of the studies only report the number of selected features without giving detailed information on them. However, it is possible to obtain selected feature domains especially in the field of bioinformatics.

Table 7.21. Features selected with HCB-F for Leukemia and Lung Cancer datasets

Leukemia		**Lung Cancer**	
Feature	**Occurrence**	**Feature**	**Occurrence**
1882 (Cystatin C)	**3**	**7200 (37157_at)**	**2**
1834 (CD33)	**3**	**8005 (37954_at)**	**2**
2288 (adipsin)	**1**	3333 (33327_at)	1
3252 (Glutathione)	1	3334 (33328_at)	1
4847 (Zyxin)	**1**	1136 (2047_s_at)	1
2020 (Fumarylacetoacetate)	**1**	7765 (37716_at)	1
760 (Cystatin A)	1	3844 (33833_at)	1

Lists of selected features with HCB-f are given in Table 7.21 for Leukemia and Lung Cancer datasets. The features represented in bold were also selected in the original articles that presented the datasets (Golub et al., 1999) and (Gordon et al., 2002).

The performance of HCB was compared to classification trees using CART in Section 6.3. In order to make a similar comparison, experimental results of classification tree used together with the filter-based feature selection method Fisher score (F-score) are taken from the study of Li et al. (2017).

Table 7.22. HCB-f compared to classification tree and F-score (Li et al., 2017)

	HCB-f		**Decision tree classifier with F-score**			
Dataset	**SF_{Best}**	**A_{Best}**	**SF_{min}**	**A**	**SF_{Best}**	**A_{Best}**
Colon	5	91.67	5	74.05	115	79.29
Leukemia	2	100.0	5	90.35	260	97.50
Lung Cancer	2	97.22	5	68.90	155	92.55

According to Table 7.22, HCB-f outperforms classification tree with F-score in terms of best test accuracies. The minimal set of features selected by F-score consists of five features, and their corresponding test accuracies in decision tree algorithm (given in column A) are below than the accuracies reached by HCB-f with only fice or two features. On the other hand, the best accuracies of the classification tree algorithm are obtained with over 100 features and still lower than those obtained by HCB-f.

CHAPTER 8

CONCLUSION

In this thesis, three MIP optimization models, Model-MOB, Model-MO, and Model-MOU are proposed to classify datasets with numerical features by the use of hyperboxes. Using the hyperboxes as classifiers, the results are highly interpretable as the boundaries of each class are determined in terms of feature values. Model-MOB minimizes the weighted sum of misclassified samples, overlap samples, and the number of hyperboxes. In order to improve performance of Model-MOB, hyperbox-to-class assignments are given as input to Model-MO and the term for the number of hyperboxes is removed from the objective function. As a contribution to the existing literature, hyperboxes are allowed to overlap to avoid overfitting. Furthermore, in Model-MOU, samples are not forced to be assigned to a hyperbox, but the numbers of overlap and unassigned samples are minimized in the objective function along with the number of misclassified samples. With this approach, Model-MOU can find feasible solutions in reasonable times even for the datasets with highly overlapping classes. In addition, allowing the overlap prevents overfitting of the dataset, which is an undesirable situation in the training phase.

The resulting hyperboxes of the MIP models are used to classify new samples in the test phase. However, it is not possible to classify the overlap and uncovered samples using these hyperboxes. For the classification of these samples, a distance-based heuristic, COUS, is developed. It is seen in the experiments that accuracy of correctly classifying the uncovered samples are above 90% for most datasets. The accuracy of classifying overlap samples is lower, which is around 50%. This is natural because these samples fall into the overlapping regions of multiple classes. However, the results still provide some information about the classes of the overlap samples since their possible classes are limited by approximately two classes. This indicates the indifference of overlap samples between those two classes.

Although Model-MOU is a more efficient model compared to Model-MOB and Model-MO, it is still intractable for datasets with large number of samples and features. In order to overcome this complexity problem a matheuristic for hyperbox classifying of binary classes is developed in this study. The initial prototype of this matheuristic, namely ICB, uses Model-MO as the base model. In each iteration, Model-MO is solved, pure hyperboxes with no misclassified samples are extracted with the hyperbox trimming algorithm (HTA), and respective sample-to-hyperbox assignments are fixed. Then, an additional hyperbox is allocated to the class that does not have a free hyperbox before the following iteration. The ICB algorithm is tested for four simulated datasets and results are found to be promising. However, it is seen that ICB also becomes intractable for multi-feature and larger sized datasets. Analyzing the causes of this intractability, Hyperbox Classification for Binary dataset (HCB) matheuristic is finalized with some modifications on ICB. HCB uses different strategies for fixing sample-to-hyperbox assignments in HTA and additional hyperbox allocation. Most importantly, it uses Model-MOU, which relaxes sample-to-hyperbox assignmets, for the generation of hyperboxes in each iteration. According to the experimental results, HCB outperforms ICB in terms of run time, iteration count, and final accuracy.

When the performances of HCB and CART are compared, it is observed that, for most of the cases, HCB provides better test scores using fewer classifier rules. This is a success indicator in terms of optimality of the number of hyperboxes as well as the accuracy. Unfortunately, HCB does not have any control on the feature set, and it is not efficient enough to solve the problems in reasonable times as the number of features increases.

The MIP models also support feature selection. Relevance or effectiveness of the features to resolve the overlap situation is not all the same. For some datasets, overlapping of classes can be resolved without the use of all features in the classifier. When the solutions of the problem instances are analyzed, it is observed that the decision variables related with the overlap samples are indicators of the irrelevant or redundant features. An algorithm is developed based on this property of the model. It is seen that a completely redundant feature can easily be detected by this algorithm.

Also, it is possible to rank the features according to their levels of effectiveness. However, due to the complexity problem, this ranking is not possible for larger sized problems.

Finally, HCB matheuristic algorithm is extended into another matheuristic algorithm, HCB-f, to facilitate feature selection. This algorithm first ranks the features using Model-MOU. It then adds features into the model iteratively based on this feature ranking. Basically, the HCB-f algorithm starts as HCB where the dataset is defined by only one feature. The power of generated hyperboxes decreases throughout the iterations, and whenever it becomes impossible to trim a hyperbox with no more than two samples in it, a new feature is included in the model. The conducted experiments show that the HCB-f algorithm terminates with 100% training accuracy without the use of all features. HCB-f is not only beneficial for the feature selection, but also reduces the complexity of the problems utilizing fewer features.

In reported experiments, HCB-f focuses on use of a minimal feature set by allowing to generate hyperboxes even with only three samples. As the treshold to trim a hyperbox is increased, the algorithm would tend to add new features instead of adding a new hyperbox. Some further experiments may be performed to see the interaction between the number of features and the number of hyperboxes.

The proposed MIP formulations of Model-MOB, Model-MO, and Model-MOU are all capable of solving multi-class datasets. However, HCB and HCB-f are designed for binary class datasets in this study. These algorithms can be modified to solve multi-class datasets as a future extension of this study.

As other future research issues:

- Decision variables related with overlap samples can be further explored in all three MIP models as indicators of future effectiveness, ranking, or selection. Specifically, since Model-MOU does not enforce sample-to-hyperbox assignments, overlap decision variables in this model can be even more informative.

- Some of the datasets used in the experiments have missing values for some samples. The MIP models and the heuristics seem to handle these missing values without any problems. However, the effect of missing values on the proposed classification approach needs to be studied further.
- Possible ways of speeding up the heuristics, specifically reducing the solution time of Model-MOU, should be searched for. This would allow using the proposed approach for much larger datasets.

REFERENCES

Aggarwal, C. C., 2014. Data classification: Algorithms and applications. New York, NY: Chapman and Hall.

Alon, U., Barkai, N., Notterman, D. A., Gish, K., Ybarra, S., Mack, D., and Levine, A. J., 1999. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. Proceedings of the National Academy of Sciences of the United States of America, *96*(12), 6745-50.

Astorino, A., & Gaudioso, M., 2002. Polyhedral separability through successive LP. Journal of Optimization theory and applications, 112(2), 265-293.

Bagirov, A. M., 2005. Max–min separability. Optimization Methods and Software, 20(2-3), 277-296.

Bagirov, A. M., Ugon, J., & Webb, D., 2011. An efficient algorithm for the incremental construction of a piecewise linear classifier. Information Systems, 36(4), 782-790.

Bertolazzi, P., Felic, G., Festa, P., 2010. Mathematical models for feature selection and their application in bioinformatics, Nettab BBCC November 2010 Neaples.

Boser, B. E., Guyon, I. M., & Vapnik, V. N., 1992. A training algorithm for optimal margin classifiers. In Proceedings of the fifth annual workshop on Computational learning theory (pp. 144-152). ACM.

Brown, G. W., 1947. Discriminant functions, Ann. Math. Statist., 18: 514- 528.

Burges, C. J., 1998. A tutorial on support vector machines for pattern recognition. Data mining and knowledge discovery, 2(2), 121-167.

Chandrashekar, G., & Sahin, F., 2014. A survey on feature selection methods. Computers & Electrical Engineering, 40(1), 16-28.

Cover, T. M., & Hart, P. E., 1967. Nearest neighbor pattern classification. Information Theory, IEEE Transactions on, 13(1), 21-27.

Demyanov, A., 2005. On the solution of min-sum-min problems. Journal of Global Optimization, 31(3), 437-453.

Dua, D. and Karra Taniskidou, E., 2017. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Eckstein, J., Hammer, P. L., Liu, Y., Nediak, M., and Simeone, B., 2002. The maximum box problem and its application to data analysis. Computational Optimization and Applications, 23, 285 – 298.

Engels, R., 1999. Component-Based User Guidance in Knowledge Discovery and Data Mining. Sankt Augustin: Infix.

Fisher, R. A., 1936. The use of multiple measurements in taxonomic problems. Annals of eugenics, 7(2), 179-188.

Gasimov, R. N., and Ozturk, G., 2006. Separation via polyhedral conic functions. Optimisation Methods and Software, 21(4), 527-540.

Glahn, H. R., 1968. Canonical correlation and its relationship to discriminant analysis and multiple regression. Journal of the atmospheric sciences, 25(1), 23-31.

Golub T.R., Slonim, D.K., Tamayo, C., et al., 1999. Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. Science, 5439(286), 531-537.

Gordon, G.J., Jensen R.V., Hsiao, L.L., et al., 2002. Translation of microarray data into clinically relevant cancer diagnostic tests using gene expression ratios in lung cancer and mesothelioma. Cancer Research, 62, 4963-4967.

Hammer, P. L., Liu, Y., Simeone, B., and Szedmák, S., 2004. Saturated system of homogeneous boxes and logical analysis of numerical data. Discrete Applied Mathematics, 144 (2004) 103 – 109.

Kotsiantis, S. B., 2007. Supervised Machine Learning: A Review of Classification Techniques. Informatica, 31, 249-268.

Kumar, V., and Minz, S., 2014. Feature Selection. SmartCR, 4(3), 211-229.

Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R.P, Tang, J., and Liu, H., 2017. Feature selection: A data perspective. ACM Computing Surveys (CSUR), 50(6): 94:1-94:45, 2017.

Lovell, M. C., 1983. Data Mining, The Review of Economics and Statistics, vol. 65, no:1, 1-12.

Kuncheva, L.I., Rodriguez, J.J, 2018. On feature selection protocols for very low-sample-size data. Pattern Recognition, 81, 660-673.

Mafarja, M., Aljarah, I., Heidari, A. A., Hammouri, A. I., Faris, H., Al-Zoubi, A., Mirjalili, S., 2018. Evolutionary population dynamics and grasshopper optimization approaches for feature selection problems. Knowledge-Based Systems, vol. 145, 25-45.

Maskooki, A., 2013. Improving the efficiency of a mixed integer linear programming based approach for multi-class classification problem. Computer and Industrial Engineering, 66, 383-388.

Murthy, S. K., 1998. Automatic construction of decision trees from data: A multi-disciplinary survey. Data mining and knowledge discovery, 2(4), 345-389.

Park, Y., and Sklansky, J., 1989. Automated design of multiple-class piecewise linear classifiers. Journal of Classification, 6(1), 195-222.

Orsenigo, C., and Vercellis, C., 2007. Accurately learning from few examples with a polyhedral classifier. Computational optimization and applications, 38(2), 235-247.

Saeys, Y., Inza, I., and Larrañaga, P., 2007. A review of feature selection techniques in bioinformatics. bioinformatics, 23(19), 2507-2517.

Scholkopft, B.,and Mullert, K. R., 1999. Fisher discriminant analysis with kernels. Neural networks for signal processing IX, 1, 1.

Schulmeister, B., and Wysotzki, F., 1994. The piecewise linear classifier DIPOL92. In Machine Learning: ECML-94 (pp. 411-414). Springer Berlin Heidelberg.

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

Shunmugapriya, P., Kanmani, S., 2017. A hybrid algorithm using ant and bee colony optimization for feature selection and classification (AC-ABC Hybrid). Swarm and Evolutionary Computation. Vol. 36, 27–36.

Simpson, P., 1991. Fuzzy min-max classification with neural networks. Heuristics, vol. 4, no.1, 1-9.

Sklansky, J., and Michelotti, L., 1980. Locally trained piecewise linear classifiers. Pattern Analysis and Machine Intelligence, IEEE Transactions on, (2), 101-111.

Spinelli, V., 2014. Prunning boxes in a box-based classification method. Advances in Data Analysis and Classification, 10 (3), 285 – 304.

Uney, F., Turkay, M., 2006. A mixed-integer programming approach to multi-class data classification problem. European Journal of Operational Research, 173, 910-920.

Xu, G., Papageorgiou, L. G., 2009. An MILP model for multi-class data classification. Computer and Industrial Engineering, 56, 1205-1215.

Yang, L., Liu, S., Tsoka, S., and Papageorgiou, L. G., 2015, Sample re-weighting hyper-box classifiers for multi-class data. Computer and Industrial Engineering, 85, 44-56.

Zhang, G. P., 2000. Neural networks for classification: a survey. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 30(4), 451-462.

APPENDICES

A. Experimental Results for Objective Function Coefficients of Model-MOB

Table A1. Test of Coefficients for Simulated 2

	Coefficients			Training results				Test results				
								1st step results			Final results	
Run	**c_1**	**c_2**	**c_3**	**B**	**M**	**O**	**A**	**M**	**O**	**U**	**M**	**A**
1	1.00	1.00	1.00	3	12	0	0.800	8	0	6	8	0.733
2	1.00	1.00	0.10	3	0	31	0.483	2	11	5	11	0.633
3	1.00	1.00	0.01	2	0	46	0.233	5	24	0	18	0.400
4	1.00	0.10	1.00	2	15	0	0.750	5	0	6	5	0.833
5	1.00	0.10	0.10	2	15	0	0.750	5	0	6	5	0.833
6	1.00	0.10	0.01	2	0	46	0.233	5	24	0	18	0.400
7	1.00	0.01	1.00	2	15	0	0.750	5	0	6	5	0.833
8	1.00	0.01	0.10	2	15	0	0.750	5	0	6	5	0.833
9	1.00	0.01	0.01	2	15	0	0.750	5	0	7	5	0.833
10	0.10	1.00	1.00	14	0	0	1.000	3	0	15	5	0.833
11	0.10	1.00	0.10	10	0	4	0.933	0	3	15	2	0.933
12	0.10	1.00	0.01	3	0	31	0.483	2	11	5	11	0.633
13	0.10	0.10	1.00	5	6	0	0.900	4	0	13	5	0.833
14	0.10	0.10	0.10	3	12	0	0.800	4	0	5	4	0.867
15	0.10	0.10	0.01	3	0	31	0.483	2	11	5	11	0.633
16	0.10	0.01	1.00	2	15	0	0.750	5	0	6	5	0.833
17	0.10	0.01	0.10	2	15	0	0.750	5	0	7	5	0.833
18	0.10	0.01	0.01	2	15	0	0.750	5	0	6	5	0.833
19	0.01	1.00	1.00	14	0	0	1.000	3	0	15	5	0.833
20	0.01	1.00	0.10	14	0	0	1.000	3	0	15	5	0.833
21	0.01	1.00	0.01	10	0	4	0.933	0	3	15	4	0.867
22	0.01	0.10	1.00	14	0	0	1.000	3	0	15	5	0.833
23	0.01	0.10	0.10	14	0	0	1.000	0	3	15	4	0.867
24	0.01	0.10	0.01	10	0	4	0.933	0	4	12	4	0.867
25	0.01	0.01	1.00	14	0	0	1.000	3	0	15	5	0.833
26	0.01	0.01	0.10	5	6	0	0.900	4	0	13	5	0.833
27	0.01	0.01	0.01	5	6	0	0.900	4	0	13	5	0.833

B: Number of hyperboxes
M: Number of misclassified samples
O: Number of overlap samples
U: Number of uncovered samples
A: Accuracy

Table A.2. Test of Coefficients for Iris

	Coefficients			Training results				Test results				
								1st step results			Final results	
Run	**c_1**	**c_2**	**c_3**	**B**	**M**	**O**	**A**	**M**	**O**	**U**	**M**	**A**
1	1.00	1.00	1.00	4	0	3	0.975	1	1	1	1	0.967
2	1.00	1.00	0.10	3	0	6	0.950	0	3	1	1	0.967
3	1.00	1.00	0.01	3	0	6	0.950	1	2	1	2	0.933
4	1.00	0.10	1.00	3	4	0	0.967	4	0	1	4	0.867
5	1.00	0.10	0.10	3	2	2	0.967	1	1	1	1	0.967
6	1.00	0.10	0.01	3	0	6	0.950	1	2	1	1	0.967
7	1.00	0.01	1.00	3	4	0	0.967	5	0	1	5	0.833
8	1.00	0.01	0.10	3	4	0	0.967	1	4	1	2	0.933
9	1.00	0.01	0.01	3	1	3	0.967	1	1	1	1	0.967
10	0.10	1.00	1.00	4	2	0	0.983	1	0	2	1	0.967
11	0.10	1.00	0.10	4	0	2	0.983	1	1	0	2	0.933
12	0.10	1.00	0.01	3	0	6	0.950	0	3	1	1	0.967
13	0.10	0.10	1.00	5	0	0	1.000	1	0	5	1	0.967
14	0.10	0.10	0.10	3	0	6	0.950	0	3	1	3	0.900
15	0.10	0.10	0.01	3	0	6	0.950	0	3	1	2	0.933
16	0.10	0.01	1.00	3	12	0	0.900	1	2	1	3	0.900
17	0.10	0.01	0.10	3	4	0	0.967	5	0	1	5	0.833
18	0.10	0.01	0.01	3	4	0	0.967	5	0	1	5	0.833
19	0.01	1.00	1.00	4	0	0	1.000	1	1	2	1	0.967
20	0.01	1.00	0.10	4	0	3	0.975	1	1	1	2	0.933
21	0.01	1.00	0.01	3	0	6	0.950	1	2	1	1	0.967
22	0.01	0.10	1.00	4	2	0	0.983	1	0	1	1	0.967
23	0.01	0.10	0.10	4	2	0	0.983	1	0	1	1	0.967
24	0.01	0.10	0.01	4	0	1	0.992	1	1	2	1	0.967
25	0.01	0.01	1.00	4	2	0	0.983	1	0	1	1	0.967
26	0.01	0.01	0.10	4	2	0	0.983	1	0	1	1	0.967
27	0.01	0.01	0.01	3	1	3	0.967	1	1	1	2	0.933

B: Number of hyperboxes
M: Number of misclassified samples
O: Number of overlap samples
U: Number of uncovered samples
A: Accuracy

Table A.3. Test of Coefficients for Wine

	Coefficients			Training results				Test results				
								1st step results			Final results	
Run	**c_1**	**c_2**	**c_3**	**B**	**M**	**O**	**A**	**M**	**O**	**U**	**M**	**A**
1	1.00	1.00	1.00	3	0	7	0.951	1	0	12	1	0.972
2	1.00	1.00	0.10	3	0	7	0.951	0	1	11	0	1.000
3	1.00	1.00	0.01	3	0	7	0.951	0	1	11	0	1.000
4	1.00	0.10	1.00	3	0	7	0.951	1	0	12	1	0.972
5	1.00	0.10	0.10	3	0	7	0.951	0	1	12	1	0.972
6	1.00	0.10	0.01	3	0	7	0.951	1	0	12	1	0.972
7	1.00	0.01	1.00	3	0	7	0.951	1	0	12	1	0.972
8	1.00	0.01	0.10	3	0	7	0.951	1	1	12	1	0.972
9	1.00	0.01	0.01	3	0	7	0.951	0	1	11	0	1.000
10	0.10	1.00	1.00	3	0	7	0.951	0	1	11	0	1.000
11	0.10	1.00	0.10	3	0	7	0.951	0	1	12	1	0.972
12	0.10	1.00	0.01	3	0	7	0.951	0	1	12	0	1.000
13	0.10	0.10	1.00	3	3	2	0.965	1	1	13	2	0.944
14	0.10	0.10	0.10	3	0	7	0.951	1	0	12	1	0.972
15	0.10	0.10	0.01	3	0	7	0.951	0	1	11	0	1.000
16	0.10	0.01	1.00	3	6	1	0.951	0	0	11	0	1.000
17	0.10	0.01	0.10	3	0	7	0.951	1	0	12	1	0.972
18	0.10	0.01	0.01	3	0	7	0.951	0	1	12	1	0.972
19	0.01	1.00	1.00	3	0	7	0.951	0	1	11	0	1.000
20	0.01	1.00	0.10	3	0	7	0.951	0	1	11	0	1.000
21	0.01	1.00	0.01	3	0	7	0.951	0	1	11	0	1.000
22	0.01	0.10	1.00	3	3	2	0.965	1	1	13	2	0.944
23	0.01	0.10	0.10	3	0	7	0.951	1	1	13	2	0.944
24	0.01	0.10	0.01	3	0	7	0.951	0	1	12	0	1.000
25	0.01	0.01	1.00	3	6	1	0.951	1	1	12	2	0.944
26	0.01	0.01	0.10	3	6	1	0.951	1	1	13	2	0.944
27	0.01	0.01	0.01	3	0	7	0.951	1	0	12	1	0.972

B: Number of hyperboxes
M: Number of misclassified samples
O: Number of overlap samples
U: Number of uncovered samples
A: Accuracy

Table A.4. Test of coefficients for Seed

	Coefficients			Training results				Test results				
								1st step results			Final results	
Run	**c_1**	**c_2**	**c_3**	**B**	**M**	**O**	**A**	**M**	**O**	**U**	**M**	**A**
1	1.00	1.00	1.00	3	1	37	0.775	1	12	3	5	0.881
2	1.00	1.00	0.10	3	0	33	0.805	2	9	3	4	0.905
3	1.00	1.00	0.01	3	0	33	0.805	2	9	3	8	0.810
4	1.00	0.10	1.00	3	2	24	0.846	2	7	4	7	0.833
5	1.00	0.10	0.10	3	0	33	0.805	2	9	3	4	0.905
6	1.00	0.10	0.01	3	0	57	0.663	1	11	2	5	0.881
7	1.00	0.01	1.00	3	0	33	0.805	2	9	3	9	0.786
8	1.00	0.01	0.10	3	57	4	0.639	20	4	1	21	0.500
9	1.00	0.01	0.01	3	0	33	0.805	2	9	3	4	0.905
10	0.10	1.00	1.00	3	54	16	0.586	8	1	3	9	0.786
11	0.10	1.00	0.10	3	0	36	0.787	2	9	3	10	0.762
12	0.10	1.00	0.01	3	0	33	0.805	2	9	3	4	0.905
13	0.10	0.10	1.00	3	111	23	0.207	19	4	1	23	0.452
14	0.10	0.10	0.10	3	0	35	0.793	2	9	3	4	0.905
15	0.10	0.10	0.01	3	0	33	0.805	2	9	3	3	0.929
16	0.10	0.01	1.00	3	0	38	0.775	2	10	3	5	0.881
17	0.10	0.01	0.10	3	1	35	0.787	2	9	2	4	0.905
18	0.10	0.01	0.01	3	0	33	0.805	2	9	3	4	0.905
19	0.01	1.00	1.00	3	0	33	0.805	2	9	3	3	0.929
20	0.01	1.00	0.10	3	0	63	0.627	14	7	2	16	0.619
21	0.01	1.00	0.01	3	0	33	0.805	2	9	3	4	0.905
22	0.01	0.10	1.00	4	15	39	0.680	3	5	3	4	0.905
23	0.01	0.10	0.10	4	0	33	0.805	2	9	5	9	0.786
24	0.01	0.10	0.01	3	0	33	0.805	2	9	3	4	0.905
25	0.01	0.01	1.00	3	0	40	0.763	3	10	2	12	0.714
26	0.01	0.01	0.10	3	1	35	0.787	1	12	3	5	0.881
27	0.01	0.01	0.01	3	0	35	0.793	2	9	3	4	0.905

B: Number of hyperboxes
M: Number of misclassified samples
O: Number of overlap samples
U: Number of uncovered samples
A: Accuracy

B. Cross-validation Results for Model-MOB

Table B.1. *Results of cross-validations for Model-MOB*

		Training results				Test results				
						1st step results			Final results	
	Run	B	M	O	A	M	O	U	M	A
simulated 1	1	7	0	0	1.000	1	0	7	1	0.967
	2	8	0	0	1.000	3	0	4	3	0.900
	3	7	0	0	1.000	1	2	5	2	0.933
	4	6	0	0	1.000	2	0	4	2	0.933
	5	6	0	0	1.000	0	0	6	0	1.000
simulated 2	1	14	0	0	1.000	3	0	15	5	0.833
	2	14	0	0	1.000	5	1	5	6	0.800
	3	13	0	0	1.000	4	0	8	5	0.833
	4	11	0	0	1.000	3	0	5	4	0.867
	5	13	0	0	1.000	4	1	5	4	0.867
Iris	1	4	0	0	1.000	1	1	2	1	0.967
	2	4	0	1	0.992	1	2	1	1	0.967
	3	4	1	1	0.983	0	0	7	0	1.000
	4	5	0	0	1.000	0	3	1	1	0.967
	5	4	0	0	1.000	2	0	3	2	0.933
Wine	1	3	0	7	0.951	0	1	11	0	1.000
	2	3	0	5	0.965	1	0	15	2	0.944
	3	3	0	8	0.944	2	2	10	4	0.889
	4	3	0	7	0.951	0	1	11	1	0.972
	5	3	0	10	0.930	2	1	9	2	0.944
Seed	1	3	0	33	0.804	2	9	3	3	0.929
	2	4	1	34	0.792	1	8	1	2	0.933
	3	3	0	35	0.792	2	9	3	3	0.900
	4	3	2	26	0.833	3	5	5	3	0.900
	5	4	0	39	0.768	2	10	0	4	0.867

B: Number of hyperboxes
M: Number of misclassified samples
O: Number of overlap samples
U: Number of uncovered samples
A: Accuracy

C. Cross-validation Results for Model-MO

Table C.1. *Results of Cross-validations for Model-MO*

		Scenerio 1					Scenerio 2				
	Run	**L**	**B**	**M**	**O**	**A**	**L**	**B**	**M**	**O**	**A**
simulated 1	**1**	20	7	0	0	1.000	4	4	4	0	0.933
	2		8	0	0	1.000		4	4	0	0.933
	3		7	0	0	1.000		4	4	0	0.933
	4		6	0	0	1.000		4	4	0	0.933
	5		6	0	0	1.000		4	3	0	0.950
simulated 2	**1**	20	15	0	0	1.000	4	4	10	0	0.833
	2		10	2	0	0.967		4	8	0	0.867
	3		14	0	0	1.000		4	9	0	0.850
	4		11	0	0	1.000		4	1	0	0.983
	5		11	0	0	1.000		4	6	0	0.900
Iris	**1**	10	6	0	0	1.000	6	4	1	0	0.992
	2		7	0	0	1.000		4	1	0	0.992
	3		7	0	0	1.000		5	1	1	0.983
	4		7	0	0	1.000		5	1	0	0.992
	5		7	0	0	1.000		5	1	0	0.992
Wine	**1**	8	4	2	1	0.979	6	4	0	2	0.986
	2		4	0	3	0.979		4	0	3	0.979
	3		4	1	8	0.937		4	0	9	0.937
	4		4	0	2	0.986		4	0	2	0.986
	5		3	1	7	0.944		4	0	4	0.972
Seed	**1**	15	3	8	7	0.911	6	4	4	4	0.952
	2		3	8	8	0.905		4	4	4	0.952
	3		3	6	22	0.833		5	5	0	0.970
	4		3	4	13	0.899		4	6	0	0.964
	5		3	7	6	0.923		4	6	1	0.958

Table C.1 (continued)

		Scenerio 1					Scenerio 2				
	Run	**L**	**B**	**M**	**O**	**A**	**L**	**B**	**M**	**O**	**A**
Breast Cancer	**1**	8	2	19	288	0.438	4	2	35	145	0.670
	2		2	1	121	0.777		3	5	88	0.830
	3		2	12	96	0.802		4	11	131	0.740
	4		3	7	109	0.788		2	8	111	0.782
	5		2	0	413	0.244		2	10	114	0.773
Ecoli	**1**	15	6	0	131	0.394	10	5	9	83	0.648
	2		5	1	99	0.537		7	16	5	0.920
	3		5	0	138	0.361		9	19	20	0.851
	4		5	3	119	0.449		5	27	2	0.889
	5		5	0	133	0.384		6	21	4	0.904
Sonar	**1**	8	2	0	91	0.452	4	3	1	72	0.560
	2		2	8	49	0.657		3	16	50	0.602
	3		2	5	74	0.524		3	19	45	0.614
	4		2	13	64	0.536		4	12	45	0.657
	5		2	16	45	0.633		4	15	46	0.633
Ionosphere	**1**	8	2	96	0	0.657	4	2	96	0	0.657
	2		2	97	0	0.654		2	97	0	0.654
	3		2	100	0	0.643		2	100	0	0.643
	4		2	108	0	0.614		2	108	0	0.614
	5		2	102	0	0.636		2	102	0	0.636
Glass	**1**	10	5	56	61	0.316	10	5	47	58	0.386
	2		8	18	76	0.450		6	4	102	0.380
	3		7	57	49	0.380		5	9	69	0.544
	4		7	14	91	0.386		7	50	55	0.386
	5		8	20	78	0.427		7	3	91	0.450

B: Number of hyperboxes
M: Number of misclassified samples
O: Number of overlap samples
U: Number of uncovered samples
A: Accuracy

D. Results of Model-MO for All Nodes in Search Tree

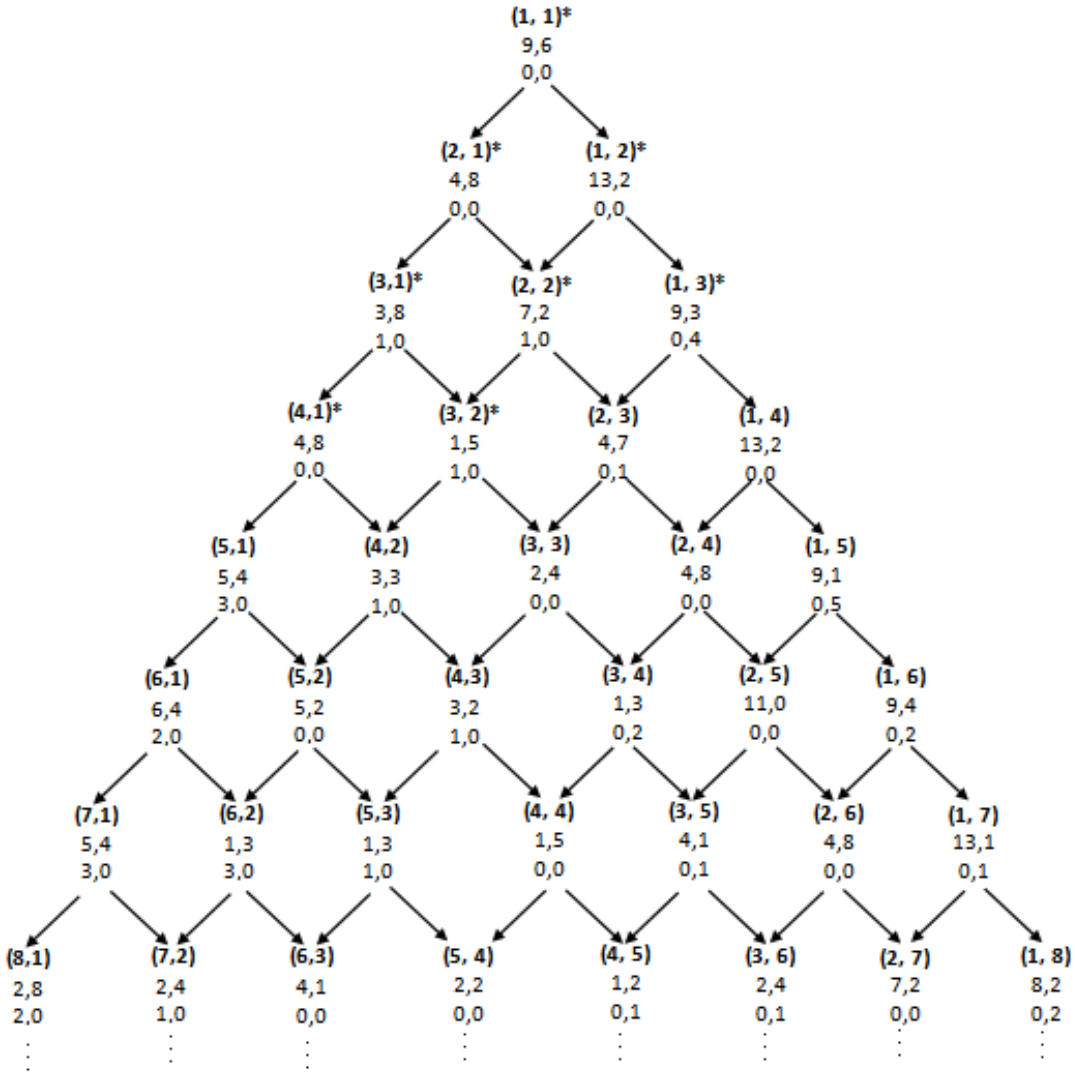


Figure D.1. Model-MO results for Simulated 2 dataset for all nodes

Note: 100% accuracy is achieved at node (8,6)

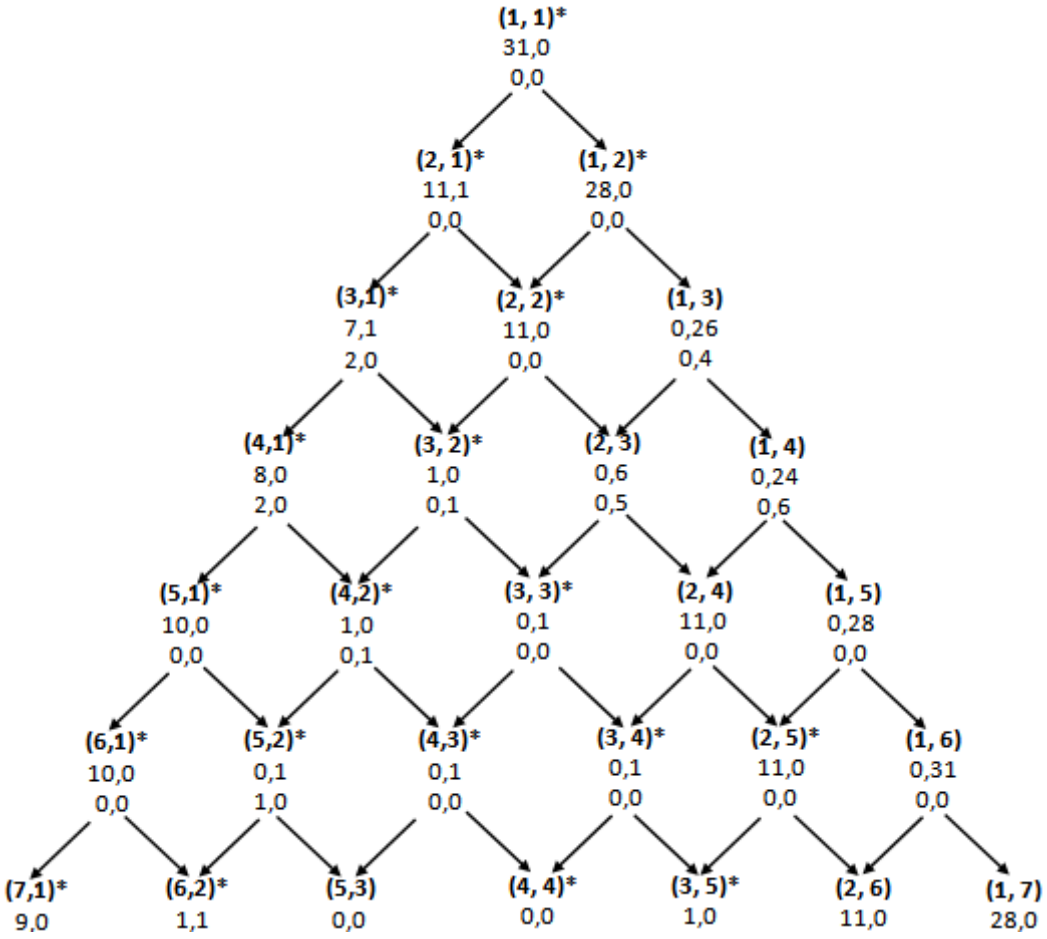


Figure D.2. Model-MO results of Simulated 3 dataset for all nodes

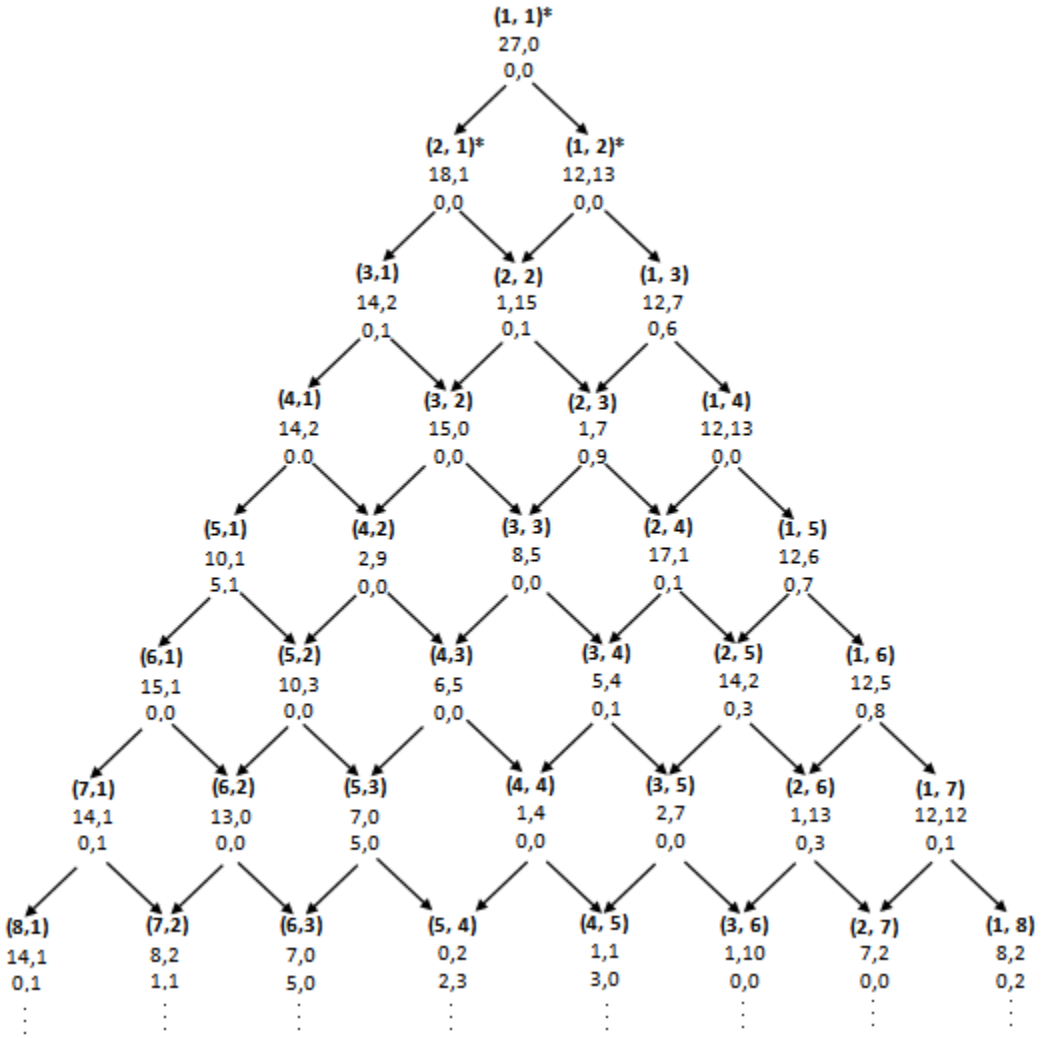


Figure D.3. Model-MO results for Simulated 4 dataset for all nodes

Note: 100% accuracy is achieved at node (7,5)

E. Preliminary Results of HCB

Table E.1. Preliminary Result of HCB for Simulated 2 dataset

Iter.	Alloc.	M	O	U	A_M	Class of Trimmed hyperbox	# of samples to be fixed	A_{TB}	Model CPU time (sec)	HTA CPU time (sec)
1	(1,1)	12,2	0,0	1,0	75.00	1, 2	25	41.67	11.481	0.171
2	(2,2)	5,10	0,0	1,0	73.33	2	9	56.67	6.747	0.047
3	(2,3)	5,0	0,0	5,3	78.33	1, 2	6	66.67	5.885	0.102
4	(3,3)	3,3	0,0	4,1	81.67	1, 2	6	76.67	6.043	0.149
5	(4,4)	0,1	0,1	3,3	86.67	2	2	80.00	5.524	1.719
6	(4,5)	0,1	0,1	3,2	88.33	1, 2	3	85.00	5.12	1.813
7	(5,6)	0,0	0,1	3,2	90.00	1	2	88.33	5.248	2.317
8	(6,6)	0,0	0,0	3,2	91.67	1	1	90.00	5.565	2.375
9	(7,6)	0,0	0,0	2,2	93.33	1, 2	2	93.33	4.794	1.703
10	(8,7)	0,0	0,1	0,1	96.67	2	1	95.00	5.562	1.43
11	(8,8)	1,0	0,0	0,0	98.33	1	1	96.67	3.216	0.216
12	(9,8)	0,0	0,0	0,0	100.00			100.0	3.685	

Table E.2. Preliminary Result of HCB for Simulated 4 dataset

Iter.	Alloc.	M	O	U	A_M	Class of Trimmed hyperbox	# of samples to be fixed	A_{TB}	Model CPU time (sec)	HTA CPU time (sec)
1	(1,1)	17,0	0,0	10,0	73.00	1	23	23.00	17.344	4.766
2	(2,1)	14,3	0,0	2,0	81.00	1, 2	21	44.00	13.474	0.312
3	(3,2)	14,1	0,0	1,2	82.00	1, 2	15	59.00	10.259	0.112
4	(4,3)	9,0	0,1	4,7	79.00	2	11	70.00	8.371	1.320
5	(4,4)	0,0	0,0	8,5	87.00	1, 2	17	87.00	8.677	2.922
6	(5,5)	0,0	0,0	4,2	94.00	1	4	91.00	6.534	1.219
7	(6,5)	0,0	0,0	2,2	96.00	2	3	94.00	6.639	1.227
8	(6,6)	0,0	0,0	2,0	98.00	1	2	96.00	6.465	1.367
9	(7,6)	0,0	0,0	1,0	99.00	1	1	97.00	4.268	1.102
10	(8,6)	0,0	0,0	0,0	100.00					

Table E.3. Preliminary Result for HCB for Breast Cancer dataset

Iter.	Alloc.	M	O	U	A_M	Class of Trimmed hyperbox	# of samples to be fixed	A_{TB}	Model CPU time (sec)	HTA CPU time (sec)
1	(1,1)	11,3	5,7	8,7	92.49	1	224	41.03	1200.00	0.273
2	(2,1)	9,2	5,3	16,22	89.93	1, 2	214	80.22	1200.00	0.534
3	(3,2)	3,9	0,1	11,53	85.90	2	21	84.07	1200.00	1.848
4	(3,3)	4,1	1,0	13,39	89.38	1, 2	15	86.81	1200.00	1.416
5	(4,4)	4,2	4,0	8,16	91.39	1, 2	13	89.19	1200.00	1.500
6	(5,5)	2,1	0,0	5,27	93.59	1, 2	15	91.94	1200.00	4.218
7	(6,6)	1,1	0,0	3,18	95.79	2	9	93.59	750.886	2.119
8	(6,7)	2,0	0,0	3,12	96.89	1, 2	6	94.69	1200.00	1.423
9	(7,8)	0,0	0,0	1,10	97.99	2	4	95.42	216.04	1.594
10	(7,9)	0,0	0,0	1,7	98.53	2	3	95.97	112.224	1.348
11	(7,10)	0,0	0,0	1,4	99.08	2	3	96.52	54.009	1.317
12	(7,11)	0,0	0,1	1,1	99.45	1, 2	4	97.25	35.994	1.5
13	(8,12)	0,0	0,0	0,0	100.0			100.0		

F. Iteration Details of HCB

Table F.1. Iteration Details of HCB – Simulated 4 dataset (remaining folds)

	Training Phase												Test Phase	
				Model-MOU Results				Class of trimmed hyperbox	# of samples to be fixed		CPU times (sec.)			
	Iter.	L	Box alloc.	M	O	U	A_M			A_{TB}	Model MOU	HTA	M	A
Fold 2	1	2	(1,1)	18,0	2,2	7,0	71.0	1	13	13.0	29.08	2.58	6	76.0
	2	3	(2,1)	11,5	0,0	0,8	76.0	2	10	23.0	20.16	1.91	7	72.0
	3	4	(2,2)	11,5	0,0	0,8	76.0	1	10	33.0	17.08	2.49	6	76.0
	4	5	(3,2)	6,8	0,1	2,1	81.0	2	9	42.0	15.71	1.92	7	72.0
	5	6	(3,3)	14,0	0,1	0,3	82.0	1	8	50.0	14.91	0.82	7	72.0
	6	7	(4,3)	5,0	1,0	0,10	84.0	2	8	58.0	15.77	0.11	5	80.0
	7	8	(4,4)	4,0	0,0	1,6	89.0	2	8	66.0	13.54	2.54	4	84.0
	8	9	(4,5)	5,0	0,0	3,3	89.0	1	5	71.0	14.23	2.25	4	84.0
	9	10	(5,5)	0,0	0,0	6,0	94.0	1	14	85.0	12.63	1.72	1	96.0
	10	11	(6,5)	0,0	0,0	2,0	98.0	1	2	87.0	12.85	0.52	3	88.0
	11	12	(7,5)	0,0	0,0	0,0	100.0		13	100.0	13.23		1	96.0
Fold 3	1	2	(1,1)	21,0	0,0	4,7	68.0	1	21	21.0	34.61	1.50	5	80.0
	2	3	(2,1)	17,2	0,1	0,1	79.0	1-2	32	53.0	30.21	2.16	5	80.0
	3	5	(3,2)	1,4	0,6	3,4	82.0	2	9	62.0	13.47	1.23	7	72.0
	4	6	(3,3)	1,4	0,0	5,2	88.0	1-2	18	8.0	12.51	0.15	6	76.0
	5	8	(4,4)	2,0	0,0	5,2	91.0	1	5	85.0	13.61	1.26	4	84.0
	6	9	(5,4)	0,0	0,0	3,2	95.0	1	5	90.0	12.17	1.03	3	88.0
	7	10	(6,4)	0,0	0,0	1,2	97.0	2	5	95.0	12.10	1.14	3	88.0
	8	11	(6,5)	0,0	0,0	1,0	99.0	1	3	98.0	12.17	1.12	3	88.0
	9	12	(7,5)	0,0	0,0	0,0	100.0		2	100.0	12.35		3	88.0
Fold 4	1	2	(1,1)	25,0	0,0	0,0	75.0	1	20	20.0	19.25	1.46	11	56.0
	2	3	(2,1)	16,2	0,0	0,0	82.0	1-2	16	36.0	16.01	0.11	7	72.0
	3	5	(3,2)	15,0	0,0	0,3	82.0	1-2	11	47.0	12.21	1.47	9	64.0
	4	7	(4,3)	7,0	3,1	0,4	85.0	1	21	68.0	10.55	0.09	7	72.0
	5	9	(5,4)	7,0	3,0	1,3	86.0	1-2	15	83.0	7.84	1.06	6	76.0
	6	11	(6,5)	2,0	0,0	3,3	92.0	1-2	6	89.0	6.81	1.45	6	76.0
	7	13	(7,6)	1,0	0,0	2,3	94.0	1-2	5	94.0	7.02	1.71	6	76.0
	8	15	(8,7)	1,1	0,0	0,0	98.0	1-2	4	98.0	7.01	0.16	4	84.0
	9	17	(9,8)	0,0	0,0	0,0	100.0		2	100.0	7.27		4	84.0
Fold 5	1	2	(1,1)	21,2	0,0	0,4	73.0	1	20	20.0	44.41	0.12	8	68.0
	2	3	(2,1)	14,1	0,0	5,5	75.0	1-2	18	38.0	15.61	0.09	5	80.0
	3	5	(3,2)	2,15	0,0	4,0	79.0	1	21	59.0	34.67	0.12	4	84.0
	4	6	(3,3)	0,1	0,0	12,4	83.0	1-2	21	80.0	9.36	2.00	4	84.0
	5	8	(4,4)	3,0	0,0	7,1	89.0	1	6	86.0	7.50	1.57	3	88.0
	6	9	(5,4)	4,0	0,0	2,2	92.0	1	3	89.0	7.00	1.27	4	84.0
	7	10	(6,4)	3,0	0,0	1,2	95.0	1	3	92.0	7.36	1.96	2	92.0
	8	11	(7,4)	0,0	0,0	1,2	97.0	1-2	5	97.0	6.68	1.78	3	88.0
	9	13	(8,5)	0,0	0,0	0,1	99.0	2	1	98.0	6.57	0.21	2	92.0
	10	14	(8,6)	0,0	0,0	0,0	100.0		2	100.0	6.62		2	92.0

Table F.2. Iteration Details of HCB – Breast Cancer dataset (remaining folds)

	Training Phase												Test Phase	
				Model-MOU Results				Class of trimmed hyperbox	# of samples to be fixed		CPU times (sec.)			
	Iter	L	Box alloc.	M	O	U	A_M			A_{TB}	Model MOU	HTA	M	A
Fold 2	1	2	(1,1)	8,2	3,7	6,2	94.87	1	326	59.71	1200	0.195	9	93.33
	2	3	(2,1)	9,0	6,0	9,9	93.96	1 - 2	96	77.29	1200	0.595	10	92.66
	3	5	(3,2)	3,4	0,0	9,67	84.80	2	30	82.78	1200	1.196	7	94.81
	4	6	(3,3)	1,1	3,1	10,51	87.73	2	16	85.71	1200	1.164	7	94.81
	5	7	(3,4)	1,2	0,0	9,45	89.56	2	11	87.73	1200	1.090	6	95.56
	6	8	(3,5)	0,0	1,0	14,32	91.39	2	13	90.11	1200	2.446	8	94.07
	7	9	(3,6)	1,2	2,3	16,27	92.31	2	5	91.03	1200	2.105	7	94.81
	8	10	(3,7)	0,1	1,0	11,24	93.22	2	6	92.12	1200	1.346	7	94.81
	9	11	(3,8)	0,1	1,1	9,15	95.05	2	10	93.96	1200	3.150	7	94.81
	10	12	(3,9)	0,1	1,1	9,11	95.05	1 - 2	10	95.79	1200	2.030	7	94.81
	11	14	(4,10)	0,1	1,1	5,10	97.44	2	3	96.34	213	1.940	7	94.81
	12	15	(4,11)	0,0	1,1	5,7	97.44	2	3	96.89	654	0.960	7	94.81
	13	16	(4,12)	0,0	0,1	6,7	97.99	2	4	97.62	458	0.765	7	94.81
	14	17	(4,13)	0,0	0,1	6,4	99.08	1	3	98.17	957	1.120	7	94.81
	15	18	(5,13)	0,0	0,1	0,1	99.63	1	4	98.90	546	0.612	7	94.81
	16	19	(6,13)				100.0		6	100.0	321		7	94.81
Fold 3	1	2	(1,1)	9,6	6,9	2,4	93.41	1 - 2	476	87.18	1200	2.174	10	92.75
	2	4	(2,2)	4,2	1,1	8,36	90.48	2	17	90.29	1200	3.693	8	94.20
	3	5	(2,3)	4,2	1,1	5,15	94.87	2	21	94.14	867	3.364	10	92.75
	4	6	(2,4)	6,2	1,1	3,6	96.52	2	9	95.79	91	2.772	8	94.20
	5	7	(3,4)	5,1	1,1	1,7	97.07	1	8	97.25	54	5.054	8	94.20
	6	8	(3,5)	4,1	2,2	1,2	97.80	2	6	98.35	38	5.260	9	93.48
	7	10	(4,6)	4,1	1,1	0,1	98.53	1 - 2	4	99.08	27	0.303	9	93.48
	8	12	(5,7)	1,0	0,1	0,0	99.63	1 - 2	4	99.82	27	2.850	9	93.48
	9	13	(6,7)				100.0		1	100.0	28		9	93.48
Fold 4	1	2	(1,1)	9,5	9,5	1,5	93.96	1	338	61.90	1200	0.453	9	93.50
	2	3	(2,1)	12,2	6,0	3,8	94.32	1 - 2	85	77.47	1200	0.655	9	93.50
	3	5	(3,2)	12,2	6,0	0,9	94.69	1 - 2	35	83.88	1200	0.824	8	94.20
	4	7	(4,3)	11,2	3,4	0,8	94.87	2	21	87.73	1200	1.036	6	95.70
	5	8	(4,4)	10,2	7,0	0,9	94.87	1 - 2	13	90.11	1200	1.043	8	94.20
	6	10	(5,5)	6,2	4,0	1,14	94.87	2	12	92.31	1200	1.254	7	94.90
	7	11	(5,6)	6,3	2,0	1,14	95.24	2	12	94.51	1200	1.277	6	95.70
	8	12	(5,7)	0,3	0,1	3,12	96.52	2	11	96.52	1200	1.402	6	95.70
	9	13	(5,8)	3,2	0,0	1,4	98.17	2	6	97.62	645	1.325	6	95.70
	10	14	(5,9)	1,1	0,0	1,2	99.08	1	8	99.08	846	2.005	6	95.70
	11	15	(6,9)	0,0	0,0	0,0	100.0		5	100.0	191		6	95.70

Table F.2 (continue)

				Training Phase									Test Phase	
				Model-MOU Results				Class of trimmed hyperbox	# of samples to be fixed		CPU times (sec.)			
	Iter.	L	Box alloc.	M	O	U	A_M			A_{TB}	Model MOU	HTA	M	A
Fold 5	1	2	(1,1)	4,2	3,5	14,6	93.77	1	296	54.21	1200	1.409	6	95.60
	2	3	(2,1)	4,2	6,8	4,12	93.41	1 - 2	102	72.89	1200	2.124	6	95.60
	3	5	(3,2)	6,1	4,2	4,8	95.42	1	18	76.19	1200	1.264	7	94.90
	4	6	(4,2)	6,1	2,2	3,10	95.60	2	46	84.62	1200	2.525	6	95.60
	5	7	(4,3)	4,1	1,2	5,3	97.07	1 - 2	39	91.76	1200	1.277	7	94.90
	6	9	(5,4)	3,1	1,2	4,9	96.34	2	14	94.32	876	2.003	8	94.20
	7	10	(5,5)	2,1	1,1	4,6	97.25	2	6	95.42	594	0.877	7	94.90
	8	11	(5,6)	1,1	1,1	2,6	97.80	2	3	95.97	1200	0.457	7	94.90
	9	12	(5,7)	0,1	0,1	2,6	98.17	1 - 2	8	97.44	648	1.086	7	94.90
	10	14	(6,8)	0,0	0,0	1,4	99.08	2	5	98.35	912	0.526	7	94.90
	11	15	(6,9)	0,0	0,0	1,2	99.45	2	3	98.90	214	2.805	7	94.90
	12	16	(6,10)	0,0	0,0	0,1	99.82	2	4	99.63	178	2.736	7	94.90
	13	17	(6,11)	0,0	0,0	0,0	100.0	2	2	100.0	54		7	94.90

Table F.3. Iteration details of HCB – Skin subset 1

			Training Phase									Test Phase	
			Model-MOU Results				Class of trimmed box	# of samples to be fixed		CPU times (sec.)			
	It.	Box alloc.	M	O	U	A_M			A_{TB}	Model MOU	HTA	M	A
Fold 1	1	(1,1)	400,0	0,0	0,0	71.43	2	547	39.07	4.21	5.28	121	65.43
	2	(1,2)	10,22	2,195	0,2	83.50	2	214	54.36	1200	1.26	39	88.86
	3	(1,3)	0,18	1,83	0,7	92.21	1 - 2	321	77.29	1200	1.68	27	92.29
	4	(2,4)	6,0	0,0	58,20	94.00	1 - 2	193	91.07	1179	3.16	12	96.57
	5	(3,5)	6,0	0,19	36,15	94.57	1	21	92.57	1200	2.12	11	96.86
	6	(4,5)	6,0	1,0	46,16	95.07	1	21	94.07	1200	1.63	12	96.57
	7	(5,5)	9,2	0,0	26,13	96.43	1	13	95.00	1200	2.13	7	98.00
	8	(6,5)	0,0	0,0	26,16	97.00	1	19	96.36	1200	1.82	5	98.57
	9	(7,5)	0,1	0,0	16,15	97.71	1 - 2	20	97.79	637.82	3.11	4	98.86
	10	(8,6)	0,1	0,5	7,8	98.50	1 - 2	16	98.93	140.78	2.69	5	98.57
	11	(9,7)	0,0	0,0	0,4	99.71	1 - 2	11	99.71	38.09	2.10	4	98.86
	12	(9,9)	0,0	0,0	0,0	100.0		4	100.0	90.72		4	98.86
Fold 2	1	(1,1)	400	0,0	0	71.43	2	535	38.21	9.85	3.02	83	76.29
	2	(1,2)	13,6	198,17	0,13	82.36	1	172	50.50	1200	1.28	67	80.86
	3	(2,2)	87,1	0,124	7,12	83.50	2	212	65.64	1200	2.52	47	86.57
	4	(2,3)	87,0	62,8	5,10	87.64	1	142	75.79	687.0	2.14	18	94.86
	5	(3,3)	0,3	0,0	0,100	92.64	2	150	86.50	242.59	1.80	23	93.43
	6	(3,4)	0,3	0,0	11,15	97.93	2	74	91.79	151.4	2.02	9	97.43
	7	(3,5)	0,3	0,0	11,8	98.43	1	67	96.57	48.84	1.90	5	98.57
	8	(4,5)	0,3	0,0	0,16	98.64	1 - 2	29	98.64	52.76	2.42	2	99.43
	9	(5,6)	0,0	0,0	0,12	99.14	1 - 2	10	99.36	24.70	1.98	0	100.0
	10	(5,8)	0,0	0,0	0,0	100.0		9	100.0	27.73		1	99.71
Fold 3	1	(1,1)	400,0	0,0	0,0	71.43	2	541	38.64	9.95	3.43	61	82.57
	2	(1,2)	50,9	158,8	6,12	82.14	1 - 2	453	71.00	1200	2.30	24	93.14
	3	(2,3)	51,2	1,0	25,93	94.00	2	131	80.36	1200	2.27	34	90.29
	4	(2,4)	21,0	1,0	28,34	97.36	1	86	86.50	325.15	1.70	17	95.14
	5	(3,4)	0,11	0,0	0,26	82.14	2	77	92.00	63.02	2.59	7	98.00
	6	(3,5)	0,9	0,1	0,2	99.14	2 - 2	90	98.43	256.86	1.50	8	97.71
	7	(3,7)	0,0	0,0	5,5	99.29	2 - 2	12	99.29	415.72	1.80	4	98.86
	8	(3,9)	0,0	0,0	0,0	100.0		10	100.0	33.04		4	98.86

Table F.3 (continue)

			Training Phase									Test Phase	
			Model-MOU Results				Class of trimmed box	# of samples to be fixed		CPU times (sec.)			
	It.	Box alloc.	M	O	U	A_M			A_{TB}	Model MOU	HTA	M	A
Fold 4	1	(1,1)	400,0	0,0	0,0	71.43	2	557	39.79	35.22	5.91	100	71.43
	2	(1,2)	45,16	142,6	2,37	82.29	1	288	60.36	1200	4.97	51	85.43
	3	(2,2)	24,5	1816,0	0,20	83.21	1 - 2	283	80.57	268.86	4.74	31	91.14
	4	(3,3)	7,0	0,0	0,92	92.93	2	149	91.21	39.55	0.93	24	93.14
	5	(3,4)	0,3	0,0	0,92	97.93	2	70	96.21	24.90	1.36	5	98.57
	6	(3,5)	0,2	0,0	0,14	98.86	1 - 2	37	98.86	34.64	3.42	2	99.43
	7	(3,7)	0,0	0,0	0,5	99.64	1 - 2	14	99.86	41.12	1.37	2	99.43
	8	(3,8)	0,0	0,0	0,0	100.0		2	100.0	30.31	5.79	2	99.43
Fold 5	1	(1,1)	400,0	0,0	0,0	28.57	2	542	38.71	14.30	4.63	111	68.38
	2	(1,2)	2,8	199,18	0,0	82.29	1 - 2	313	61.07	1200	3.04	47	86.61
	3	(2,3)	0,24	17,7	0,142	86.43	1 - 2	218	76.64	293.56	3.25	60	82.91
	4	(3,4)	1,26	2,4	0,11	96.86	2	132	86.07	339.88	1.82	53	84.90
	5	(3,5)	0,24	2,4	0,2	97.71	1 - 2	105	93.57	132.18	1.66	6	98.29
	6	(4,6)	0,11	2,15	0,0	97.86	1 - 1	28	95.57	222.75	1.36	1	99.72
	7	(6,6)	0,10	2,16	0,1	98.86	1	28	97.57	88.43	3.42	1	99.72
	8	(7,6)	0,0	0,0	0,0	100.0		34	100.0	26.21		1	99.72

Table F.4. Iteration details of HCB – Skin subset 2

			Training Phase									Test Phase	
			Model-MOU Results				Class of trimmed box	# of samples to be fixed		CPU times (sec.)			
	It.	Box alloc.	M	O	U	A_M			A_{TB}	Model MOU	HTA	M	A
Fold 1	1	(1,1)	151,5	28,0	0,20	85.21	1 - 2	1020	72.86	245.16	1.39	34	90.31
	2	(2,2)	0,2	2,0	2,31	97.36	2	188	86.29	28.22	1.40	6	98.29
	3	(2,3)	0,2	0,0	0,20	98.43	2 -2	155	97.36	120.40	1.43	4	98.86
	4	(2,5)	0,0	0,2	0,3	99.64	2 - 2	17	98.57	13.35	1.22	1	99.72
	5	(2,7)	0,0	0,0	0,0	100.0		20	100.0	30.11		1	99.72
Fold 2	1	(1,1)	125,7	44,0	0,27	86.43	1 - 2	1025	73.21	523.12	1.34	57	83.76
	2	(2,2)	0,1	0,0	11,20	97.71	2	200	87.50	67.04	1.83	11	96.87
	3	(2,3)	3,4	0,0	7,3	98.79	1 - 2	158	98.79	36.01	2.14	11	96.87
	4	(3,4)	0,0	0,0	0,5	99.64	2	5	99.14	24.15	0.99	12	96.58
	5	(3,5)	0,0	0,0	0,0	100.0		12	100.0	18.56		2	99.43
Fold 3	1	(1,1)	160,1	16,0	0,24	73.33	1 - 2	1013	72.36	426.15	0.17	32	90.86
	2	(2,2)	7,8	0,0	1,0	99.71	2 - 2	365	98.43	4.81	25.81	2	99.43
	3	(2,4)	0,0	0,0	0,0	100.0		22	100.0	18.30	18.12	0	100.0
Fold 4	1	(1,1)	145,6	18,0	0,23	86.36	1 - 2	1017	72.64	426.51	4.24	45	87.14
	2	(2,2)	0,5	8,0	0,17	97.86	2	65	77.29	82.16	2.13	9	97.43
	3	(2,3)	3,2	6,1	9,0	98.50	1	49	80.79	46.15	1.32	5	98.57
	4	(3,3)	3,0	3,1	7,0	99.00	1 - 2	185	94.00	19.20	1.99	4	98.86
	5	(4,4)	0,1	2,3	0,1	99.57	2 - 2	42	97.00	22.61	0.96	4	98.86
	6	(4,6)	0,0	0,0	0,0	100.0		42	100.0	19.07		0	100.0
Fold 5	1	(1,1)	145,5	18,0	11,19	85.93	1 - 2	1020	72.86	246.09	0.17	34	90.29
	2	(2,2)	6,1	8,0	5,2	98.43	1 - 2	354	98.14	43.12	1.40	9	97.43
	3	(3,3)	3,0	2,0	0,1	99.57	1 - 2	15	99.21	37.47	1.07	0	100.0
	4	(4,4)	0,0	0,0	0,0	100.0		11	100.0	12.00		0	100.0

Table F.5. Iteration details of HCB – Skin subset 3

			Training Phase									Test Phase	
			Model-MOU Results				Class of trimmed box	# of samples to be fixed	A_{TB}	CPU times (sec.)			
	It.	Box alloc.	M	O	U	A_M				Model MOU	HTA	M	A
Fold 1	1	(1,1)	800,0	0,0	0,0	71.43	2	424	15.14	6	10.65	563	19.57
	2	(1,2)	800,0	0,0	0,21	70.68	2	253	24.18	1200	9.55	527	24.71
	3	(1,3)	544,0	51,0	245,264	60.57	1-2	135	29.00	1200	5.39	487	30.43
	4	(2,4)	0,975	0,42	4,251	54.57	1	182	35.50	1200	17.22	452	35.43
	5	(3,4)	471,324	1,15	53,117	64.96	1-2	517	53.96	1200	6.31	215	69.29
	6	(4,5)	29,230	18,15	110,334	74.79	2	183	60.50	1200	4.55	189	73.00
	7	(4,6)	1,21	53,6	102,407	78.93	2	184	67.07	1200	1.81	102	85.43
	8	(4,7)	78,16	21,73	105,254	80.25	1-2	269	76.68	1200	2.13	72	89.71
	9	(5,8)	18,17	29,38	116,244	83.50	1-2	138	81.61	1200	3.57	72	89.71
	10	(6,9)	4,8	13,54	211,230	81.43	1-2	62	83.82	1200	1.45	72	89.71
	11	(7,10)	1,1	47,13	211,221	82.50	2	21	84.57	1200	1.53	56	92.00
	12	(7,11)	1,2	4,49	188,214	83.64	1-2	65	86.89	1200	3.73	30	95.71
	13	(8,12)	1,23	32,19	59,62	93.00	1-2	141	91.93	1200	2.66	17	97.57
	14	(8,13)	1,23	31,19	59,28	94.25	2	64	94.21	1200	2.4	17	97.57
	15	(9,13)	1,2	31,13	68,49	94.14	1	44	95.79	1200	4.09	14	98.00
	16	(10,13)	10,9	5,13	26,39	96.36	1	39	97.18	955	2.3	17	97.57
	17	(11,13)	1,6	0,13	13,26	97.89	1	25	98.07	1034	24.47	17	97.57
	18	(11,14)	1,4	0,0	13,18	98.71	2	30	99.14	709	3.79	17	97.57
	19	(12,15)	1,3	0,0	9,15	99.00	1-2	14	99.64	490	2.66	13	98.14
	20	(13,16)	0,0	0,0	0,0	100.0		10	100.0	232	3.65	6	99.14
Fold 2	1	(1,1)	800,0	0,0	0,0	71.43	2	410	14.64	8	24.27	507	27.57
	2	(1,2)	800,0	0,0	0,438	55.79	2	637	37.39	1200	12.31	518	26.00
	3	(1,3)	427,214	0,0	113,290	62.71	1-2	340	49.54	1200	7.60	511	27.00
	4	(2,4)	110,36	238,206	3,121	74.46	1-2	479	66.64	1200	3.07	311	55.57
	5	(3,5)	0,12	7,3	115,382	81.46	2	203	73.89	1200	2.68	175	75.00
	6	(3,6)	10,8	0,0	260,130	85.43	1	173	80.07	1200	4.16	162	76.86
	7	(4,6)	12,9	21,5	134,149	88.21	2	179	86.46	1200	6.12	106	84.86
	8	(4,7)	0,10	0,0	93,209	88.86	1	57	88.50	1200	1.05	106	84.86
	9	(5,7)	15,0	32,25	5,0	97.25	1-2	142	93.57	1200	3.96	63	91.00
	10	(6,8)	0,19	0,0	35,94	94.71	2	44	95.14	1200	2.51	47	93.29
	11	(6,9)	0,4	3,2	24,51	97.00	1-2	53	97.04	1200	1.53	19	97.29
	12	(7,10)	0,0	5,2	2,23	98.86	2	20	97.75	714	1.33	11	98.43
	13	(7,11)	0,0	1,2	16,10	98.96	1	12	98.18	384	1.29	17	97.57
	14	(8,11)	0,0	1,2	3,23	98.96	2	12	98.61	612	2.46	15	97.86
	15	(8,12)	0,0	1,1	13,3	99.36	1	5	98.79	514	2.23	15	97.86
	16	(9,12)	0,0	1,1	3,9	99.50	1-2	10	99.14	224	1.09	11	98.43
	17	(10,13)	0,0	1,0	2,6	99.68	1-2	15	99.68	332	2.03	11	98.43
	18	(11,14)	0,0	0,0	0,0	100.0		9	100.0	264		11	98.43

Table F.5 (continue)

			Training Phase									Test Phase	
			Model-MOU Results				Class of trimmed box	# of samples to be fixed		CPU times (sec.)			
	It.	Box alloc.	M	O	U	A_M			A_{TB}	Model MOU	HTA	M	A
Fold 3	1	(1,1)	800,0	0,0	0,0	71.43	2	506	18.07	9	26.12	530	24.28
	2	(1,2)	800,0	0,0	0,304	60.57	2	609	39.82	1200	9.55	560	20.00
	3	(1,3)	410,51	0,20	210,290	64.96	1-2	301	50.57	1200	12.13	550	21.43
	4	(2,4)	220,160	0,6	17,303	74.79	1-2	457	66.89	1200	5.45	215	69.29
	5	(3,5)	108,62	9,47	37,290	80.25	2	198	73.96	1200	7.16	102	85.43
	6	(3,6)	129,45	0,22	235,88	81.46	1	138	78.89	1200	2.35	106	84.86
	7	(4,6)	108,32	0,27	35,260	83.50	2	96	82.32	1200	3.85	72	89.71
	8	(4,7)	119,39	10,27	175,120	82.50	1	157	87.93	1200	1.56	63	91.00
	9	(5,7)	16,27	4,18	82,145	89.71	1-2	127	92.46	1200	3.22	49	93.00
	10	(6,8)	19,23	6,17	45,86	93.00	2	39	93.86	1200	1.08	30	95.71
	11	(6,9)	17,5	5,18	32,87	94.14	1-2	47	95.54	1200	2.75	17	97.57
	12	(7,10)	8,12	0,6	38,76	95.71	1-2	32	96.68	1200	2.04	17	97.57
	13	(8,11)	7,15	0,3	62,15	96.36	1	28	97.68	1200	1.65	17	97.57
	14	(9,11)	0,2	1,4	5,17	98.96	2	5	97.86	815	1.31	15	97.86
	15	(9,12)	3,1	2,2	27,24	97.89	1	14	98.36	642	1.45	33	95.29
	16	(10,12)	0,0	2,5	4,9	99.50	1-2	24	99.21	564	1.86	8	98.86
	17	(11,13)	0,0	0,0	3,6	99.68	1-2	18	99.86	648	1.03	33	95.29
	18	(12,14)	0,0	0,0	0,0	100.0		4	100.0	325		33	95.29
Fold 4	1	(1,1)	800,0	0,0	0,0	71.43	2	462	16.50	9	20.12	540	22.86
	2	(1,2)	800,0	0,0	0,16	70.86	2	248	25.36	1200	14.52	558	20.29
	3	(1,3)	544,0	55,0	245,200	62.71	1-2	156	30.93	1200	8.35	500	28.57
	4	(2,4)	227,82	54,42	325,251	64.96	1	182	37.43	1200	6.82	452	35.43
	5	(3,4)	368,221	10,17	47,52	74.46	1-2	407	51.96	1200	7.23	335	52.14
	6	(4,5)	47,286	18,15	62,125	80.25	2	183	58.50	1200	3.56	340	51.43
	7	(4,6)	1,21	53,21	102,321	81.46	2	192	65.36	1200	2.89	235	66.43
	8	(4,7)	78,21	27,73	136,123	83.64	1-2	263	74.75	1200	3.56	190	72.85
	9	(5,8)	56,19	18,85	92,42	88.86	1-2	126	79.25	1200	2.19	139	80.15
	10	(6,9)	26,15	12,23	126,198	85.71	1-2	80	82.11	1200	3.48	114	83.71
	11	(7,10)	1,1	47,13	211,221	86.14	2	21	82.86	1200	2.13	117	83.24
	12	(7,11)	2,6	8,9	156,110	89.61	1-2	69	85.32	1200	3.5	98	86.00
	13	(8,12)	5,2	8,12	123,36	93.36	1-2	141	90.36	1200	1.16	86	87.71
	14	(9,13)	1,2	6,1	108,23	94.96	1-2	96	93.79	1200	1.65	63	91.00
	15	(10,14)	1,2	5,12	54,22	96.46	1	44	95.36	1200	1.23	55	92.14
	16	(11,14)	2,1	0,7	39,29	97.21	1	39	96.75	867	2.56	38	94.54
	17	(12,14)	1,1	2,3	21,4	98.86	1	37	98.07	495	2.08	21	97.00
	18	(13,14)	0,1	2,1	12,23	98.61	2	30	99.14	651	1.89	11	98.43
	19	(13,15)	0,0	2,1	7,3	99.64	1-2	14	99.64	563	0.59	4	99.43
	20	(14,16)	0,0	0,0	0,0	100.0		10	100.0	321		11	98.43

Table F.5 (continue)

			Training Phase									Test Phase	
			Model-MOU Results				Class of trimmed box	# of samples to be fixed		CPU times (sec.)			
	It.	Box alloc.	M	O	U	A_M			A_{TB}	Model MOU	HTA	M	A
Fold 5	1	(1,1)	800,0	0,0	0,0	71.43	2	512	18.29	12	16.74	500	28.57
	2	(1,2)	800,0	0,0	82,100	64.86	2	248	27.14	1200	10.61	540	22.86
	3	(1,3)	216,328	28,12	256,58	67.93	1-2	164	33.00	1200	7.75	528	24.57
	4	(2,4)	84,120	26,0	237,235	74.92	1	168	39.00	1200	3.43	474	32.29
	5	(3,4)	177,63	34,22	192,156	77.00	1-2	407	53.54	1200	1.86	292	58.29
	6	(4,5)	168,121	10,44	67,86	82.29	2	183	60.07	1200	2.36	232	66.86
	7	(4,6)	47,86	18,15	62,117	87.68	2	105	63.82	1200	8.78	175	75.00
	8	(4,7)	1,16	40,21	102,220	85.71	1-2	263	73.21	1200	3.55	116	83.50
	9	(5,8)	18,21	27,60	68,87	89.95	1-2	141	78.25	1200	2.16	78	88.86
	10	(6,9)	46,19	18,85	102,42	88.86	1-2	101	81.86	1200	2.35	102	85.43
	11	(7,10)	16,8	12,23	75,157	89.61	2	21	82.61	1200	2.26	72	89.71
	12	(7,11)	10,12	11,16	54,138	91.39	1-2	76	85.32	1200	1.96	72	89.71
	13	(8,12)	2,6	4,11	98,67	93.29	1-2	150	90.68	1200	1.26	73	89.61
	14	(9,13)	5,2	8,12	103,35	94.11	1-2	97	94.14	1200	1.70	63	91.00
	15	(10,14)	3,6	15,1	79,8	96.00	1-2	56	96.14	1200	1.26	47	93.29
	16	(11,15)	3,6	5,16	54,22	96.21	1	26	97.07	1200	2.05	30	95.71
	17	(12,15)	2,1	0,7	39,29	98.43	1-2	44	98.64	1200	2.03	17	97.57
	18	(13,16)	1,2	2,3	20,4	98.86	1	17	99.25	913	1.23	9	98.71
	19	(14,16)	2,1	2,6	12,7	98.92	2	5	99.43	768	1.08	9	98.71
	20	(14,17)	1,1	3,7	13,3	99.00	1	6	99.64	812	1.09	9	98.71
	21	(15,17)	1,1	1,2	5,12	99.21	1-2	7	99.89	562	2.06	9	98.71
	22	(16,18)	0,0	0,0	0,0	100.0		3	100.0	298		9	98.71

Table F.6. Iteration details of HCB – Skin subset 4

			Training Phase									Test Phase	
			Model-MOU Results				Class of trimmed box	# of samples to be fixed		CPU times (sec.)			
	It.	Box alloc.	M	O	U	A_M			A_{TB}	Model MOU	HTA	M	A
Fold 1	1	(1,1)	800,0	0,0	0,0	71.43	2	1620	57.86	1200.0	7.664	102	85.43
	2	(1,2)	1,2	1,30	9,15	98.04	1 - 2	1014	94.07	1200.0	0.586	26	96.29
	3	(2,3)	0,0	0,0	21,0	99.25	1	100	97.64	1200.0	1.143	1	99.86
	4	(3,3)	0,0	0,0	0,0	100.0		66	100.0	11.4		0	100.0
Fold 2	1	(1,1)	800,0	0,0	0,0	71.43	2	1618	57.79	1200.0	15.214	84	88.00
	2	(1,2)	2,2	6,28	4,2	98.57	2	337	69.82	248.4	1.062	31	95.57
	3	(1,3)	0,16	13,0	6,0	98.75	1	662	93.46	235.8	2.44	4	99.43
	4	(2,3)	0,0	0,0	0,0	100.0		183	100.0	21.5		1	99.86
Fold 3	1	(1,1)	6,38	18,1	104,0	85.29	2	1616	57.71	1200.0	2.105	73	89.57
	2	(1,2)	2,29	7,5	0,0	98.46	1 - 2	795	86.11	1200.0	1.183	15	97.86
	3	(2,3)	0,5	3,4	6,0	99.36	1 - 1	365	99.14	863.2	1.953	0	100.00
	4	(4,3)	0,0	0,0	0,0	100.0		24	100.0	13.9		0	100.0
Fold 4	1	(1,1)	0,44	0,0	0,349	85.96	2	1607	57.39	1200.0	3.852	56	92.00
	2	(1,2)	6,30	0,0	0,0	98.36	1	349	69.86	1200.0	0.613	28	96.00
	3	(1,3)	0,18	0,13	0,5	95.82	1 - 2	729	95.89	163.1	1.957	12	98.29
	8	(2,3)	0,0,	0,0	0,0	100.0		115	100.0	14.8		0	100.0
Fold 5	1	(1,1)	1,1	0,40	0,346	86.57	2	1615	57.68	1200.0	3.848	210	70.00
	2	(1,2)	5,30	1,0	0,0	98.71	2	346	70.04	1200.0	0.582	25	96.43
	3	(1,3)	13,0	10,0	6,0	98.96	1 - 2	702	95.11	1200.0	0.859	1	99.86
	4	(3,3)	0,0	0	0,0	100.0		137	100.0	15.7		0	100.0

Table F.7. Iteration details of HCB – Skin subset 5

			Training Phase									Test Phase	
			Model-MOU Results				Class of trimmed box	# of samples to be fixed	A_{TB}	CPU times (sec.)		M	A
	It.	Box alloc.	M	O	U	A_M				Model MOU	HTA		
Fold 1	1	(1,1)	1600,0	0,0	0,0	71.43	2	3212	57.36	11.481	0.171	1484	73.50
	2	(1,2)	1,41	65,312	23,53	91.16	1 - 2	1727	88.20	6.747	0.047	156	97.21
	3	(2,3)	2,3	179,0	2,1	95.71	1	343	94.32	5.885	0.102	144	97.43
	4	(3,3)	0,0	20,1	0,41	98.89	2	176	97.46	6.043	0.149	76	98.64
	5	(3,4)	0,1	20,8	3,30	98.89	2	71	98.73	5.524	1.719	24	99.57
	6	(3,5)	0,2	5,10	3,2	99.43	1	41	99.61	5.12	1.813	156	97.21
	7	(3,6)	0,0	0,0	0,0	100.0		20	100.0	21.307		0	100.0
Fold 2	1	(1,1)	800,0	0,200	0,0	82.14	2	3214	57.39	13.1	14.56	246	82.43
	2	(1,2)	2,38	162,298	75,45	88.93	1	1129	77.55	938.3	9.25	26	98.14
	3	(2,2)	160,760	25,15	235,205	75.00	1-2	786	91.59	826.3	7.46	19	98.64
	4	(3,3)	4,2	80,25	2,1	97.96	2	306	97.05	765.2	3.21	12	99.14
	5	(3,4)	0,1	12,6	20,12	99.09	1-2	77	98.43	265.2	1.02	14	99.00
	6	(4,5)	1,1	4,3	2,1	99.79	2	76	99.79	235.5	2.30	3	99.79
	7	(4,6)	0,0	0,0	0,0	100.0		12	100	23.3		3	99.79
Fold 3	1	(1,1)	1600,0	0,0	0,0	71.43	2	3229	57.66	15.6	16.15	202	85.57
	2	(1,2)	2,38	57,315	36,34	91.39	1-2	1628	86.73	846.1	6.12	91	93.50
	3	(2,3)	2,3	168,25	15,2	96.16	1	480	95.3	658.2	2.35	28	98.00
	4	(3,3)	1,1	6,7	25,8	99.14	1-2	215	99.14	356.2	1.05	10	99.29
	5	(4,4)	0,0	1,0	0,0	99.98	2	47	99.98	265.5	1.56	1	99.93
	6	(4,5)	0,0	0,0	0,0	100.0		1	100	56.2		1	99.93
Fold 4	1	(1,1)	1600,0	0,0	0,0	71.43	2	3213	57.38	26.6	26.10	423	69.79
	2	(1,2)	2,15	95,125	65,21	94.23	2	684	69.59	985.2	5.46	64	95.43
	3	(2,3)	2,3	156,65	17,41	94.93	1-2	1007	87.57	765.2	6.18	122	91.29
	4	(3,3)	2,1	23,15	22,35	98.25	1	550	97.39	468.5	2.66	18	98.71
	5	(4,3)	0,1	3,4	15,7	99.46	1-2	111	99.38	562.2	2.01	5	99.64
	6	(5,4)	0,0	0,0	0,0	100.0		35	100	120.1		3	99.79
Fold 5	1	(1,1)	1600,0	0,0	0,0	71.43	2	3213	57.38	19.8	19.20	215	84.64
	2	(1,2)	35,68	226,103	26,85	90.30	1	981	74.89	879.5	4.68	50	96.43
	3	(2,2)	2,18	61,301	35,9	92.39	2	632	86.18	798.6	9.15	31	97.79
	4	(2,3)	4,2	55,26	23,25	97.59	1-2	628	97.39	816.0	1.05	3	99.79
	5	(3,4)	1,2	6,9	15,5	99.32	1-2	112	99.39	552.2	2.06	2	99.86
	6	(4,5)	0,0	0,0	0,0	100.0		34	100	68.5		1	99.93

G. Purity and Power scores of Hyperboxes generated by HCB

Table G.1. Purity and power scores of hyperboxes generated by HCB – Simulated 4 dataset

			Training Phase								Test Phase				
			Model MOU					After HTA			After HTA				
	Iter.	Box	N_{kl}	M_{kl}	Purity	Power	A_M	N_{kl}	Power	A_{TB}	N_{kl}	M_{kl}	Purity	Power	A
Fold 2	1	1	18	1	0.94	0.34	71.0	13	0.25	13.0	5	1	0.80	0.45	76.0
	2	2	46	15	0.67	0.98	76.0	10	0.21	23.0	5	1	0.80	0.36	72.0
	3	3	12	1	0.92	0.23	76.0	10	0.19	33.0	2	0	1.00	0.18	76.0
	4	4	37	15	0.59	0.79	81.0	9	0.19	42.0	3	0	1.00	0.21	72.0
	5	5	12	1	0.92	0.23	82.0	8	0.15	50.0	2	0	1.00	0.18	72.0
	6	8	19	4	0.79	0.4	84.0	8	0.17	58.0	3	1	0.67	0.27	80.0
	7	7	8	0	1.00	0.17	89.0	8	0.17	66.0					84.0
	8	6	5	0	1.00	0.09	89.0	5	0.09	71.0	3	0	1.00	0.07	84.0
	9	10	14	0	1.00	0.26	94.0	14	0.26	85.0					96.0
	10	11	2	0	1.00	0.04	98.0	2	0.04	87.0					88.0
	11	12	1	0	1.00	0.02		1	0.02						
	11	9	12	0	1.00	0.26	100	12	0.26	100	1	0	1.00	0.07	96.0
Fold 3	1	1	21	0	1.00	0.42	68.0	21	0.42	21.0	5	1	0.80	0.38	80.0
	2	2	45	17	0.62	0.90		22	0.44	43.0	4	0	1.00	0.33	
	2	3	13	3	0.77	0.26	79.0	10	0.20	53.0	3	1	0.67	0.23	80.0
	3	4	15	5	0.67	0.30	82.0	9	0.18	62.0	2	0	1.00	0.17	72.0
	4	5	16	4	0.75	0.32		9	0.18	71.0	1	0	1.00	0.08	
	4	6	10	1	0.90	0.20	88.0	9	0.18	80.0	1	0	1.00	0.08	76.0
	5	8	6	2	0.67	0.12	91.0	5	0.10	85.0	2	0	1.00	0.15	84.0
	6	9	5	0	1.00	0.10	95.0	5	0.10	90.0	1	0	1.00	0.08	88.0
	7	7	5	0	1.00	0.10	97.0	5	0.10	95.0					88.0
	8	10	2	0	1.00	0.04	99.0	2	0.04	97.0					88.0
	9	11	2	0	1.00	0.04		2	0.04						
	9	12	1	0	1.00	0.02	100	1	0.02	100					88.0
Fold 4	1	1	23	0	1	0.46	73.0	23	0.46	23.0	3	0	1.00	0.25	64.0
	2	2	47	14	0.70	0.94		12	0.24		2	0	1.00	0.15	
	2	3	11	3	0.73	0.22	81.0	9	0.18	44.0	2	0	1.00	0.17	76.0
	3	4	5	1	0.80	0.1		4	0.08		2	4	N/A	0.17	
	3	5	35	14	0.60	0.7	82.0	11	0.22	59.0	2	0	1.00	0.23	64.0
	4	7	11	0	1.00	0.22	79.0	11	0.22	70.0	3	0	1.00	0.25	72.0
	5	6	12	9	0.25	0.24		8	0.16		1	0	1.00	0.23	
	5	8	10	1	0.90	0.2	87.0	9	0.18	87.0	2	1	0.50	0.15	68.0
	6	9	4	0	1.00	0.08	94.0	4	0.08	91.0	1	0	1.00	0.08	72.0
	7	10	3	0	1.00	0.06	96.0	3	0.06	94.0					80.0
	8	11	2	0	1.00	0.04	98.0	2	0.04	96.0					80.0
	9	12	1	0	1.00	0.02	99.0	1	0.02	97.0					80.0
	10	13	1	0	1.00	0.02		1	0.02						
	10	14	2	0	1.00	0.04	100	2	0.04	100					80.0

Table G.1 (continue)

			Training Phase								Test Phase				
			Model MOU					After HTA			After HTA				
	Iter.	Box	N_{kl}	M_{kl}	Purity	Power	A_M	N_{kl}	Power	A_{TB}	N_{kl}	M_{kl}	Purity	Power	A
Fold 5	1	1	18	1	0.94	0.34	71.0	13	0.25	13.0	5	1	0.80	0.45	76.0
	2	2	46	15	0.67	0.98	76.0	10	0.21	23.0	5	1	0.80	0.36	72.0
	3	3	12	1	0.92	0.23	76.0	10	0.19	33.0	2	0	1.00	0.18	76.0
	4	4	37	15	0.59	0.79	81.0	9	0.19	42.0	3	0	1.00	0.21	72.0
	5	5	12	1	0.92	0.23	82.0	8	0.15	50.0	2	0	1.00	0.18	72.0
	6	8	19	4	0.79	0.4	84.0	8	0.17	58.0	3	1	0.67	0.27	80.0
	7	7	8	0	1.00	0.17	89.0	8	0.17	66.0					84.0
	8	6	5	0	1.00	0.09	89.0	5	0.09	71.0	3	0	1.00	0.07	84.0
	9	10	14	0	1.00	0.26	94.0	14	0.26	85.0					96.0
	10	11	2	0	1.00	0.04	98.0	2	0.04	87.0					88.0
	11	12	1	0	1.00	0.02		1	0.02						
	11	9	12	0	1.00	0.26	100	12	0.26	100	1	0	1.00	0.07	96.0

Table G.2. Purity and power scores of hyperboxes generated by HCB – Breast Cancer dataset

			Training Phase								Test Phase				
			Model MOU					After HTA			After HTA				
	Iter.	Box	N_{kl}	M_{kl}	Purity	Power	A_M	N_{kl}	Power	A_{TB}	N_{kl}	M_{kl}	Purity	Power	A
Fold 1	1	1	340	4	0.988	0.960	92.5	224	0.633	61.0	55	0	1.000	0.640	93.4
	2	2	116	2	0.983	0.604		109	0.568		34	1	0.971	0.667	94.9
	2	3	159	14	0.912	0.449	89.9	105	0.297	84.1	23	0	1.000	0.267	94.9
	3	5	21	0	1.000	0.109	85.9	21	0.109	86.3	5	1	0.800	0.098	93.4
	4	4	12	0	1.000	0.034		12	0.034		2	0	1.000	0.023	95.6
	4	6	21	4	0.810	0.109	89.4	15	0.078	89.9	3	1	0.667	0.059	95.6
	5	7	6	1	0.833	0.017		5	0.014		1	0	1.000	0.012	95.6
	5	8	12	5	0.583	0.063	91.4	8	0.042	92.1	2	1	0.500	0.039	95.6
	6	9	4	0	1.000	0.011		4	0.011						
	6	10	12	1	0.917	0.063	93.6	11	0.057	95.8	2	0	1.000	0.039	95.6
	7	12	9	0	1.000	0.047	95.8	9	0.047	96.2	1	2	-1.000	0.020	95.6
	8	11	2	0	1.000	0.006		2	0.006						
	8	13	4	0	1.000	0.021	96.9	4	0.021	97.6	1	0	1.000	0.020	96.4
	9	15	5	1	0.800	0.026	98.0	4	0.021	98.2	1	0	1.000	0.020	95.6
	10	16	3	0	1.000	0.016	98.5	3	0.016	0.0					
	11	17	3	0	1.000	0.016	99.1	3	0.016	99.1	1	0	1.000	0.020	95.6
	12	14	2	0	1.000	0.006		2	0.006						
	12	18	2	0	1.000	0.010	99.5	2	0.010	99.6					95.6
	13	19	1	0	1.000	0.003		1	0.003						
	13	20	2	0	1.000	0.010	100	2	0.010	100					95.6
Fold 2	1	1	340	4	0.988	0.960	94.9	326	0.921	59.7	78	0	1.000	9.750	93.3
	2	3	10	0	1.000	0.028		10	0.028		3	0	1.000	0.429	
	2	2	177	15	0.915	0.922	94.0	86	0.448	77.3	21	1	0.952	2.625	92.6
	3	5	35	3	0.914	0.182	84.8	30	0.156	82.8	3	0	1.000	0.375	94.8
	4	6	21	4	0.810	0.109	87.7	16	0.083	85.7	1	0	1.000	0.125	94.8
	5	7	13	1	0.923	0.068	89.6	11	0.057	87.7	0	1	N/A	0.000	95.6
	6	8	16	1	0.938	0.083	91.4	13	0.068	90.1					95.6
	7	9	7	2	0.714	0.036	92.3	5	0.026	91.0					94.8
	8	10	6	0	1.000	0.031	93.2	6	0.031	92.1	1	1	0.000	0.125	94.8
	9	11	10	1	0.900	0.052	95.1	10	0.052	94.0					94.8
	10	4	10	2	0.800	0.028		6	0.017	95.1					94.8
	10	12	4	0	1.000	0.021	95.1	4	0.021	95.8					94.8
	11	14	3	0	1.000	0.016		3	0.016	96.3	5	0	1.000	0.625	94.8
	12	15	3	0	1.000	0.016	97.4	3	0.016	96.9					94.8
	13	16	4	0	1.000	0.021	97.4	4	0.021	97.6					94.8
	14	13	3	0	1.000	0.008	98.0	3	0.008	98.2	1	0	1.000	0.125	94.8
	15	18	4	0	1.000	0.011	99.1	4	0.011	98.9					94.8
	16	17	2	0	1.000	0.010	99.6	2	0.010						94.8
	16	19	4	0	1.000	0.011	100	4	0.011	100					94.8

Table G.2 (continue)

			Training Phase								Test Phase				
			Model MOU					After HTA			After HTA				
	Iter.	Box	N_{kl}	M_{kl}	Purity	Power	A_M	N_{kl}	Power	A_{TB}	N_{kl}	M_{kl}	Purity	Power	A
Fold 3	1	1	332	12	0.964	0.949		331	0.946		64	1	0.984	8.000	
	1	2	178	18	0.899	0.908	93.4	145	0.740	87.2	34	0	1.000	4.857	92.8
	2	4	17	0	1.000	0.087	90.5	17	0.087	90.3	6	0	1.000	0.750	94.2
	3	5	21	0	1.000	0.107	94.9	21	0.107	94.1	11	0	1.000	1.375	92.8
	4	3	11	1	0.909	0.031	96.5	9	0.026	95.8	3	0	1.000	0.375	92.4
	5	6	10	1	0.900	0.051	97.1	8	0.041	97.3					
	6	7	5	1	0.800	0.014		4	0.011						93.5
	6	8	6	3	0.500	0.031	97.8	2	0.010	98.4	1	1	0.000	0.125	93.5
	7	9	1	0	1.000	0.003		1	0.003						
	7	10	3	0	1.000	0.015	98.5	3	0.015	99.1					93.5
	8	11	2	0	1.000	0.006		2	0.006	99.5					
	8	12	2	0	1.000	0.006	99.6	2	0.006	99.8					93.5
	9	13	2	0	1.000	0.006	100	2	0.006	100					93.5
Fold 4	1	1	342	9	0.974	0.961	94	338	0.949	61.9	59	1	0.983	7.375	93.5
	2	2	174	18	0.897	0.916		80	0.421		15	0	1.000	2.143	
	2	3	5	0	1.000	0.014	94.3	5	0.014	77.5					93.5
	3	4	7	0	1.000	0.020		7	0.020		1	0	1.000	0.125	
	3	5	94	18	0.809	0.495	94.7	28	0.147	83.9	12	1	0.917	1.500	94.2
	4	7	65	18	0.723	0.342	94.9	21	0.111	87.7	14	0	1.000	1.750	95.7
	5	6	3	0	1.000	0.008		3	0.008						
	5	8	43	17	0.605	0.226	94.9	10	0.053	90.1	5	1	0.800	0.625	94.2
	6	10	30	10	0.667	0.158	94.9	12	0.063	92.3	3	0	1.000	0.375	94.9
	7	11	20	8	0.600	0.105	95.2	12	0.063	94.5	2	0	1.000	0.250	95.7
	8	12	21	1	0.952	0.111	96.5	11	0.058	96.5	1	0	1.000	0.125	95.7
	9	13	10	3	0.700	0.053	98.2	6	0.032	97.6	1	1	0.000	0.125	95.7
	10	14	7	2	0.714	0.037	99.1	5	0.026	98.5					95.7
	11	9	3	0	1.000	0.008		3	0.008						
	11	15	5	0	1.000	0.026	100	5	0.026	100					95.7
Fold 5	1	1	321	7	0.978	0.902	93.8	296	0.831	54.2	49	0	1.000	6.125	95.6
	2	3	64	5	0.922	0.180		28	0.079		6	0	1.000	0.857	
	2	2	83	9	0.892	0.437	93.4	74	0.389	72.9	23	1	0.957	2.875	95.6
	3	4	28	6	0.786	0.079	95.4	18	0.051	76.2	3	0	1.000	0.375	94.9
	4	5	54	3	0.944	0.284	95.6	46	0.242	84.6	14	1	0.929	1.750	95.6
	5	6	10	1	0.900	0.028		8	0.022						
	5	7	35	2	0.943	0.184	97.1	31	0.163	91.8	7	0	0.000	0.875	94.9
	6	9	18	2	0.889	0.095	96.3	14	0.074	94.3	1	1	0.000	0.125	94.2
	7	10	8	2	0.750	0.042	97.3	6	0.032	95.4					94.9
	8	11	3	1	0.667	0.016	97.8	3	0.016	96.0					94.9
	9	8	4	0	1.000	0.011		4	0.011	96.7					
	9	12	5	1	0.800	0.026	98.2	4	0.021	97.4					94.9
	10	14	6	1	0.833	0.032	99.1	5	0.026	98.4	1	1	0.000	0.125	94.9
	11	15	3	0	1.000	0.016	99.5	3	0.016	98.9					94.9
	12	16	4	0	1.000	0.021	99.8	4	0.021	99.6					94.9
	13	17	2	0	1.000	0.006	100	2	0.006	100					94.9

Table G.3. Purity and power scores of hyperboxes generated by HCB – Skin subset 1

			Training Phase								Test Phase				
			Model MOU					After HTA			After HTA				
	Iter.	Box	N_{kl}	M_{kl}	Purity	Power	A_M	N_{kl}	Power	A_{TB}	N_{kl}	M_{kl}	Purity	Power	A
Fold 1	1	2	1000	400	0.600	1.000	71.4	547	0.547	39.1	143	1	0.9930	0.572	65.4
	2	3	236	10	0.958	0.236	83.5	214	0.214	54.4	46	0	1.0000	0.184	88.9
	3	1	392	25	0.936	0.980		183	0.458		39	2	0.9487	0.390	
	3	4	138	0	1.000	0.138	92.2	138	0.138	77.3	38	0	1.0000	0.152	92.3
	4	5	160	6	0.963	0.400		118	0.295		32	0	1.0000	0.320	
	4	6	75	0	1.000	0.188	94.0	75	0.188	91.1	13	0	1.0000	0.130	96.6
	5	7	39	19	0.513	0.039	94.6	21	0.021	92.6	2	0	1.0000	0.008	96.9
	6	9	26	1	0.962	0.065	95.1	21	0.053	94.1	3	0	1.0000	0.030	96.6
	7	10	23	2	0.913	0.058	96.4	13	0.033	95.0	4	0	1.0000	0.040	98.0
	8	11	19	0	1.000	0.048	97.0	19	0.048	96.4	55	0	1.0000	0.550	98.6
	9	8	10	0	1.000	0.010		10	0.010						
	9	12	10	1	0.900	0.025	97.7	10	0.025	97.8	4	0	1.0000	0.040	98.9
	10	13	8	0	1.000	0.020		8	0.020		2	0	1.0000	0.020	
	10	14	8	1	0.875	0.008	98.5	8	0.008	98.9	1	0	1.0000	0.004	98.6
	11	15	8	0	1.000	0.020		8	0.020		2	0	1.0000	0.020	
	11	16	3	0	1.000	0.003	99.7	3	0.003	99.7					98.9
	12	17	2	0	1.000	0.002		2	0.002						
	12	18	2	0	1.000	0.002	100	2	0.002	100					98.9
Fold 2	1	2	1000	400	0.600	1.000	71.4	535	0.535	38.2	137	0	1.0000	0.548	76.3
	2	1	354	8	0.977	0.885	82.4	172	0.430	50.5	19	0	1.0000	0.190	80.9
	3	3	321	211	0.343	0.321	83.5	212	0.212	65.6	35	0	1.0000	0.140	86.6
	4	4	142	0	1.000	0.355	87.6	142	0.355	75.8	34	0	1.0000	0.340	94.9
	5	5	150	0	1.000	0.150	92.6	150	0.150	86.5					
	6	7	74	0	1.000	0.074	97.9	74	0.074	91.8	19	0	1.0000	0.076	93.4
	7	6	87	2	0.977	0.218	98.4	67	0.168	96.6	12	0	1.0000	0.120	97.4
	8	8	10	0	1.000	0.010		10	0.010		3	0	1.0000	0.012	
	8	9	20	2	0.900	0.050	98.6	19	0.048	98.6	15	0	1.0000	0.150	98.6
	9	10	3	0	1.000	0.008		3	0.008		1	0	1.0000	0.010	
	9	11	7	0	1.000	0.007	99.1	7	0.007	99.4	2	0	1.0000	0.008	100
	10	12	7	0	1.000	0.007		7	0.007		1	0	1.0000	0.004	
	10	13	2	0	1.000	0.002	100	2	0.002	100	0	1	N/A	0.000	99.7
Fold 3	1	2	1000	400	0.600	1.000	71.4	541	0.541	38.6	46	2	0.9565	0.184	82.6
	2	1	333	17	0.949	0.833		239	0.598		140	0	1.0000	1.400	
	2	3	491	215	0.562	0.491	82.1	214	0.214	71	38	1	0.9737	0.152	93.1
	3	5	201	52	0.741	0.201	94.0	131	0.131	80.4	16	1	0.9375	0.064	90.3
	4	4	86	0	1.000	0.215	97.4	86	0.215	86.5	33	0	1.0000	0.330	95.1
	5	6	77	0	1.000	0.077	82.1	77	0.077	92.0	18	0	1.0000	0.072	98.0
	6	7	85	10	0.882	0.213		75	0.188		32	0	1.0000	0.320	
	6	8	15	0	1.000	0.015	99.1	15	0.015	98.4	4	0	1.0000	0.016	97.7
	7	9	7	0	1.000	0.018		7	0.018						
	7	10	5	0	1.000	0.005	99.3	5	0.005	99.3	2	0	1.0000	0.008	98.9
	8	11	5	0	1.000	0.013		5	0.013						
	8	12	5	0	1.000	0.005	100	5	0.005	100	8	0	1.0000	0.008	98.9

Table G.3 (Continue)

			Training Phase								Test Phase				
			Model MOU					After HTA			After HTA				
	Iter.	Box	N_{kl}	M_{kl}	Purity	Power	A_M	N_{kl}	Power	A_{TB}	N_{kl}	M_{kl}	Purity	Power	A
Fold 4	1	2	1000	400	0.600	1.000	71.4	542	0.542	39.8	35	0	1.0000	0.140	71.4
	2	1	347	22	0.937	0.868	82.3	104	0.260	60.4	138	0	1.0000	1.380	85.4
	3	3	342	210	0.386	0.342		209	0.209	74.3	43	0	1.0000	0.172	
	3	4	91	3	0.967	0.228	83.2	140	0.350	80.6	13	0	1.0000	0.130	91.1
	4	6	140	0	1.000	0.140	92.9	78	0.078	91.2	15	0	1.0000	0.060	93.1
	5	7	70	0	1.000	0.070	97.9	132	0.132	96.2	47	0	1.0000	0.188	98.6
	6	5	24	0	1.000	0.060		80	0.200	97.9	7	0	1.0000	0.070	
	6	8	13	0	1.000	0.013	98.9	25	0.025	98.9					99.4
	7	9	10	1	0.900	0.010		14	0.035	99.5	8	0	1.0000	0.080	
	7	10	5	0	1.000	0.005	99.6	14	0.014	99.9	1	0	1.0000	0.004	99.4
	8	11	2	0	1.000	0.002	100	28	0.070	100	17	0	1.0000	0.170	99.4
Fold 5	1	2	1000	400	0.600	1.000	28.6	18	0.045	38.7	17	0	1.0000	0.170	68.4
	2	1	389	26	0.933	0.973		16	0.040	46.1	23	0	1.0000	0.230	
	2	3	332	201	0.395	0.332	82.3	542	0.542	61.1	35	0	1.0000	0.140	86.6
	3	4	299	31	0.896	0.748		104	0.260	71.1	138	0	1.0000	1.380	
	3	5	83	17	0.795	0.083	86.4	209	0.209	76.6	43	0	1.0000	0.172	82.9
	4	7	134	1	0.993	0.134	96.9	140	0.350	86.1	13	0	1.0000	0.130	84.9
	5	6	182	28	0.846	0.455		78	0.078	91.8	15	0	1.0000	0.060	
	5	8	25	0	1.000	0.025	97.7	132	0.132	93.6	47	0	1.0000	0.188	98.3
	6	9	90	26	0.711	0.225		80	0.200	94.6	7	0	1.0000	0.070	
	6	10	14	0	1.000	0.014	97.9	25	0.025	95.6					99.7
	7	11	29	0	1.000	0.073	98.9	14	0.035	97.6	8	0	1.0000	0.080	99.7
	8	12	18	0	1.000	0.045		14	0.014	98.9	1	0	1.0000	0.004	
	8	13	16	0	1.000	0.040	100	28	0.070	100	17	0	1.0000	0.170	99.7

Table G.4. Purity and power scores of hyperboxes generated by HCB – Skin subset 2

			Training Phase								Test Phase				
			Model MOU					After HTA			After HTA				
	Iter.	Box	N_{kl}	M_{kl}	Purity	Power	A_M	N_{kl}	Power	A_{TB}	N_{kl}	M_{kl}	Purity	Power	A
Fold 1	1	1	249	8	0.968	0.623		245	0.613		68	1	0.985	0.680	
	1	2	944	179	0.810	0.944	85.2	775	0.775	72.9	193	0	1.000	0.769	90.3
	2	4	190	2	0.989	0.190	97.4	188	0.188	86.3	30	0	1.000	0.120	98.3
	3	3	155	0	1.000	0.388		155	0.388		49	0	1.000	0.490	
	3	5	17	0	1.000	0.017	98.4	17	0.017	98.6	1	0	1.000	0.004	98.9
	4	6	13	0	1.000	0.013		13	0.013		1	0	1.000	0.004	
	4	7	4	0	1.000	0.004	99.6	4	0.004	99.8					99.7
	5	8	1	0	1.000	0.001		1	0.001	99.9					
	5	9	2	0	1.000	0.002	100	2	0.002	100	1	0	1.000	0.004	99.7
Fold 2	1	1	282	7	0.975	0.705		246	0.615		52	0	1.000	0.520	
	1	2	935	166	0.822	0.935	86.4	779	0.779	73.2	167	1	0.994	0.665	83.8
	2	4	200	0	1.000	0.200	97.7	200	0.200	87.5	72	0	1.000	0.287	96.9
	3	5	19	4	0.789	0.019		15	0.015	88.6	10	0	1.000	0.040	
	3	6	151	3	0.980	0.378	98.8	143	0.358	98.8	44	1	0.977	0.440	96.9
	4	3	11	0	1.000	0.028		11	0.028	99.6	3	0	1.000	0.030	
	4	7	1	0	1.000	0.001	99.6	1	0.001	99.6					97.7
	5	8	5	0	1.000	0.005	100	5	0.005	100					99.4
Fold 3	1	1	241	1	0.996	0.603		240	0.600		63	0	1.000	0.630	
	1	2	959	176	0.816	0.959	85.6	773	0.773	72.4	165	0	1.000	0.657	90.9
	2	3	160	0	1.000	0.400		160	0.400	83.8	35	0	1.000	0.350	
	2	4	205	1	0.995	0.205	97.7	205	0.205	98.4	70	0	1.000	0.279	99.4
	3	5	9	0	1.000	0.009		9	0.009	99.1	10	0	1.000	0.040	
	3	6	13	0	1.000	0.013	100	13	0.013	100	2	0	1.000	0.008	100
Fold 4	1	1	254	7	0.972	0.635		250	0.625		51	0	1.000	0.510	
	1	2	767	0	1.000	0.767	86.4	767	0.767	72.6	150	0	1.000	0.598	87.1
	2	4	81	10	0.877	0.081	97.9	65	0.065	77.3	10	0	1.000	0.040	97.4
	3	3	53	1	0.981	0.133	98.5	49	0.123	80.8					98.6
	4	6	86	4	0.953	0.215		74	0.185	86.1					
	4	5	111	0	1.000	0.111	99.0	111	0.111	94.0	85	0	1.000	0.339	98.9
	5	7	30	2	0.933	0.075		27	0.068	95.9	45	0	1.000	0.450	
	5	8	16	1	0.938	0.016	99.6	15	0.015	97.0	2	0	1.000	0.008	98.9
	6	9	14	0	1.000	0.014		14	0.014	98.0	5	0	1.000	0.020	
	6	10	30	0	1.000	0.030		28	0.028	100	2	0	1.000	0.008	100
Fold 5	1	1	253	3	0.988	0.633		238	0.595	17.0	8	0	1.000	0.080	
	1	2	950	177	0.814	0.950	85.9	782	0.782	72.9	155	0	1.000	0.618	90.3
	2	3	154	0	1.000	0.385		154	0.385	83.9	93	0	1.000	0.930	
	2	4	200	0	1.000	0.200	98.4	200	0.200	98.1	75	0	1.000	0.299	97.4
	3	5	4	1	0.750	0.010		3	0.008	98.4	55	0	1.000	0.550	
	3	6	12	0	1.000	0.012	99.6	12	0.012	99.2	3	0	1.000	0.012	100
	4	7	5	0	1.000	0.013		5	0.013	99.6	2	0	1.000	0.020	
	4	8	6	0	1.000	0.006	100	6	0.006	100	12	0	1.000	0.048	100

Table G.5. Purity and power scores of hyperboxes generated by HCB – Skin subset 3

			Training Phase								Test Phase				
			Model MOU					After HTA			After HTA				
	Iter.	Box	N_{kl}	M_{kl}	Purity	Power	A_M	N_{kl}	Power	A_{TB}	N_{kl}	M_{kl}	Purity	Power	A
Fold 1	1	2	2000	800	0.600	1.000	71.4	424	0.212	15.1	72	0	1.000	0.144	19.6
	2	3	1577	800	0.493	0.789	70.7	253	0.127	24.2	60	1	0.983	0.120	24.7
	3	1	11	0	1.000	0.014		11	0.014						
	3	4	1030	595	0.422	0.515	60.6	124	0.062	29	22	0	1.000	0.044	30.4
	4	5	743	117	0.843	0.929	54.6	182	0.228	35.5	17	0	1.000	0.085	35.4
	5	6	720	436	0.394	0.360		430	0.215		75	0	1.000	0.150	
	5	7	200	37	0.815	0.250	65	87	0.109	54					69.3
	6	9	195	46	0.764	0.098	74.8	183	0.092	60.5	2	0	1.000	0.004	73.0
	7	10	190	6	0.968	0.095	78.9	184	0.092	67.1	43	0	1.000	0.086	85.4
	8	8	357	29	0.919	0.446		155	0.194						
	8	11	134	45	0.664	0.067	80.3	114	0.057	76.7	6	0	1.000	0.012	89.7
	9	12	220	32	0.855	0.275		119	0.149		19	0	1.000	0.095	
	9	13	40	18	0.550	0.020	83.5	19	0.010	81.6					89.7
	10	14	37	7	0.811	0.046		33	0.041		5	0	1.000	0.025	
	10	15	29	0	1.000	0.015	81.4	29	0.015	83.8					89.7
	11	17	22	2	0.909	0.011	82.5	21	0.011	84.6					92.0
	12	16	30	1	0.967	0.038		30	0.038		6	0	1.000	0.030	
	12	18	35	0	1.000	0.018	83.6	35	0.018	86.9	58	0	1.000	0.116	95.7
	13	20	142	1	0.993	0.071	93	141	0.071	91.9	129	0	1.000	0.258	97.6
	14	19	123	28	0.772	0.154	94.3	64	0.080	94.2	16	0	1.000	0.080	97.6
	15	22	56	1	0.982	0.070	94.1	44	0.055	95.8	18	0	1.000	0.090	98
	16	23	45	18	0.600	0.056	96.4	39	0.049	97.2	55	0	1.000	0.275	97.6
	17	21	25	0	1.000	0.013	97.9	25	0.013	98.1	7	0	1.000	0.014	97.6
	18	24	28	3	0.893	0.035		20	0.025		37	0	1.000	0.185	
	18	25	10	0	1.000	0.005	98.7	10	0.005	99.1	3	0	1.000	0.006	97.6
	19	26	12	2	0.833	0.015		10	0.013		14	0	1.000	0.070	
	19	27	4	0	1.000	0.002	99	4	0.002	99.6	2	0	1.000	0.004	98.1
	20	28	8	1	0.875	0.010		6	0.008	99.9	5	0	1.000	0.025	
	20	29	4	0	1.000	0.002	99.4	4	0.002	100	19	0	1.000	0.038	99.1

Table G.5 (continue)

			Training Phase								Test Phase				
			Model MOU					After HTA			After HTA				
	Iter.	Box	N_{kl}	M_{kl}	Purity	Power	A_M	N_{kl}	Power	A_{TB}	N_{kl}	M_{kl}	Purity	Power	A
	1	2	2000	800	0.600	1.000	71.4	410	0.205	14.6	1	0	1.000	0.002	27.6
	2	3	1238	800	0.354	0.619	55.8	637	0.319	37.4	153	0	1.000	0.306	26.0
	3	1	473	214	0.548	0.591		191	0.239	44.2	17	0	1.000	0.085	
	3	4	491	427	0.130	0.246	62.7	149	0.075	49.5	31	0	1.000	0.062	27.0
	4	5	420	136	0.676	0.525		276	0.345	59.4	14	0	1.000	0.070	
	4	6	768	454	0.409	0.384	74.5	203	0.102	66.6					55.6
	5	8	207	7	0.966	0.104	81.5	203	0.102	73.9	98	0	1.000	0.196	75.0
	6	7	215	15	0.930	0.269	85.4	173	0.216	80.1	23	0	1.000	0.115	76.9
	7	10	179	0	1.000	0.224	88.2	179	0.224	86.5	37	1	0.973	0.185	84.9
	8	9	77	10	0.870	0.039	88.9	57	0.029	88.5	105	0	1.000	0.210	84.9
	9	11	106	0	1.000	0.053		106	0.053	92.3	27	4	0.852	0.054	
	9	12	87	19	0.782	0.044	97.3	36	0.018	93.6	69	1	0.986	0.138	91.0
Fold 2	10	13	48	3	0.938	0.060	94.7	44	0.055	95.1	30	0	1.000	0.150	93.3
	11	14	56	15	0.732	0.028		32	0.016	96.3	24	0	1.000	0.048	
	11	15	23	1	0.957	0.012	97.0	21	0.011	97.0	8	0	1.000	0.016	97.3
	12	17	23	1	0.957	0.012	98.9	20	0.010	97.8	2	0	1.000	0.004	97.3
	13	16	34	4	0.882	0.043	99.0	12	0.015	98.2	4	1	0.750	0.020	97.6
	14	18	12	0	1.000	0.015	99.0	12	0.015	98.6	22	0	1.000	0.110	97.9
	15	19	5	0	1.000	0.006	99.4	5	0.006	98.8	1	0	1.000	0.005	97.9
	16	20	6	0	1.000	0.003		6	0.003	99.0	6	0	1.000	0.012	
	16	21	4	0	1.000	0.002	99.5	4	0.002	99.1	10	0	1.000	0.020	98.6
	17	22	14	1	0.929	0.018		13	0.016	99.6	5	1	0.800	0.025	
	17	23	2	0	1.000	0.001	99.7	2	0.001	99.7	1	0	1.000	0.002	98.4
	18	24	7	0	1.000	0.009		7	0.009	99.9	4	0	1.000	0.020	
	18	25	2	0	1.000	0.001	100	2	0.001	100					98.4

Table G.5 (continue)

		Training Phase								Test Phase				
		Model MOU					After HTA			After HTA				
	Iter.	Box	N_{kl}	M_{kl}	Purity	Power	A_M	N_{kl}	Power	A_{TB}	N_{kl}	M_{kl}	Purity	Power	A
Fold 3	1	2	2000	800	0.6000	1.0000	71.43	506	0.2530	18.07	44	0	1.000	0.088	24.28
	2	3	1194	800	0.3300	0.5970	60.57	609	0.3045	39.82	142	3	0.979	0.284	20.00
	3	1	486	118	0.7572	0.6075		161	0.2013		21	0	1.000	0.105	
	3	4	482	365	0.2427	0.2410	64.96	140	0.0700	50.57	28	0	1.000	0.056	21.43
	4	5	420	136	0.6762	0.5250		229	0.2863		35	0	1.000	0.175	
	4	6	768	454	0.4089	0.3840	74.79	228	0.1140	66.89	15	0	1.000	0.030	69.29
	5	8	207	7	0.9662	0.1035	80.25	198	0.0990	73.96	75	2	0.973	0.150	85.43
	6	7	215	15	0.9302	0.2688	81.46	138	0.1725	78.89	42	0	1.000	0.210	84.86
	7	9	179	0	1.0000	0.2238	83.50	96	0.1200	82.32	37	1	0.973	0.185	89.71
	8	10	77	10	0.8701	0.0385	82.50	157	0.0785	87.93	81	0	1.000	0.162	91.00
	9	11	106	0	1.0000	0.1325		82	0.1025		27	2	0.926	0.135	
	9	12	87	19	0.7816	0.0435	89.71	45	0.0225	92.46	69	1	0.986	0.138	93.00
	10	13	55	9	0.8364	0.0688	93.00	39	0.0488	93.86	30	0	1.000	0.150	95.71
	11	14	56	5	0.9107	0.0700		37	0.0463		10	0	1.000	0.050	
	11	15	23	17	0.2609	0.0115	94.14	10	0.0050	95.54	8	0	1.000	0.016	97.57
	12	17	26	8	0.6923	0.0130		20	0.0100		2	0	1.000	0.004	
	12	16	34	12	0.6471	0.0425	95.71	12	0.0150	96.68	4	1	0.750	0.020	97.57
	13	18	42	15	0.6429	0.0525	96.36	28	0.0350	97.68	11	0	1.000	0.055	97.57
	14	19	9	2	0.7778	0.0113	98.96	5	0.0063	97.86	1	0	1.000	0.005	97.86
	15	20	16	1	0.9375	0.0200	97.89	14	0.0175	98.36	6	0	1.000	0.030	95.29
	16	21	16	2	0.8750	0.0080		11	0.0055	0.00	2	0	1.000	0.004	
	16	22	19	5	0.7368	0.0238	99.50	13	0.0163	99.21	5	1	0.800	0.025	98.86
	17	23	9	0	1.0000	0.0045		9	0.0045		1	0	1.000	0.002	
	17	24	9	0	1.0000	0.0113	99.68	9	0.0113	99.86	4	0	1.000	0.020	95.29
	18	25	1	0	1.0000	0.0013		1	0.0013						
	18	26	3	0	1.0000	0.0015	100.00	3	0.0015	100.0					95.29

Table G.5 (continue)

			Training Phase								Test Phase				
			Model MOU					After HTA			After HTA				
	Iter.	Box	N_{kl}	M_{kl}	Purity	Power	A_M	N_{kl}	Power	A_{TB}	N_{kl}	M_{kl}	Purity	Power	A
	1	2	2000	800	0.6000	1.0000	71.43	462	0.2310	16.50	68	2	0.9706	0.1360	22.86
	2	3	1432	800	0.4413	0.7160	70.86	248	0.1240	25.36	60	1	0.9833	0.1200	20.29
	3	1	24	0	1.0000	0.0300		24	0.0300						
	3	4	1030	544	0.4718	0.5150	62.71	132	0.0660	30.93	22	0	1.0000	0.0440	28.57
	4	5	594	82	0.8620	0.7425	64.96	182	0.2275	37.43	17	0	1.0000	0.0850	35.43
	5	6	816	221	0.7292	0.4080		320	0.1600		75	2	0.9733	0.1500	
	5	7	681	368	0.4596	0.8513	74.46	87	0.1088	51.96					52.14
	6	9	215	47	0.7814	0.1075	80.25	183	0.0915	58.50	2	0	1.0000	0.0040	51.43
	7	10	198	1	0.9949	0.0990	81.46	192	0.0960	65.36	43	0	1.0000	0.0860	66.43
	8	8	357	21	0.9412	0.4463		155	0.1938		12	0	1.0000	0.0600	
	8	11	336	78	0.7679	0.1680	83.64	108	0.0540	74.75	36	1	0.9722	0.0720	72.85
	9	12	205	19	0.9073	0.2563		102	0.1275		34	2	0.9412	0.1700	
	9	13	129	56	0.5659	0.0645	88.86	24	0.0120	79.25					80.15
	10	14	201	15	0.9254	0.2513		51	0.0638		5	0	1.0000	0.0250	
Fold 4	10	15	82	26	0.6829	0.0410	85.71	29	0.0145	82.11					83.71
	11	17	68	11	0.8382	0.0340	86.14	21	0.0105	82.86					83.24
	12	16	45	6	0.8667	0.0563		34	0.0425		6	0	1.0000	0.0300	
	12	18	37	1	0.9730	0.0185	89.61	35	0.0175	85.32	58	0	1.0000	0.1160	86.00
	13	20	142	2	0.9859	0.0710	93.36	141	0.0705	90.36	78	1	0.9872	0.1560	87.71
	14	19	72	2	0.9722	0.0900		64	0.0800		16	0	1.0000	0.0800	91.00
	14	21	35	1	0.9714	0.0175	94.96	32	0.0160	93.79					
	15	22	49	2	0.9592	0.0613	96.46	44	0.0550	95.36	18	0	1.0000	0.0900	92.14
	16	23	45	2	0.9556	0.0563	97.21	39	0.0488	96.75	26	1	0.9615	0.1300	94.54
	17	24	38	1	0.9737	0.0475	98.86	37	0.0463	98.07	7	0	1.0000	0.0350	97.00
	18	25	20	0	1.0000	0.0250		20	0.0250		17	1	0.9412	0.0850	
	18	26	11	1	0.9091	0.0055	98.61	10	0.0050	99.14					98.43
	19	27	10	0	1.0000	0.0125		10	0.0125		14	0	1.0000	0.0700	
	19	28	4	0	1.0000	0.0020	99.64	4	0.0020	99.64	2	0	1.0000	0.0040	99.43
	20	29	6	0	1.0000	0.0075		6	0.0075	99.86	2	0	1.0000	0.0100	
	20	30	4	0	1.0000	0.0050	100.00	4	0.0050	100.0	1	0	1.0000	0.0050	98.43

Table G.5 (continue)

			Training Phase								Test Phase				
			Model MOU					After HTA			After HTA				
	Iter.	Box	N_{kl}	M_{kl}	Purity	Power	A_M	N_{kl}	Power	A_{TB}	N_{kl}	M_{kl}	Purity	Power	A
Fold 5	1	2	2000	800	0.6000	1.0000	71.43	512	0.0000	18.29	72	2	0.9722	0.1440	28.57
	2	3	1512	800	0.4709	0.7560	64.86	248	0.1240	27.14	56	1	0.9821	0.1120	22.86
	3	1	715	328	0.5413	0.8938		32	0.0400						
	3	4	918	216	0.7647	0.4590	67.93	132	0.0660	33.00	35	0	1.0000	0.0700	24.57
	4	5	612	120	0.8039	0.7650	74.92	168	0.2100	39.00	17	0	1.0000	0.0850	32.29
	5	6	816	177	0.7831	0.4080		320	0.1600		75	1	0.9867	0.1500	
	5	7	266	63	0.7632	0.3325	77.00	87	0.1088	53.54					58.29
	6	9	215	121	0.4372	0.1075	82.29	183	0.0915	60.07	19	0	1.0000	0.0380	66.86
	7	10	352	168	0.5227	0.1760	87.68	105	0.0525	63.82	39	0	1.0000	0.0780	75.00
	8	8	345	16	0.9536	0.4313		155	0.1938		23	0	1.0000	0.1150	
	8	11	113	1	0.9912	0.0565	85.71	108	0.0540	73.21	36	1	0.9722	0.0720	83.50
	9	12	205	21	0.8976	0.2563		102	0.1275		34	2	0.9412	0.1700	
	9	13	79	18	0.7722	0.0395	89.95	39	0.0195	78.25					88.86
	10	14	169	19	0.8876	0.2113		72	0.0900		25	0	1.0000	0.1250	
	10	15	86	46	0.4651	0.0430	88.86	29	0.0145	81.86					85.43
	11	17	64	16	0.7500	0.0320	89.61	21	0.0105	82.61					89.71
	12	16	68	12	0.8235	0.0850		35	0.0438		6	0	1.0000	0.0300	
	12	18	76	10	0.8684	0.0380	91.39	41	0.0205	85.32	32	1	0.9688	0.0640	89.71
	13	20	136	6	0.9559	0.1700		124	0.1550		78	1	0.9872	0.3900	
	13	19	80	2	0.9750	0.0400	93.29	26	0.0130	90.68					89.61
	14	21	66	2	0.9697	0.0825		64	0.0800		16	0	1.0000	0.0800	
	14	22	42	5	0.8810	0.0210	94.11	33	0.0165	94.14					91.00
	15	23	54	6	0.8889	0.0675		30	0.0375		18	0	1.0000	0.0900	
	15	24	36	3	0.9167	0.0180	96.00	26	0.0130	96.14					93.29
	16	26	35	2	0.9429	0.0438	96.21	26	0.0325	97.07	10	0	1.0000	0.0500	95.71
	17	25	28	1	0.9643	0.0350		27	0.0338		7	0	1.0000	0.0350	
	17	27	19	2	0.8947	0.0095	98.43	17	0.0085	98.64					97.57
	18	29	17	2	0.8824	0.0085	98.86	17	0.0213	99.25	2	0	1.0000	0.0100	98.71
	19	28	5	1	0.8000	0.0063	98.92	5	0.0025	99.43					98.71
	20	30	8	1	0.8750	0.0100	99.00	6	0.0075	99.64					98.71
	21	21	5	1	0.8000	0.0063		4	0.0050						
	21	32	4	1	0.7500	0.0020	99.21	3	0.0038	99.89					98.71
	22	33	2	0	1.0000	0.0025		2	0.0010						
	22	34	1	0	1.0000	0.0005	100.0	1	0.0013	100.0					98.71

Table G.6. Purity and power scores of hyperboxes generated by HCB – Skin subset 4

			Training Phase								Test Phase				
			Model MOU					After HTA			After HTA				
	Iter.	Box	N_{kl}	M_{kl}	Purity	Power	A_M	N_{kl}	Power	A_{TB}	N_{kl}	M_{kl}	Purity	Power	A
Fold 1	1	2	2000	800	0.600	1.000	71.4	1620	0.810	82.1	136	0	1.000	0.272	85.4
	2	1	797	32	0.960	0.996		679	0.849		139	0	1.000	0.695	
	2	3	351	10	0.972	0.176	98.0	335	0.168	36.2	119	0	1.000	0.238	96.3
	3	4	100	0	1.000	0.125	99.3	100	0.125	39.8	58	0	1.000	0.290	99.9
	4	5	45	0	1.000	0.023		45	0.023		45	0	1.000	0.090	
	4	6	21	0	1.000	0.026	100	21	0.026	42.1	5	0	1.000	0.025	100
Fold 2	1	2	2000	800	0.600	1.000	71.4	1618	0.809	57.8	401	0	1.000	0.802	88.0
	2	3	360	10	0.972	0.180	98.6	337	0.169	69.8	89	0	1.000	0.178	95.6
	3	1	781	29	0.963	0.976	98.8	662	0.828	93.5	167	1	0.994	0.835	99.4
	4	4	45	0	1.000	0.023		45	0.023	95.1	8	0	1.000	0.016	
	4	5	138	0	1.000	0.173	100	138	0.173	100	34	0	1.000	0.170	99.9
Fold 3	1	2	1620	24	0.985	0.810	85.3	1616	0.808	57.7	344	0	1.000	0.688	89.6
	2	1	738	12	0.984	0.923		654	0.818		138	0	1.000	0.690	
	2	3	328	29	0.912	0.164	98.5	141	0.071	86.1	67	0	1.000	0.134	97.9
	3	4	140	5	0.964	0.175		122	0.153		56	0	1.000	0.280	
	3	5	243	0	1.000	0.122	99.4	243	0.122	99.1	88	0	1.000	0.176	100
	4	6	23	5	0.783	0.029		17	0.021		5	0	1.000	0.025	
	4	7	7	0	1.000	0.009	100	7	0.009	100					100
Fold 4	1	2	1607	0	1.000	0.804	86.0	1607	0.804	57.4	371	0	1.000	0.742	92.0
	2	3	369	6	0.984	0.185	98.4	349	0.175	69.9	76	0	1.000	0.152	96.0
	3	1	781	30	0.962	0.976		685	0.856		158	0	1.000	0.790	
	3	4	49	2	0.959	0.025	95.8	44	0.022	95.9	50	0	1.000	0.100	98.3
	4	5	115	0	1.000	0.144	100	115	0.144	100	93	0	1.000	0.465	100
Fold 5	1	2	1615	0	1.000	0.808	86.6	1615	0.808	57.7	376	0	1.000	0.752	70.0
	2	3	354	6	0.983	0.177	98.7	346	0.173	70.0	115	1	0.991	0.230	96.4
	3	1	784	29	0.963	0.980		663	0.829		9	0	1.000	0.045	
	3	4	26	0	1.000	0.013	99.0	39	0.020	95.1	136	0	1.000	0.272	98.5
	4	5	132	8	0.939	0.165		117	0.146		55	0	1.000	0.275	
	4	6	20	0	1.000	0.025	100	20	0.025	100	7	0	1.000	0.035	99.9

Table G.7. Purity and power scores of hyperboxes generated by HCB – Skin subset 5

			Training Phase								Test Phase				
			Model MOU					After HTA			After HTA				
	Iter.	Box	N_{kl}	M_{kl}	Purity	Power	A_M	N_{kl}	Power	A_{TB}	N_{kl}	M_{kl}	Purity	Power	A
Fold 1	1	2	4000	1600	0.600	1.000	71.4	3222	0.806	57.5	267	0	1.000	0.267	73.5
	2	1	1617	353	0.782	1.000		1081	0.676		263	0	1.000	0.658	
	2	3	695	66	0.905	0.174	91.2	646	0.162	88.4	691	0	1.000	0.691	97.2
	3	4	343	0	1.000	0.214	95.7	343	0.214	94.5	97	0	1.000	0.243	97.4
	4	6	176	0	1.000	0.110	98.9	176	0.110	97.6	39	0	1.000	0.098	98.6
	5	5	101	18	0.822	0.025	98.9	71	0.018	98.9	15	0	1.000	0.015	99.6
	6	7	43	2	0.953	0.011	99.6	41	0.010	99.6	19	0	1.000	0.019	97.2
	7	8	16	0	1.000	0.004		16	0.004		6	0	1.000	0.006	
	7	9	4	0	1.000	0.001	100	4	0.001	100	1	0	1.000	0.001	100
Fold 2	1	2	4000	1000	0.750	1.000	82.1	3214	0.804	57.4	283	0	1.000	0.283	82.4
	2	1	1219	353	0.710	0.762	88.9	1129	0.706	77.6	138	0	1.000	0.345	98.1
	3	3	683	1	0.999	0.171		681	0.170		677	1	0.999	0.677	
	3	4	174	53	0.695	0.109	75.0	105	0.066	91.6	149	0	1.000	0.373	98.6
	4	5	306	0	1.000	0.191	98.0	306	0.191	97.1	73	0	1.000	0.183	99.1
	5	7	60	0	1.000	0.038		60	0.038		42	0	1.000	0.105	
	5	8	17	0	1.000	0.004	99.1	17	0.004	98.4	5	0	1.000	0.005	99.0
	6	6	54	0	1.000	0.014		54	0.014		22	0	1.000	0.022	
	7	9	22	0	1.000	0.006	99.8	22	0.006	99.8	5	0	1.000	0.005	99.8
	7	10	12	0	1.000	0.003	100	12	0.003	100	2	0	1.000	0.002	99.8
Fold 3	1	2	4000	1600	0.600	1.000	71.4	3229	0.807	57.7	308	0	1.000	0.308	85.6
	2	1	1251	27	0.978	0.782		956	0.598		228	0	1.000	0.570	
	2	3	693	30	0.957	0.173	91.4	672	0.168	86.7	650	0	1.000	0.650	93.5
	3	4	480	0	1.000	0.300	96.2	480	0.300	95.3	124	0	1.000	0.310	98.0
	4	6	164	0	1.000	0.103		164	0.103		47	0	1.000	0.118	
	4	5	51	0	1.000	0.013	99.1	51	0.013	99.1	26	0	1.000	0.026	99.3
	5	8	17	0	1.000	0.004		17	0.004		3	0	1.000	0.003	
	6	7	30	0	1.000	0.019	100	30	0.008	100	11	0	1.000	0.011	99.9
	6	9	1	0	1.000	0.000	100	1	0.000	100	0	0			99.9
Fold 4	1	2	4000	1600	0.600	1.000	71.4	3213	0.803	57.4	292	0	1.000	0.292	69.8
	2	3	693	26	0.962	0.173	94.2	684	0.171	69.6	671	0	1.000	0.671	95.4
	3	1	1545	144	0.907	0.966		988	0.618		224	0	1.000	0.560	
	3	4	47	28	0.404	0.012	94.9	19	0.005	87.6	2	0	1.000	0.002	91.3
	4	5	550	0	1.000	0.344	98.3	550	0.344	97.4	152	0	1.000	0.380	98.7
	5	7	77	15	0.805	0.048		62	0.039		21	0	1.000	0.053	
	5	6	49	0	1.000	0.012	99.5	49	0.012	99.4	28	0	1.000	0.028	99.6
	6	8	34	0	1.000	0.021	100	34	0.009	100	4	0	1.000	0.004	
	6	9	1	0	1.000	0.000	100	1	0.000	100	0	0			99.8
Fold 5	1	2	4000	1600	0.600	1.000	71.4	3213	0.803	57.4	255	0	1.000	0.255	84.6
	2	1	1520	176	0.884	0.950	90.3	981	0.613	74.9	211	0	1.000	0.528	96.4
	3	3	640	3	0.995	0.160	92.4	632	0.158	86.2	711	0	1.000	0.711	97.8
	4	4	559	0	1.000	0.349		559	0.349		139	0	1.000	0.348	
	4	5	92	12	0.870	0.023	97.6	69	0.017	97.4	21	0	1.000	0.021	99.8
	5	6	60	0	1.000	0.038		60	0.038		49	0	1.000	0.123	
	5	7	52	0	1.000	0.013	99.3	52	0.013	99.4	6	0	1.000	0.006	99.9
	6	8	30	0	1.000	0.019		30	0.008		5	0	1.000	0.005	
	6	9	4	0	1.000	0.001	100	4	0.001	100	2	0	1.000	0.002	99.9

Table G.8. Purity and power scores of hyperboxes generated by HCB – Skin subset 6

			Training Phase								Test Phase				
			Model MOU					After HTA			After HTA				
	Iter	Box	N_{kl}	M_{kl}	Purity	Power	A_M	N_{kl}	Power	A_{TB}	N_{kl}	M_{kl}	Purity	Power	A
Fold 1	1	2	4000	1600	0.600	0.714	71.43	2068	0.369	36.93	304	0	1.000	0.304	64.43
	2	1	1314	236	0.820	0.821	76.86	974	0.609	54.32	197	0	1.000	0.493	87.21
	3	3	1098	183	0.833	0.196	77.70	592	0.106	64.89	93	0	1.000	0.093	81.86
	4	5	159	0	1.000	0.028	78.91	163	0.029	67.80	149	0	1.000	0.149	89.14
	5	6	835	340	0.593	0.149	87.46	793	0.142	81.96	344	0	1.000	0.344	89.71
	6	4	562	7	0.988	0.351		521	0.326		39	0	1.000	0.098	
	6	7	297	11	0.963	0.053	97.66	266	0.048	96.02	66	0	1.000	0.066	98.00
	7	9	45	0	1.000	0.008	98.09	45	0.008	98.04	13	0	1.000	0.013	98.29
	8	8	70	2	0.971	0.044		68	0.043		46	0	1.000	0.115	
	8	10	38	2	0.947	0.007	98.70	34	0.006	98.64	12	0	1.000	0.012	98.50
	9	11	40	3	0.925	0.025		34	0.021		85	0	1.000	0.213	
	9	12	19	0	1.000	0.003	99.64	19	0.003	99.59	5	0	1.000	0.005	99.29
	10	14	12	0	1.000	0.002	99.86	12	0.002	99.80	3	0	1.000	0.003	99.64
	11	13	3	0	1.000	0.002		3	0.002		10	0	1.000	0.025	
	11	15	8	0	1.000	0.001	100.0	8	0.001	100.0	15	0	1.000	0.015	100.0

H. Progress of HCB-f for Datasets with Smaller Number of Features

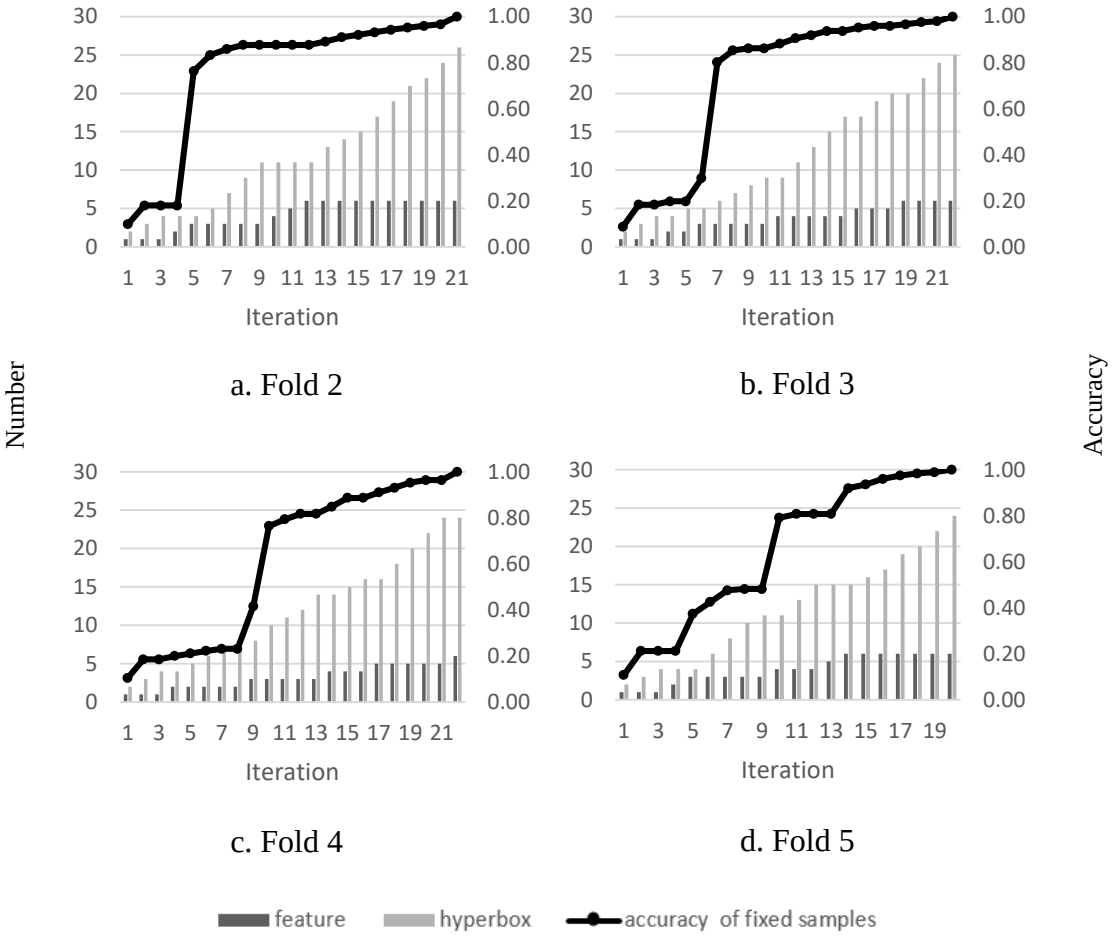


Figure H.1. Progress of HCB-f for Breast Cancer dataset

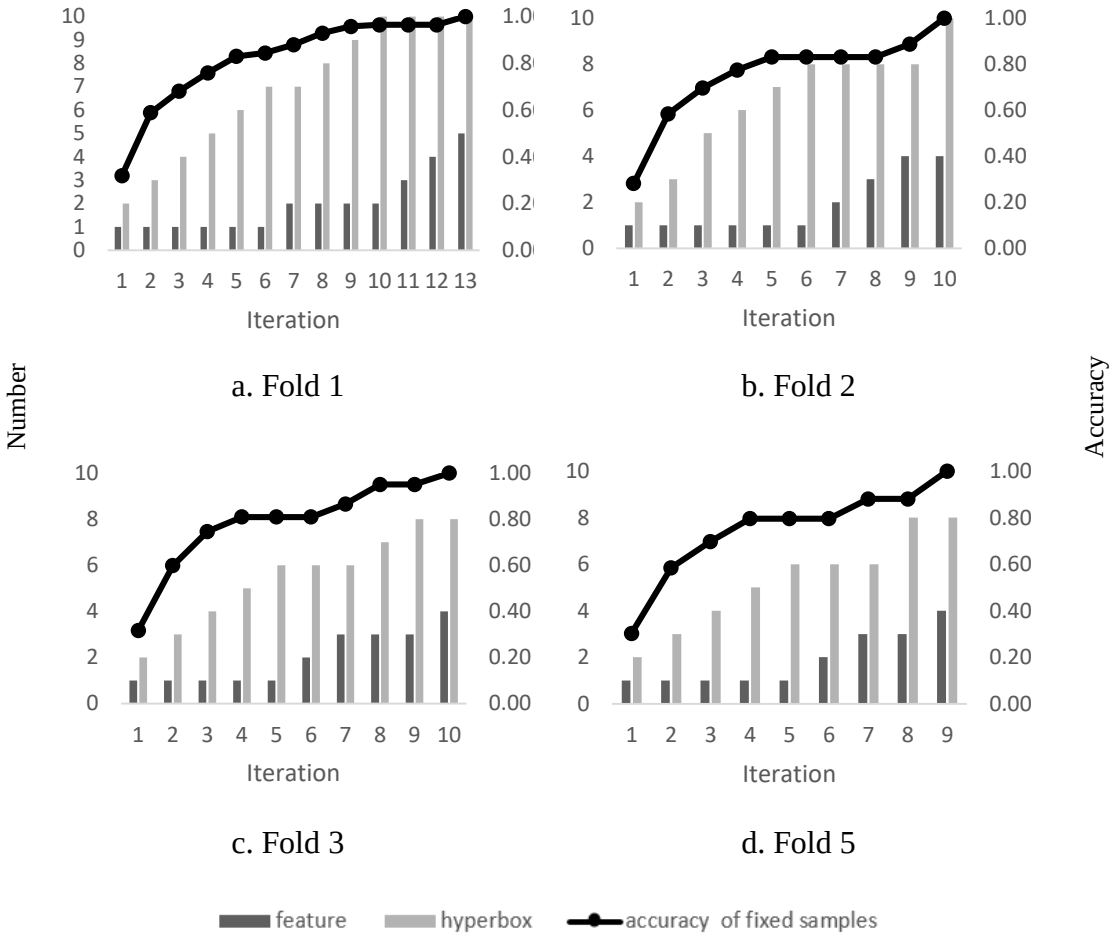


Figure H. 2. Progress of HCB-f for Wine Dataset (2 class)

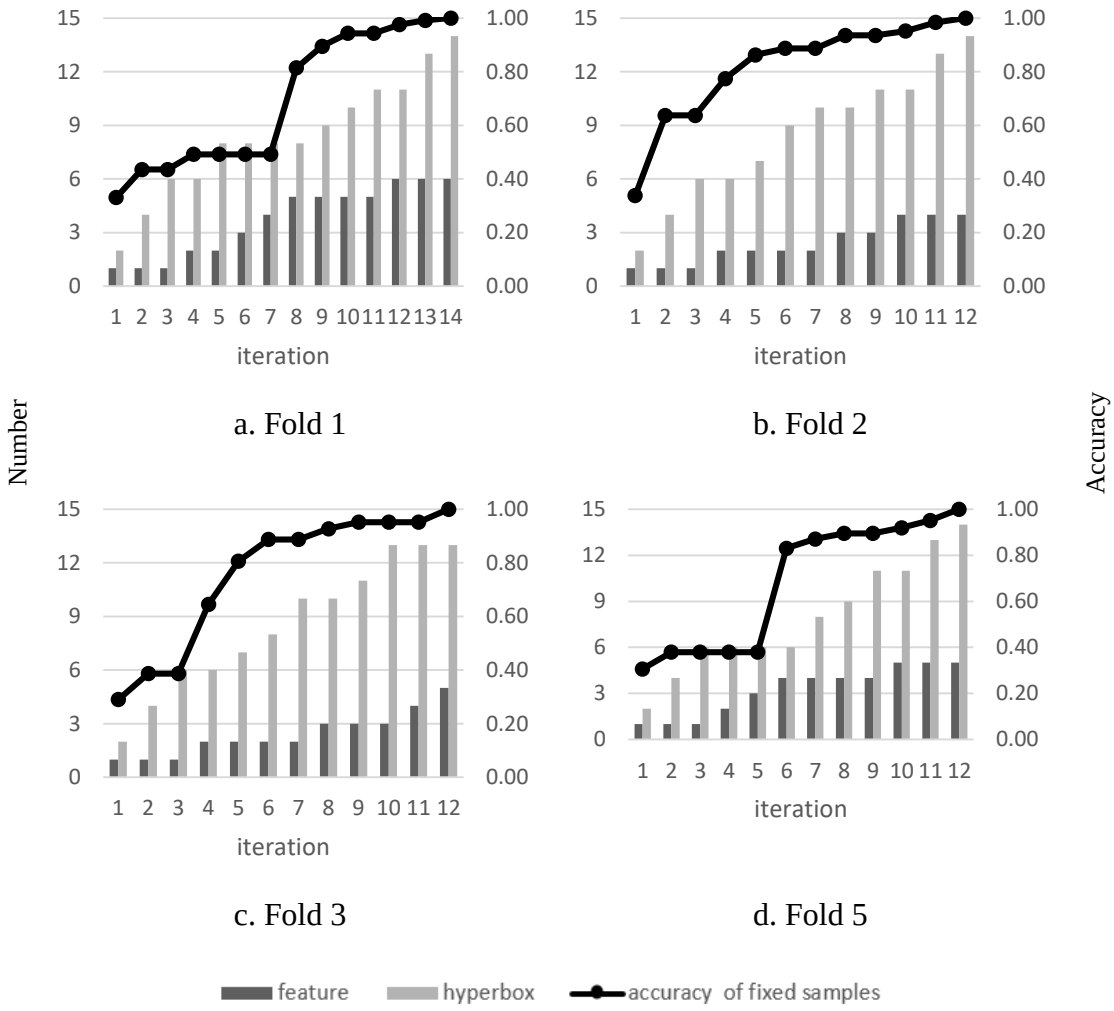


Figure H.3. Progress of HCB-f for Hepatitis Dataset

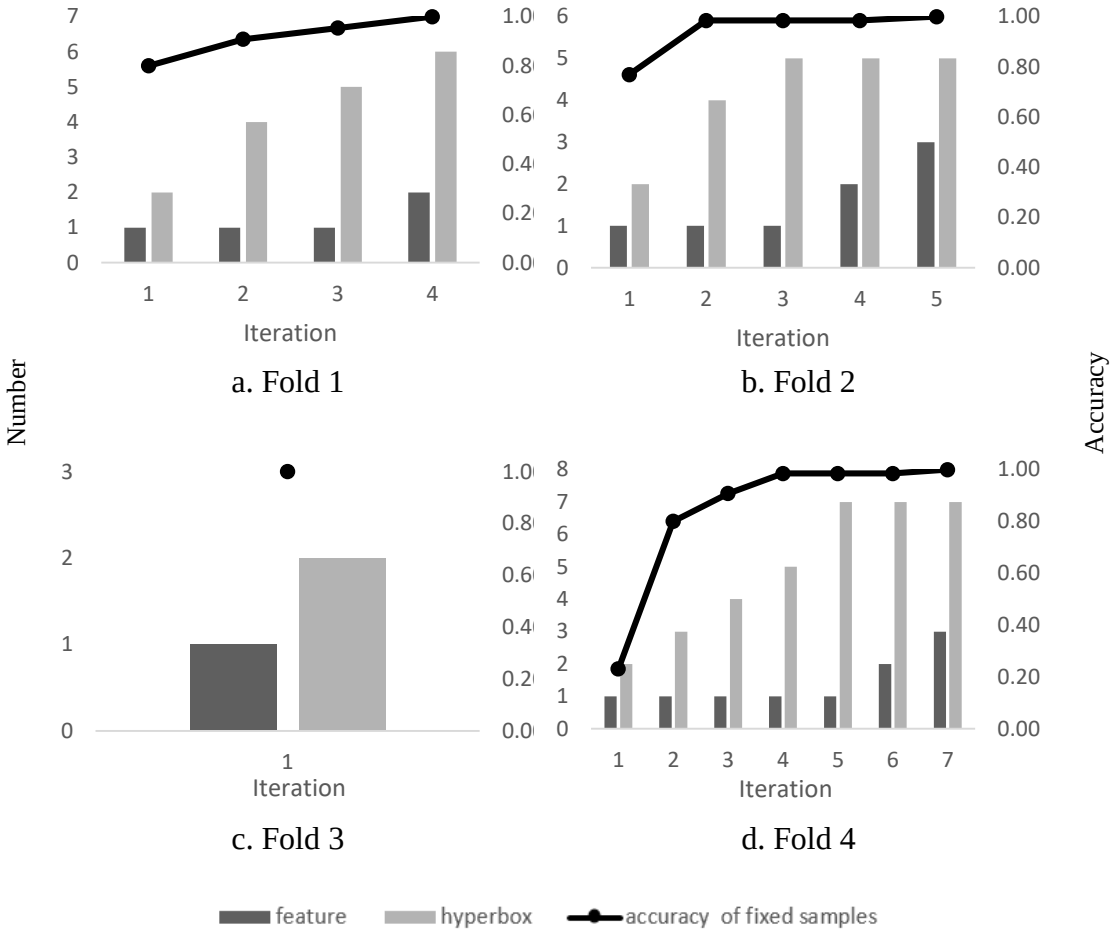


Figure H.4. Progress of HCB-f for Firm dataset

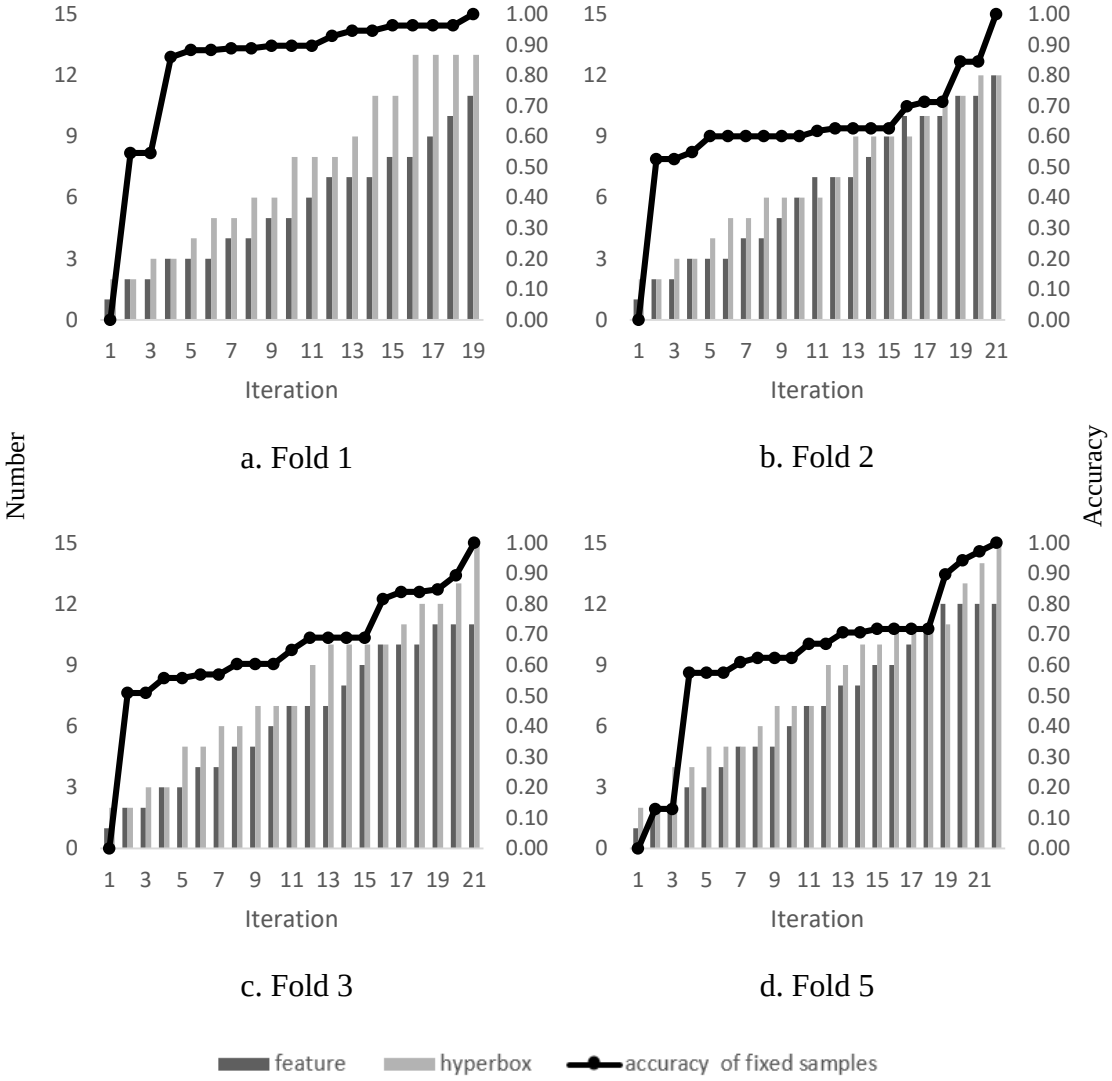


Figure H.5. Progress of HCB-f for Voting dataset

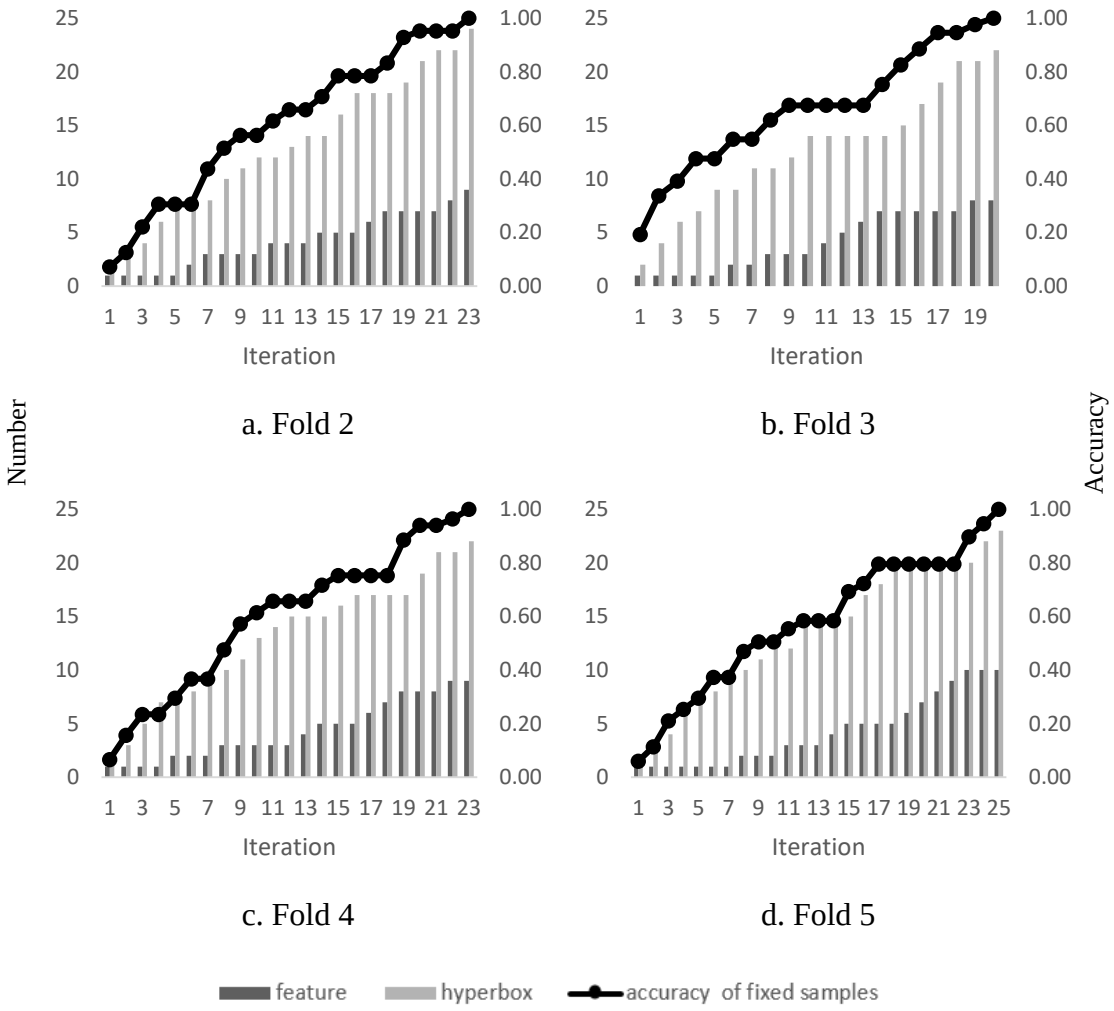


Figure H.6. Progress of HCB-f for Sonar dataset

I. Progress of HCB-f for Microarray Datasets

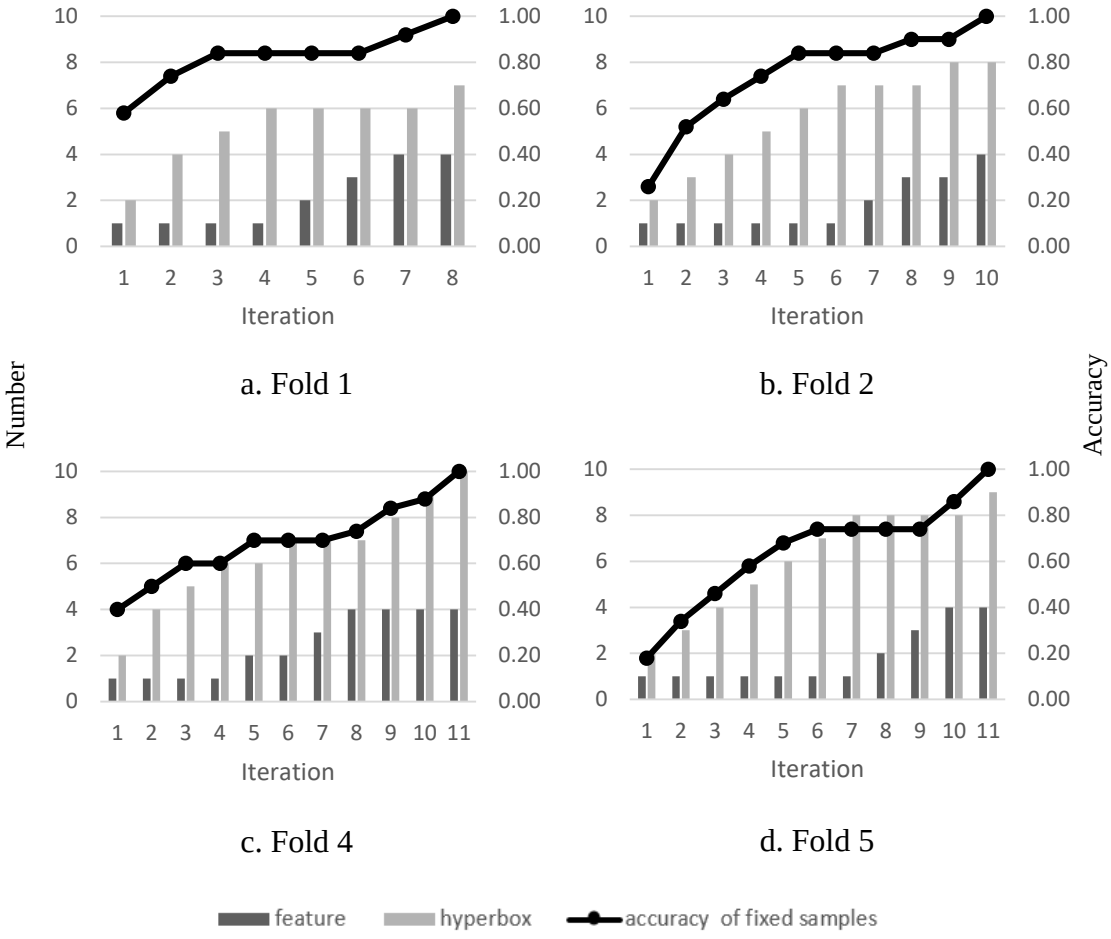


Figure I.1. Progress of HCB-f for Colon dataset

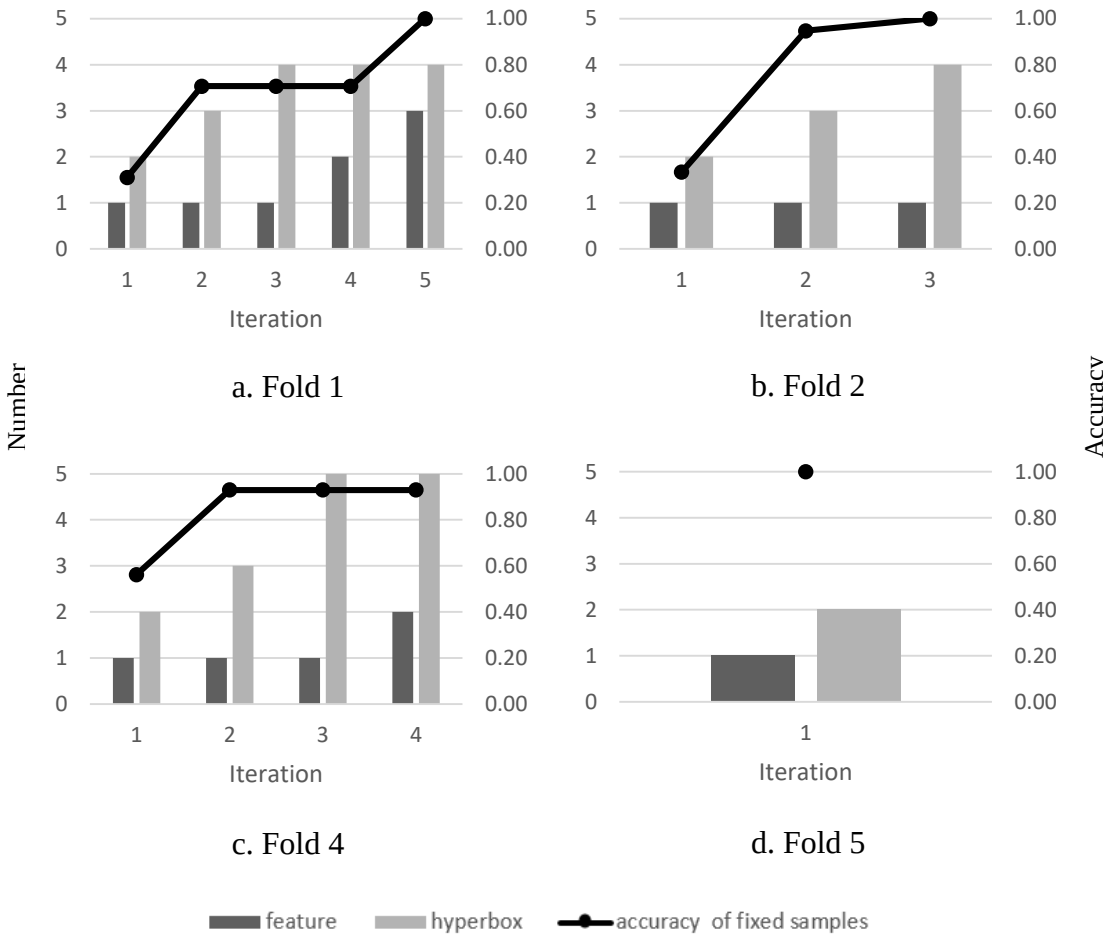


Figure I.2. Progress of HCB-f for Leukemia dataset

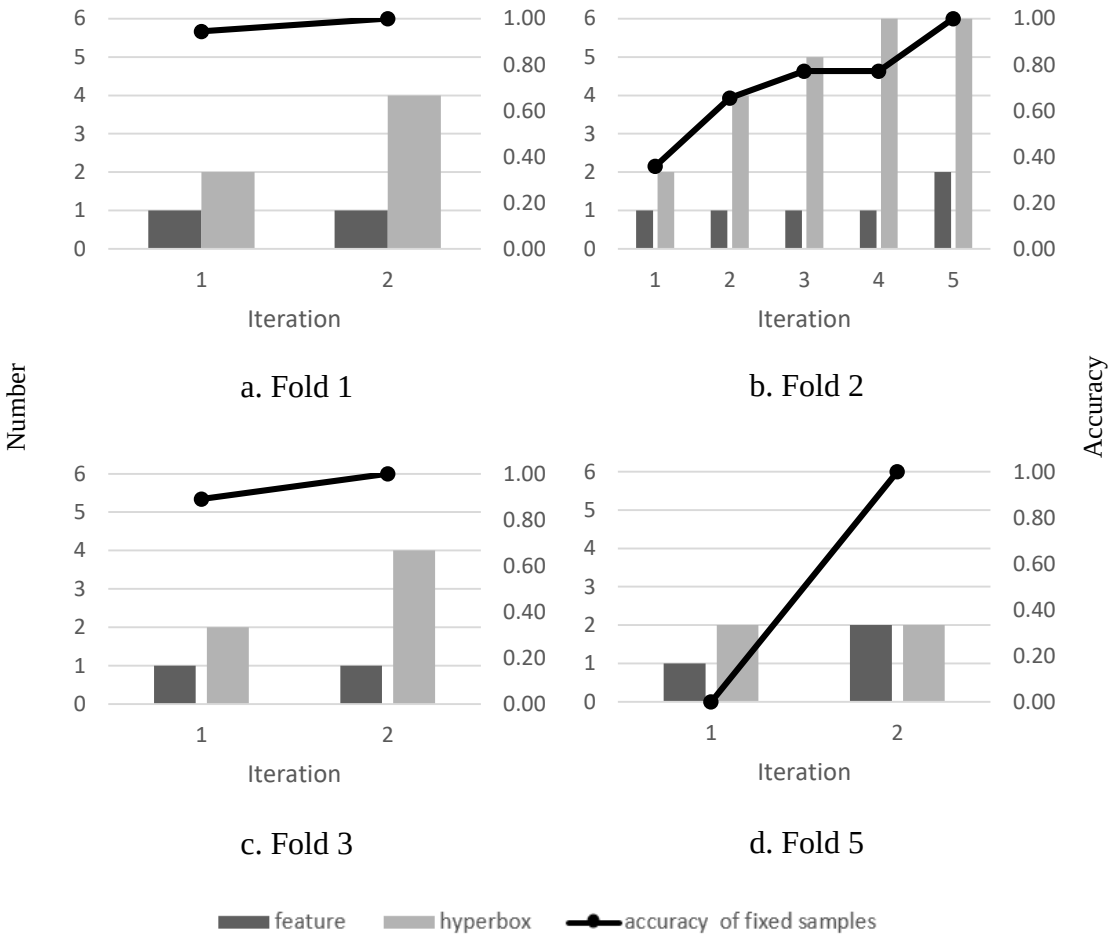


Figure I.3. Progress of HCB-f for Lung Cancer dataset

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name	: Akbulut, Derya
Nationality	: Turkish (TC)
Date and Place of Birth	: 10 May 1986, Samsun
Marital Status	: Single
Phone	: +90 536 834 1578 92
Fax	:
E-mail	: akbulut.derya@metu.edu.tr

EDUCATION

Degree	Institution	Year of Graduation
MS	Cankaya Uni., Industrial Engineering	2012
BS	Cankaya Uni., Industrial Engineering	2009
High School	Samsun Anadolu High School, Samsun	2004

WORK EXPERIENCE

Year	Place	Enrollment
2018-present	Cankaya Uni., Industrial Engineering	Lecturer
2009-2018	Cankaya Uni., Industrial Engineering	Expert

FOREIGN LANGUAGES

Advanced English

HOBBIES

Travelling, Computer Technologies, Movies,