Pós-Graduação em Ciência da Computação

AN ONLINE LOCAL POOL GENERATION METHOD FOR DYNAMIC CLASSIFIER SELECTION

By

MARIANA DE ARAÚJO SOUZA

M.Sc. Dissertation

Federal University of Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE/2018

Mariana de Araújo Souza

AN ONLINE LOCAL POOL GENERATION METHOD FOR DYNAMIC CLASSIFIER SELECTION

A M.Sc. Dissertation presented to the Center for Informatics of Federal University of Pernambuco in partial fulfillment of the requirements for the degree of Master of Science in Computer Science.

Advisor: George Darmiton da Cunha Cavalcanti
Co-Advisor: Robert Sabourin

RECIFE
2018

Mariana de Araújo Souza
An Online Local Pool Generation Method for Dynamic Classifier Selection/ Mariana de Araújo Souza. – RECIFE, 2018-
83 p. : il. (algumas color.) ; 30 cm.

Advisor George Darmiton da Cunha Cavalcanti

M.Sc. Dissertation – Universidade Federal de Pernambuco, 2018.

1. Palavra-chave1. 2. Palavra-chave2. I. Orientador. II. Universidade xxx. III. Faculdade de xxx. IV. Título

CDU 02:141:005.7

Dissertação de mestrado apresentada por **Mariana de Araújo Souza** ao programa de Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título **An Online Local Pool Generation Method for Dynamic Classifier Selection**, orientada pelo **Prof. George Darmiton da Cunha Cavalcanti** e aprovada pela banca examinadora formada pelos professores:

Prof. Adriano Lorena Inácio de Oliveira
Centro de Informática/UFPE

Prof. George Darmiton da Cunha Cavalcanti
Centro de Informática/UFPE

Prof. Marjory Cristiany Da Costa Abreu
Departamento de Informática e Matemática Aplicada/UFRN

RECIFE
2018

To my “mãe do mel”, Iraci.

Acknowledgements

This dissertation has benefited from the support of many people, so I would like to spend a few lines to sincerely thank some of them.

Firstly, I would like to thank my advisors, Prof George Darmiton and Prof Robert Sabourin, for their guidance and patience throughout the completion of this work. Their expertise and support were of great importance not only for finishing this work but also as an encouragement for achieving bigger goals as a researcher. Thank you very much!

A special thanks to Rafael Cruz, who immensely contributed to this work. It has been a pleasure working with you, and I am hugely indebted for your kindness and your valuable comments and suggestions.

I would also like to thank all members from the VIISAR research group. Our meetings, though infrequent, have widened my views on many interesting and often related research topics.

On a more personal level, I would like to thank my family and my friends for always being there for me when I needed the most.

To my fiancé, Saulo Pereira, for his support and understanding, and for rescuing me whenever I dived too much for too long into my work. I own much of my sanity to you, my love.

Many thanks to my sister, who has never failed to support me even with thousands of kilometers between us.

Last, but first in my heart, I am forever grateful to my parents for their love and support in every step of the way. Their encouragement to pursue my education was always present throughout my entire life, and without them I would not have come this far.

Educação não transforma o mundo. Educação muda pessoas. Pessoas transformam o mundo.

—PAULO FREIRE

Resumo

Técnicas de Seleção Dinâmica de Classificador (DCS) têm dificuldade em selecionar o classificador mais competente em um pool, mesmo quando a presença do mesmo é garantida. Visto que as técnicas de DCS utilizam apenas dados locais para estimar a competência de um classificador, a maneira na qual o pool é gerado poderia afetar na escolha do melhor classificador para uma dada instância. Isto é, a perspectiva global na qual os pools são gerados podem não ajudar as técnicas de DCS na seleção de um classificador competente para instâncias que são mais prováveis de ser incorretamente classificadas. Portanto, é proposto neste trabalho um método online de geração de pool de classificadores que produz um pool localmente preciso para amostras de teste em áreas de sobreposição de classes no espaço de características. Dessa forma, ao usar classificadores que foram gerados em um escopo local, poderia ser mais fácil para as técnicas de DCS selecionarem o melhor classificador para essas instâncias mais difíceis de classificar. Para as amostras posicionadas longe das bordas das classes, uma simples abordagem utilizando os vizinhos mais próximos é usada no método proposto. Nesta dissertação, uma visão geral da área de Sistemas de Multiplos Classificadores é apresentada, com foco em técnicas de seleção dinâmica. As técnicas de DCS mais relevantes também são introduzidas, e uma análise da eficácia das mesmas em selecionar o classificador mais competente para uma dada amostra em um pool gerado globalmente é apresentada. Baseado nessa análise, um método de geração local de pool de classificadores é proposto e analisado passo-a-passo. O método proposto é então avaliado usando 20 problemas de classificação, e o efeito de seus parâmetros no desempenho são analisados. Além disso, um estudo comparativo com outros métodos relacionados é realizado e os resultados experimentais mostram que as técnicas de DCS foram mais capazes de selecionar o melhor classificador para uma dada instância com o pool proposto, que foi gerado localmente, do que com um pool gerado de forma global. Ademais, o método proposto obteve uma maior taxa de acerto em comparação com os métodos relacionados para todas as técnicas de DCS, em média, e apresentou uma melhora considerável para problemas com uma alta proporção de instâncias próximas das bordas entre as classes. O método proposto também obteve um aumento significativo no desempenho em comparação com a maioria dos métodos relacionados que foram avaliados neste trabalho.

Palavras-chave: Sistemas de Múltiplos Classificadores, Geração de Pool de Classificadores, Seleção Dinâmica de Classificador

Abstract

Dynamic Classifier Selection (DCS) techniques have difficulty in selecting the most competent classifier in a pool, even when its presence is assured. Since the DCS techniques rely only on local data to estimate a classifier's competence, the manner in which the pool is generated could affect the choice of the best classifier for a given instance. That is, the global perspective in which pools are generated may not help the DCS techniques in selecting a competent classifier for instances that are likely to be misclassified. Thus, it is proposed in this work an online pool generation method that produces a locally accurate pool for test samples in overlap regions of the feature space. That way, by using classifiers that were generated in a local scope, it could be easier for the DCS techniques to select the best one for those instances they would most probably misclassify. For the instances that are far from the class borders, a simple nearest neighbors rule is used in the proposed method. In this dissertation, an overview of the area of Multiple Classifier Systems is presented, with focus on Dynamic Selection schemes. The most relevant DCS techniques are also introduced, and an analysis on their effectiveness in selecting the most competent classifier for a given instance in a globally generated pool is presented. Based on that analysis, an online local pool generation scheme is proposed and analyzed step-by-step. The proposed method is then evaluated over 20 classification problems, and the effect of its parameters on performance are analyzed. Moreover, a comparative study with other related methods is performed and the experimental results show that the DCS techniques were more able to select the best classifier for a given sample when using the proposed locally generated pool than when using a globally generated pool. Furthermore, the proposed method obtained a greater accuracy rate in comparison with the related methods for all DCS techniques, on average, and presented a considerable improvement for problems with a high proportion of borderline instances. It also yielded a significant increase in performance compared to most related methods evaluated in this work.

Keywords: Multiple Classifier Systems, Pool Generation, Dynamic Classifier Selection

List of Figures

List of Tables

List of Acronyms

Contents

1

Introduction

Multiple Classifier Systems (MCS) aim to improve the overall performance of a pattern recognition system by combining numerous base classifiers (WOŹNIAK; GRAÑA; CORCHADO, 2014; KITTLER et al., 1998; KUNCHEVA, 2014). An MCS contains three stages (CRUZ; SABOURIN; CAVALCANTI, 2018): (1) Generation, (2) Selection and (3) Aggregation. In the first stage, a pool of classifiers is generated using the training data. In the second stage, a non-empty subset of classifiers from the pool is selected to perform the classification task. In the third and last stage, the selected classifiers' predictions are combined to form the final system's output. There are two possible approaches in the selection stage: Static Selection (SS), in which the same set of classifiers is used to label all unknown instances, or Dynamic Selection (DS), which selects certain classifiers from the pool according to each query sample.

The DS techniques, which have been shown to outperform SS techniques, specially on ill-defined problems (BRITTO; SABOURIN; OLIVEIRA, 2014; CRUZ et al., 2015), are based on the idea that the classifiers in the pool are individually competent in different regions of the feature space. The aim of the selection scheme is, then, to choose the classifier(s) that is(are) best fit, according to some criterion, for classifying each unknown instance in particular (BRITTO; SABOURIN; OLIVEIRA, 2014). The amount of classifiers singled out to label a given sample separates the DS schemes in two groups (KO; SABOURIN; JR., 2008): Dynamic Classifier Selection (DCS) techniques, in which the classifier with highest estimated competence in the pool is selected, and Dynamic Ensemble Selection (DES) schemes, in which a locally accurate subset of classifiers from the pool is chosen and combined to label the test sample.

In the context of DCS, the Oracle (KUNCHEVA, 2002) can be defined as an abstract model that mimics the perfect selection scheme: it always selects the classifier that correctly labels a given instance, if the pool contains such classifier. Thus, the Oracle accuracy rate is the theoretical limit for DCS techniques. That way, the model can measure how close a DCS technique is from its maximum performance and indicates whether there is still room for improvements in classification accuracy, for a given pool of classifiers.

1.1 Motivation and Problem Statement

Since the Oracle simulates the ideal selector, it has been generally used in comparative studies with regards to other selection schemes. However, a significant gap between the model and the DS techniques have been shown in several works (DIDACI et al., 2005; KO; SABOURIN; SOUZA BRITTO JR, 2007).

The behavior of the Oracle regarding pool generation for DCS techniques was characterized in SOUZA et al. (2017). It was shown that even though the presence of one competent classifier was assured for a given instance, the DCS techniques still struggled to select it. This analysis was done using a pool generation method that guarantees an Oracle accuracy rate of 100% on the training set.

It was reasoned that the nature of the Oracle makes it not very well suited to guide the generation of a pool of classifiers for DCS since the model perceives the classification problem globally, while DCS techniques use only local data to select the most competent classifier for each instance. Thus, the difference in perspective between the Oracle model and the DCS techniques hinder the latter in the process of achieving a recognition rate closer to their theoretical maximum.

1.2 Overview of the Proposal

Based on these observations, it is proposed in this work a pool generation method which attempts to explore the Oracle's properties on a local scope. Since the Oracle and DCS techniques view the problem from different perspectives, using the model in a local setting to match these perspectives may help the DCS techniques in the choice of the most competent classifier for a given instance. This work focus only on DCS techniques since their relationship to the Oracle was already characterized previously, and so the results can be further analyzed based on certain aspects presented in the previous work.

Thus, the main idea is to use the Oracle model to guide the generation of subsets of classifiers for a specialized pool, so that each subpool is trained over a difficult region of the feature space, in hopes that using Oracle information for these regions separately will lead to a more accurate set of classifiers for the areas a globally generated pool is more prone to misclassify. A difficult region in this context is any area of the feature space in which there is overlap between the problem's classes. That way, each difficult region will be fully covered by subpools of classifiers, so that when a query instance's location is identified as a difficult area, the selection will be performed on the locally generated pool, formed by the most accurate classifiers in this region. For samples located in regions with no class overlap, the classification task is performed using a simple K-Nearest Neighbors (K-NN) rule.

1.3 Research Methodology

In this work, we aim to find out whether the use of locally generated pools is advantageous in DCS context. The research questions we intend to answer are: (1) does the use of locally generated pools aid the DCS techniques in selecting the best classifier for a given instance?, and (2) do the recognition rates improve as a result of this?. To that end, the performance of the proposed scheme is assessed using different DCS techniques over 20 public datasets, and the results are compared to a classical pool generation method and a globally generated one.

The comparative study features two performance metrics: the hit rate, which is the rate at which the DCS technique selects the correct classifier in memorization, and the accuracy rate during test. The former indicates whether the DCS techniques have more/less difficulty in selecting the most competent classifier in the pool, whilst the latter indicates if this ability results or not in a better performance during generalization.

1.4 Organization of the Dissertation

This dissertation is organized as follows. In Chapter 2, the main concepts regarding MCS are presented. The Oracle model and the most important DCS techniques are also introduced in the chapter. Then, the characterization of the Oracle for DCS techniques is presented based on a previous work.

With the most important concepts and relationships established in Chapter 2, the proposed method is then introduced in Chapter 3. A step-by-step analysis using a toy problem of the proposed method is also presented in the chapter to illustrate its generation process.

Experiments are conducted in Chapter 4. The results are then evaluated and discussed in the chapter according to the research methodology presented previously.

Finally, the main points presented in this dissertation are summarized in Chapter 5. The conclusions derived from the experimental results are summarized and this work's contributions are outlined. Finally, future works in this are suggested at the end of the chapter.

2

Background

2.1 Introduction

The "No Free Lunch Theorem" (WOLPERT; MACREADY, 1997) leads to the conclusion that there is not a single machine learning algorithm capable of yielding a better performance than all other algorithms for all problems. Since there is no universally superior classifier, an alternative approach to this would be using several classifiers to perform the classification task. In this context, MCS aim to exploit the competence of each classifier in a set, or pool of classifiers, in hopes that the combined response of the pool may outperform each single classifier in it (KITTLER et al., 1998; WOŹNIAK; GRAÑA; CORCHADO, 2014).

MCS have often been shown to yield a greater overall performance in comparison to single classifier approaches (ALKOOT; KITTLER, 1999; OPITZ; MACLIN, 1999; FERNáNDEZ-DELGADO et al., 2014). Several solutions to real world problems were also proposed using MCS, such as recommendation systems (JAHRER; TÖSCHER; LEGENSTEIN, 2010), face recognition (TORRE et al., 2015), intrusion detection (GIACINTO; ROLI; DIDACI, 2003) and credit scoring (XIAO; XIAO; WANG, 2016).

In this chapter, the main concepts of MCS are introduced, with special attention to the ones that are most closely related to this work. In Section 2.2, an overview of the main phases of MCS is presented. The Oracle model, which is an important concept in MCS literature, is also introduced in this section. Since this work is focused on Dynamic Classifier Selection, the main DCS techniques found in the literature are presented in Section 2.3. Afterwards, the performance gap between the Oracle and the DCS techniques, on which this work's proposal is based, is analyzed and discussed in Section 2.4. Finally, the main points presented in this chapter are reviewed and summarized in Section 2.5.

2.2 Overview

Figure 2.1 depicts the three stages an MCS is composed of: (1) Generation, (2) Selection and (3) Aggregation (CRUZ; SABOURIN; CAVALCANTI, 2018). In the generation stage, the training set $\mathscr{T}$ is used as input to an ensemble method, which results in a pool of classifiers $C = \{c_1, c_2, ..., c_M\}$ that contains M classifiers. In the selection stage, a non-empty subset C' of

classifiers from C is selected to perform the classification task. This subset is called an Ensemble of Classifiers (EoC) in MCS literature. The validation set $\mathscr{V}$ is often used in this stage. Moreover, the selection stage is entirely optional, that is, the EoC C' may be identical to the original pool C in an MCS. In the last stage, aggregation, the outputs of all classifiers in C' are combined to obtain the final decision of the system.

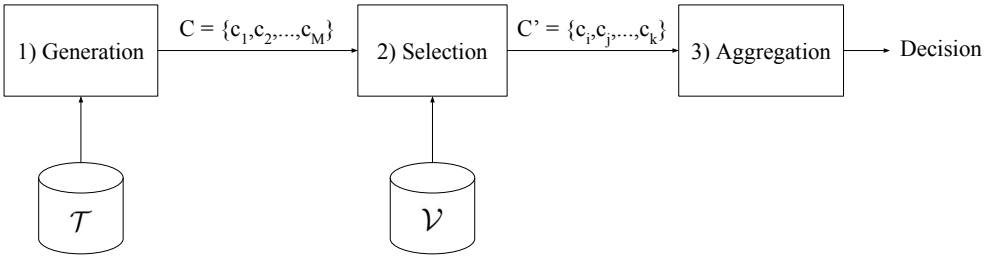


Figure 2.1: Stages of an MCS. In the first stage, a pool of classifiers $C = \{c_1, c_2, ..., c_M\}$ of size M is generated using the training set $\mathscr{T}$. In the second stage, an Ensemble of Classifiers (EoC) $C' \subseteq C$ is selected using the validation set $\mathscr{V}$. In the third stage, the final decision of the system is obtained by aggregating the individual responses of the classifiers in C'.

There are several approaches to all three stages of MCS. In this section, the main aspects regarding each stage is presented. Moreover, the Oracle model, which is later used in this work, is also presented in this section.

2.2.1 Generation

The generation stage of an MCS is responsible for producing a pool of classifiers $C = \{c_1, c_2, ..., c_M\}$ of size M using the training set, as Figure 2.1 shows. The resulting pool C should contain accurate and diverse classifiers. Diversity is the measure of how complementary the classifiers in the pool are, so that a pool that contains classifiers that make different mistakes in different regions of the feature space is more diverse than a pool with classifiers that always respond similarly. Since combining similar classifiers makes little sense, the classifiers in C should be somewhat diverse so that their strengths can contribute to the overall performance of the system (ZHOU, 2012; KUNCHEVA, 2014).

In order to obtain a diverse pool of classifiers, ensemble methods can use one or a few of the following strategies (KUNCHEVA, 2014):

- **Manipulating parameters/initialization:** For classifier models that require parameters or initialization, such as Multi-Layer Perceptron (MLP) neural networks, altering one or the other for each classifier being generated may yield a somewhat diverse pool.

- **Manipulating target labels:** In this case, the classification task may be partitioned, so that each classifier learns a different part. An example of this is the Error Correcting

Output Codes (ECOC) (DIETTERICH; BAKIRI, 1995) ensemble, in which each classifier learns to separate two groups of classes.

- **Using different classifier models:** Each classifier model have inherent characteristics which allows them to perceive the same problem from different perspectives. Thus, an *heterogeneous* pool, which contains classifiers from different models, can present a reasonable degree of diversity.

- **Manipulating training data:** Training each classifier in the pool with a different version of the original training set may yield a quite diverse pool. To achieve this, the training set can be manipulated in two ways: horizontally, in which the training distribution is modified for each classifier, and vertically, in which the alterations are done in the feature space. Classical ensemble methods, such as Bagging (BREIMAN, 1996), Boosting (SCHAPIRE et al., 1997) and Random Subspace (HO, 1998), fall into this category. Bagging and Boosting manipulate the training distribution to construct the pool of classifiers, the former via bootstrap sampling and the latter by assigning weights to each sample and updating them iteratively. Random Subspace, on the other hand, train each classifier with a random subset of the problem's features.

2.2.2 Selection

In the selection stage, the most competent classifier(s) in C is(are) chosen to label the unknown instances. These classifiers form the Ensemble of Classifiers C', as depicted in Figure 2.1, and the choice of classifiers is usually done based on validation data.

The selection of the classifiers in C can be performed statically or dynamically. In SS, the EoC C' is obtained during the training phase, and all test instances are labelled using the same EoC during generalization. The selection in SS techniques is usually based on a selection criteria, such as accuracy and diversity, which is estimated using only validation data (CRUZ; SABOURIN; CAVALCANTI, 2018). Moreover, optimization methods are often used to obtain the static ensemble (PARTALAS; TSOUMAKAS; VLAHAVAS, 2008; DOS SANTOS; SABOURIN; MAUPIN, 2009). Figure 2.2a illustrates this process.

In DS, on the other hand, the EoC C' is obtained during generalization, that is, in the test phase, as Figure 2.2b shows. The idea behind DS is that classifiers are experts in different regions of the feature space (KO; SABOURIN; JR., 2008; BRITTO; SABOURIN; OLIVEIRA, 2014). So, instead of using all classifiers in the original pool C, the DS technique selects from C, for each unknown instance $\mathbf{x_q}$, the most competent classifier(s) in the region where $\mathbf{x_q}$ is located, forming the EoC C'. Then, the selected classifiers are used in the aggregation phase to label this specific query sample $\mathbf{x_q}$.

It has been been shown that using DS techniques is more advantageous in comparison with SS approaches, specially when dealing with problems that possess a high level of uncertainty for lack of enough training data (BRITTO; SABOURIN; OLIVEIRA, 2014). Due to their

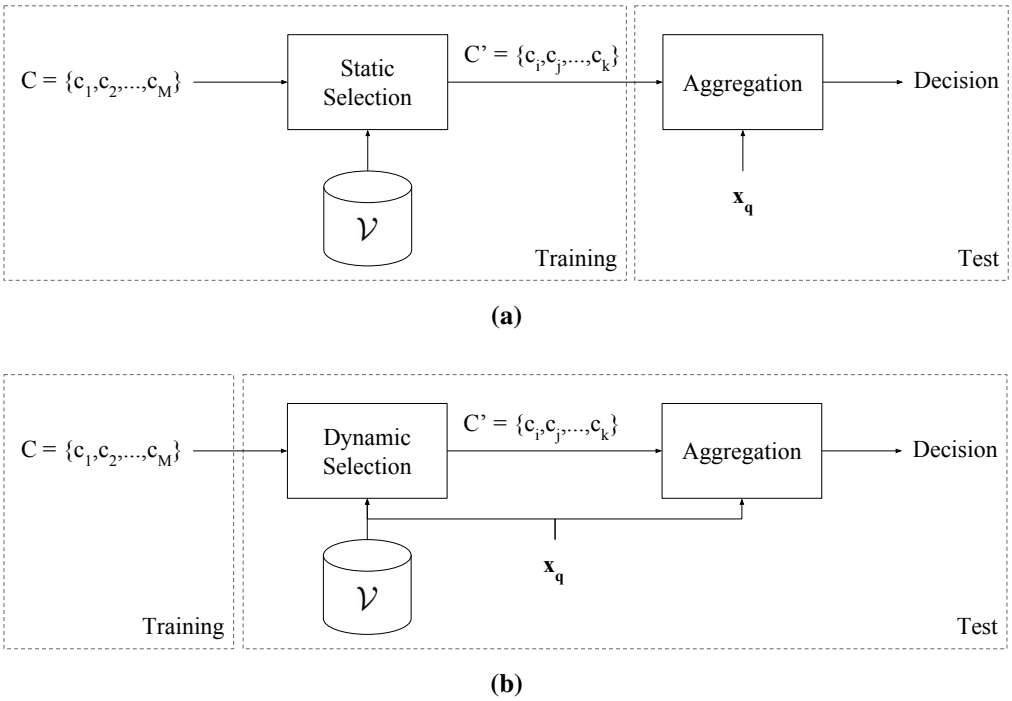


Figure 2.2: Differences between (a) Static Selection and (b) Dynamic Selection. C is the resulting pool of the generation phase, $\mathscr{V}$ is the validation set, $C' \subseteq C$ is the selected Ensemble of Classifiers (EoC) and $\mathbf{x_q}$ is the query sample. In (a), the selection occurs during the training stage. Therefore, the same EoC C' is used to label all query instances in the aggregation stage. In (b), the selection happens during the test stage. Thus, a specific EoC C' is obtained to label each query sample $\mathbf{x_q}$.

importance in this work's motivation and application, a more detailed view on the workings of DS approaches is presented later in this chapter.

2.2.3 Aggregation

In the aggregation phase, the output of the system is obtained by aggregating the responses of all previously selected classifiers in C' (Figure 2.1). The combination rule used in an MCS can be non-trainable, trainable or based on dynamic weighting (CRUZ; SABOURIN; CAVALCANTI, 2018).

Classical examples of non-trainable combiners, which have a fixed combination rule, are the Majority, Sum and Product voting schemes (KITTLER et al., 1998). Trainable combiners use a meta-classifier to learn the combination rule for each specific classification problem, and have been shown to outperform non-trainable combiners in several works (CRUZ; CAVALCANTI; REN, 2010; CRUZ et al., 2015; CRUZ; SABOURIN; CAVALCANTI, 2018).

The dynamic weighting approach, on the other hand, is similar to DS methods, in a way that the competence of each classifier in the original pool C is calculated based on a query sample $\mathbf{x_q}$ (Figure 2.2b). The main difference between both schemes is that the response of all classifiers in C are weighted by their local competence and combined to obtain the output of the system, while in DS techniques only the selected classifiers (C' of Figure 2.2b) have a say in the system's final decision.

2.2.4 The Oracle Model

An important concept related to MCS is the Oracle model. The Oracle, defined in (KUNCHEVA, 2002), is an abstract model in which, for a given test instance, if there is a classifier in the pool that correctly labels this instance, then the entire system also classifies it correctly. The Oracle is often regarded in the literature as a possible upper limit for the performance of MCS, and for that reason, it is widely used to compare performances of different fusion and selection schemes (KUNCHEVA, 2002; DIDACI et al., 2005).

The concept of the Oracle is also used in different areas of MCS. For instance, it is quite related to the construction of pools. As previously stated, diversity is an important aspect in the generation of a pool of classifiers for MCS. Intuitively, a highly diverse pool would have a high Oracle accuracy rate (KUNCHEVA; WHITAKER, 2003; BROWN et al., 2005). Moreover, in (RAUDYS, 2006), the Oracle is also noted as a sort of quality measure for a given pool of classifiers.

Another area in which the Oracle's properties are explored is DS techniques, which select a specific set of classifiers according to each query sample. For instance, the K-Nearest-Oracles Eliminate (KNORA-E) and K-Nearest-Oracles Union (KNORA-U) methods and their variants, introduced in (KO; SABOURIN; SOUZA BRITTO JR, 2007), apply the concept of the Oracle directly by selecting an ensemble formed by the classifiers, called the K nearest Oracles, which correctly classify a given query sample's neighbors. On the other hand, the Random Linear Oracle method (KUNCHEVA; RODRIGUEZ, 2007) uses, instead of single classifiers in the pool, mini ensembles with two classifiers and a random linear function, the latter functioning as an Oracle that selects one of the two classifiers to use for labeling each test instance according to its relative position to the hyperplane.

However, the Oracle model may not be a good reference model in some circumstances. For instance, given a two-class problem and a pool composed of one random classifier and another classifier complimentary to the first one. That is, when one classifier labels as Class 1, the other labels as Class 2, and vice versa. In this case, the Oracle is able to correctly classify any instance at all times. Thus, it does not fit in the Bayesian paradigm (WOŹNIAK; GRAÑA; CORCHADO, 2014). The model was also considered too optimistic to be the upper limit for DS schemes (DIDACI et al., 2005).

2.3 Dynamic Selection

As previously stated, DS schemes aim to single out the most competent classifier(s) to label an individual unknown sample. The general process for obtaining a specific EoC for each query instance can be divided in three phases (CRUZ; SABOURIN; CAVALCANTI, 2018): Region of Competence definition, competence estimation and selection approach.

In the first phase, the local area in which the query sample is located is obtained. This area is called the Region of Competence (RoC) of the query instance. Then, the competence of

each classifier in the query sample's RoC is estimated according to a given competence measure in the second phase. Finally, either the classifier with highest competence level is singled out or an ensemble composed of the most competent classifiers is selected in the last phase. If the former, the selection approach is a Dynamic Classifier Selection (DCS) scheme. If the latter, a Dynamic Ensemble Selection (DES) approach is used. Choosing more than one classifier to label the query instance can be advantageous since the risk of selecting an unsuitable one is distributed in DES schemes (KO; SABOURIN; JR., 2008).

Figure 2.3 illustrates the usual procedure for dynamically selecting classifiers. The query sample $\mathbf{x_q}$ and a set of labelled instances called the dynamic selection dataset (DSEL) are used to define the query sample's RoC (θ_q). The DSEL dataset can be either the training or validation set. Then, the competence of each classifier from the original pool C is estimated over θ_q using a competence measure. The estimated competence of classifier c_i is denoted as δ_i in this image. The *competence vector* $\boldsymbol{\delta}$, which contains the competence estimates from all classifiers in C, is then used in the selection approach, which can be a DCS or a DES method, to obtain the EoC C' to be used in the aggregation stage.

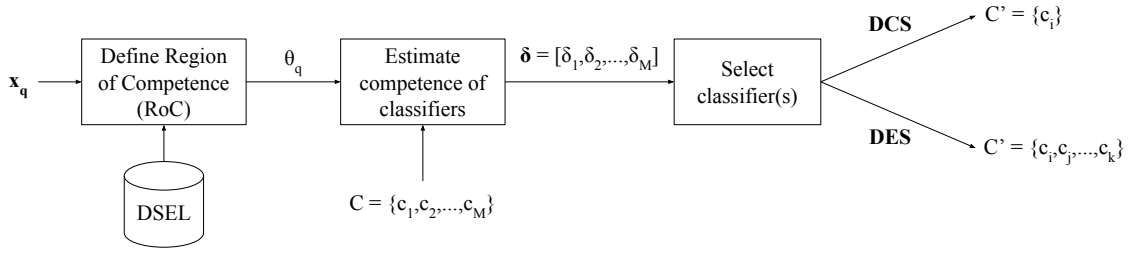


Figure 2.3: Phases of a DS scheme. DSEL is the dynamic selection dataset, which contains labelled samples, $\mathbf{x_q}$ is the query sample, θ_q is the query sample's Region of Competence (RoC), C is the pool produced in the generation phase, $\boldsymbol{\delta}$ is the competence vector composed of the estimated competences δ_i of each classifier c_i and C' is the resulting EoC of the selection phase. If the selection approach is DCS, C' will contain only one classifier from C. Otherwise, the most competent classifiers in C will be chosen to form the EoC.

The RoC definition, which has a significant impact on the performance of DS techniques (DIDACI; GIACINTO, 2004; LIMA; SERGIO; LUDERMIR, 2014; CRUZ; SABOURIN; CAVALCANTI, 2016), can be performed using four main approaches (CRUZ; SABOURIN; CAVALCANTI, 2018):

- **Clustering methods:** in this approach, the DSEL dataset is divided into clusters during the training phase. Then, for all clusters, the competence of each classifier is calculated, and an EoC with the most competent ones for each cluster is defined. During generalization, EoC used to label the query sample is the one assigned to the cluster whose centroid is closest to the test instance. This approach is quite fast since the RoC definition and also the classifiers' competence estimation is done during the training phase. Thus, the only computation it requires for selecting the EoC during generalization is calculating the distance between the query sample and the cluster's

centroids.

- **K-Nearest Neighbors rule:** the RoC is defined as the K-Nearest Neighbors of the query sample in the DSEL dataset. Besides the classical K-NN method for obtaining the query sample's neighborhood, alternative methods, such as the K-Nearest Neighbors Equality (K-NNE) (SIERRA et al., 2011) and adaptive K-NN methods (WANG; NESKOVIC; COOPER, 2007) are also used in DS context. While more computationally expensive than the clustering approach, since the distance between the query sample and all DSEL dataset is required, the local region obtained using a K-NN approach is more precise, which may lead to a better competence estimation of classifiers and thus a more suitable EoC.

- **Similarities in the decision space:** the *decision space*, which is based on the Behavior-Knowledge Space (BKS) (HUANG; SUEN, 1995), is an M-dimensional space in which each dimension corresponds to the decision of one of the M classifiers in the pool. The coordinates of a point in the decision space are then the outputs, whether in predicted labels or class scores, of all classifiers with regards to a given sample. This point is called the *output profile* of that particular instance. In such approaches, the RoC definition is performed by computing the similarity between the query sample's output profile and the the output profiles of the instances in DSEL. The instances whose output profiles are the closest to the query sample's one are selected to form its RoC.

- **Potential functions:** in this approach, the entirety of the DSEL dataset is used as the RoC of each query sample. However, each instance in DSEL has a weight assigned to it, so that samples far from the test instance have less influence in the classifiers' competence estimation than instances closer to it. Each instance's weight is calculated using a potential function, usually Gaussian, with the distance between itself and query sample input. Though removing the need to set a neighborhood size, as the K-NN approaches do, using the whole DSEL set for competence estimation is quite costly in computational terms.

The estimation of a classifier's competence, on the other hand, can be based on numerous criteria, such as Accuracy (WOODS; KEGELMEYER JR; BOWYER, 1997), Ranking (SABOURIN et al., 1993), Probabilistic (GIACINTO; ROLI, 1999), Behavior (KURZYNSKI; TRAJDOS, 2017), Oracle (KO; SABOURIN; SOUZA BRITTO JR, 2007), Data complexity (BRUN et al., 2016), Meta-learning (CRUZ et al., 2015), Diversity (DOS SANTOS; SABOURIN; MAUPIN, 2009), Data handling (XIAO et al., 2012) and Ambiguity (DOS SANTOS; SABOURIN; MAUPIN, 2008). In Section 2.3.1, a few of these approaches are touched upon in the context of DCS techniques.

2.3.1 DCS Techniques

Since the focus of this work is on pool generation for DCS techniques, the most relevant ones, according to a recent survey on dynamic selection of classifiers (CRUZ; SABOURIN; CAVALCANTI, 2018), are introduced in this section. The notation used in this section is the same as shown in Figure 2.3. Moreover, all DCS methods presented in this section use a K-NN rule with neighborhood size K in order to define the query sample's RoC θ_q. Therefore, θ_q is always composed of K instances from DSEL, that is, $\theta_q = \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_K}\}$.

Overall Local Accuracy

In the Overall Local Accuracy (OLA) method (WOODS; KEGELMEYER JR; BOWYER, 1997), the competence estimation of the classifiers is based on accuracy. The competence δ_i of classifier c_i is simply defined as the percentage of instances that it correctly classifies in the RoC θ_q. The classifier that correctly classifies most instances in θ_q will have a highest competence estimate, and will thus be selected by the method.

Local Class Accuracy

As with OLA, the selection criterion of the Local Class Accuracy (LCA) method (WOODS; KEGELMEYER JR; BOWYER, 1997) is also accuracy. However, LCA also considers the label ω_l assigned by classifier c_i to the query sample $\mathbf{x_q}$ in the competence estimation of c_i. The competence δ_i of classifier c_i is then defined as the percentage of correctly classified instances among the ones in θ_q that belong to class ω_l.

Modified Local Accuracy

The Modified Local Accuracy (MLA) method (SMITS, 2002) also estimates the competence δ_i of a classifier c_i based on accuracy, though, unlike OLA and LCA, it takes into account the distance of each instance $\mathbf{x_k} \in \theta_q$ to the query sample $\mathbf{x_q}$. The calculation of δ_i is described in Equation 2.1, in which ω_l is the label c_i attributed to $\mathbf{x_q}$ and $W_k = 1/d_{q,k}$ is the inverse of the Euclidean distance between $\mathbf{x_q}$ and $\mathbf{x_k}$.

$$\delta_i = \frac{1}{K} \sum_{\mathbf{x_k} \in \omega_l} W_k \tag{2.1}$$

It is important to note that the competence estimation in MLA disregard whether c_i correctly classifies $\mathbf{x_k}$ or not. As long as it assigns to $\mathbf{x_k}$ the same class as it did to $\mathbf{x_q}$, the former will contribute to its competence level.

Modified Classifier Rank

Based on ranking, the Modified Classifier Rank method (SABOURIN et al., 1993; WOODS; KEGELMEYER JR; BOWYER, 1997) assigns the competence δ_i of classifier c_i as the number of consecutive instances $\mathbf{x_k} \in \theta_q$ it correctly classifies. Then, the classifier with highest competence estimate is considered to be the highest in the “rank" of classifiers, and is thus selected to label the query sample $\mathbf{x_q}$.

Multiple Classifier Behavior

The Multiple Classifier Behavior (MCB) method is based on two selection criteria: accuracy and behavior. The competence estimation of a classifier c_i by the method is done as follows.

Firstly, the output profile, that is, the responses of all classifiers with regards to each instance $\mathbf{x_k} \in \theta_q$ is obtained. Then, the query sample's output profile and its similarity to all output profiles from θ_q are also calculated. The similarity is defined as the proportion of equal corresponding coordinate values between the output profile of $\mathbf{x_q}$ and the output profile of $\mathbf{x_k}$. All instances with similarities below a certain threshold are removed from θ_q. Finally, the competence δ_i of classifier c_i is calculated as the proportion of instances it correctly classifies from the remaining ones in θ_q.

If the difference between the competences of the most competent classifier and all other classifiers in the pool is above a second threshold, this classifier is considered to be significantly superior to the rest and is thus selected. Otherwise, all classifiers are combined using majority voting to label the query sample.

A Priori

The competence of a classifier c_i in the A Priori method (GIACINTO; ROLI, 1999) is estimated using the class posterior probability of c_i on the query sample's RoC θ_q, without taking into account the label ω_l the classifier assigns to the query instance $\mathbf{x_q}$. The calculation of the level of competence δ_i by the A Priori method is described in Equation 2.2, in which $P(\omega|\mathbf{x_k} \in \omega, c_i)$ is the posterior probability of the correct classification by c_i of the sample $\mathbf{x_k} \in \theta_q$, and $W_k = 1/d_{q,k}$ is the inverse of the Euclidean distance between $\mathbf{x_q}$ and $\mathbf{x_k}$.

$$\delta_i = \frac{\sum_{k=1}^{K} P(\omega|\mathbf{x_k} \in \omega, c_i) W_k}{\sum_{k=1}^{K} W_k} \tag{2.2}$$

Similarly to the MCB method, the A Priori method selects the most competent classifier only if its competence is significantly superior in comparison to the other classifiers. If not, all classifiers are combined using majority voting.

A Posteriori

The A Posteriori method (GIACINTO; ROLI, 1999) is quite similar to the A Priori method, in that it also uses the posterior probability of correct classification of a given classifier to estimate its competence. However, the method also considers the label ω_l the classifier c_i gives to the query sample $\mathbf{x_q}$ in the calculation of the classifier's level of competence δ_i (Equation 2.3).

$$\delta_i = \frac{\sum_{\mathbf{x_k} \in \omega_l} P(\omega_l | \mathbf{x_k}, c_i) W_k}{\sum_{k=1}^{K} P(\omega_l | \mathbf{x_k}, c_i) W_k} \tag{2.3}$$

In the A Posteriori method, as in the A Priori and MCB methods, if the classifier with highest competence level does not surpass all others by a given threshold, the final decision of the system is given by majority voting of all classifiers. Otherwise, the most competent classifier is selected and used to label $\mathbf{x_q}$.

2.4 Oracle-DCS Performance Gap

For DCS techniques, in which only one classifier is selected to label a query instance, the Oracle model simulates the perfect selection scheme by identifying the best expert for each particular test sample. The Oracle accuracy rate is, therefore, the theoretical limit for such techniques. That way, the model can measure how close a DCS technique is from its maximum performance and indicates whether there is still room for improvements in classification accuracy.

However, it has been shown that there is a significant performance gap between DS schemes and the Oracle (DIDACI et al., 2005; KO; SABOURIN; SOUZA BRITTO JR, 2007). For instance, in (CRUZ et al., 2015), it was shown the accuracy rate of the Oracle was almost 20 percentile points greater than the accuracy of some DCS techniques, for a pool of 100 Perceptrons generated using Bagging. In other words, the Oracle stated that, among those 100 classifiers, there was at least one that could correctly classify these 20 percentile points of the query samples, but the DCS techniques were not able to select any competent classifier in the pool for these instances.

Based on that observation, the reasons why the Oracle can display undesired behavior when used as a guide to generate a pool of classifiers for DCS schemes were investigated in (SOUZA et al., 2017). To that end, a supervised method for producing a pool of classifiers that guarantees an Oracle accuracy rate of 100% in the training set was introduced. This generation method, called the Self-Generating Hyperplanes (SGH), assures the presence of at least one competent classifier in the pool for each training sample. The behavior of the DCS techniques, when the theoretical limit for the training set is maximum, was then analyzed, and based on that analysis the relationship between the Oracle model and DCS techniques was derived.

In this section, a summary on the characterization of the Oracle for DCS is presented. Section 2.4.1 introduces the SGH method and its properties. This method was used to investigate whether the use of the Oracle as a guide to generate a pool of classifiers for DCS techniques

was advantageous, given that the presence of at least one competent classifier for each training instance is guaranteed in this scenario. Moreover, the SGH method is also used in the proposed technique due to its properties. In Section 2.4.2, a brief analysis on the relationship between the Oracle and DCS techniques in regards to pool generation is then presented based on the results from the previous work.

2.4.1 The Self-Generating Hyperplanes Method

A simplified pseudocode of the SGH method is presented in Algorithm 1. For a more in depth view on the SGH method, see Appendix A. The method consists of generating hyperplanes iteratively in such a way that each instance in the training set must be correctly classified by at least one of the base classifiers in the pool, that is, the Oracle for the training dataset is 100%. The base classifiers chosen to produce such hyperplanes were Perceptrons, due to their weakness. Using weak base classifiers can provide more differences between the DS techniques (KO; SABOURIN; JR., 2008), and hence a better comparison between them. Furthermore, to speed up the training process, the method uses a heuristic to find the Perceptrons' weights without explicitly training them.

The input to the SGH method is only the training set $\mathscr{T}$, and its output is the generated pool of classifiers (C). In each iteration (Step 3 to Step 15), the centroids of all classes in $\mathscr{T}$ are obtained in Step 4 and stored in $\mathscr{R}$. The two centroids in $\mathscr{R}$ most distant from each other, $\mathbf{r_i}$ and $\mathbf{r_j}$, are selected in Step 5. Then, a hyperplane c_m is placed between $\mathbf{r_i}$ and $\mathbf{r_j}$, dividing both points halfway from each other. The linear classifier c_m is then tested over the training set, and the instances it correctly labels are removed from $\mathscr{T}$ (Step 7 to Step 12). Then, c_m is added to C in Step 13, and the loop is repeated until $\mathscr{T}$ is completely empty. That is, the SGH method only stops generating hyperplanes when all training instances are correctly labelled by at least one classifier in C, i.e., the Oracle accuracy rate for the training set is 100%.

Apart from guaranteeing a theoretical limit of 100% in the training set, the SGH presents other advantages in comparison with classical ensemble methods such as Bagging, Random Subspace, and Boosting. It automatically defines the pool size according to the training data, so it does not require the pool size to be set beforehand as these methods do. Also, the heuristic used to train the classifiers makes the training much faster than in these methods.

It is also important to note that the algorithm is strictly deterministic, that is, it will always generate the same pool given the same input (training set). Also, the generated classifiers are simple, two-classes Perceptrons.

2.4.2 Oracle-DCS Analysis

The performance gap between the Oracle model and the DCS techniques was analyzed in (SOUZA et al., 2017) over 20 public datasets. The accuracy rates of different DCS techniques were obtained using four state-of-the-art methods: OLA, LCA, MLA, and MCB. These tech-

Algorithm 1 General procedure of the Self-generating Hyperplanes (SGH) method.

Input: $\mathscr{T} = \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_N}\}$ ▷ Training dataset
Output: C ▷ Final pool
$C \leftarrow \{\}$ ▷ Pool initially empty
$m = 1$ ▷ Classifier count
while $\mathscr{T} \neq \{\}$ **do**
  $\mathscr{R} \leftarrow getCentroids(\mathscr{T})$ ▷ Calculate each class' centroid
  $\mathbf{r_i}, \mathbf{r_j} \leftarrow selectCentroids(\mathscr{R})$ ▷ Select the most distant centroids
  $c_m \leftarrow placeHyperplane(\mathbf{r_i}, \mathbf{r_j})$ ▷ Generate hyperplane between centroids $\mathbf{r_i}$ and $\mathbf{r_j}$
  for every $\mathbf{x_n}$ in $\mathscr{T}$ **do**
    $\omega \leftarrow c_m(\mathbf{x_n})$ ▷ Test c_m over training instance
    if $\omega = y_n$ **then**
      $\mathscr{T} \leftarrow \mathscr{T} - \{\mathbf{x_n}\}$ ▷ Remove from $\mathscr{T}$ correctly classified instance
    end if
  end for
  $C \leftarrow C \cup \{c_m\}$ ▷ Add c_m to pool
  $m = m + 1$
end while
return C

niques were chosen due to their superior performance in comparison with other DCS techniques in (BRITTO; SABOURIN; OLIVEIRA, 2014). The RoC size K for each of the DCS techniques is set to 7, for they also performed the best with this configuration in this survey.

The gap between the Oracle and the DCS techniques was evaluated using Bagging, for it is a classical ensemble method used in several DS works (CRUZ et al., 2015; CRUZ; SABOURIN; CAVALCANTI, 2015), and the SGH method, since it guarantees an Oracle accuracy rate of 100% for the training data, which allows further analysis to be performed.

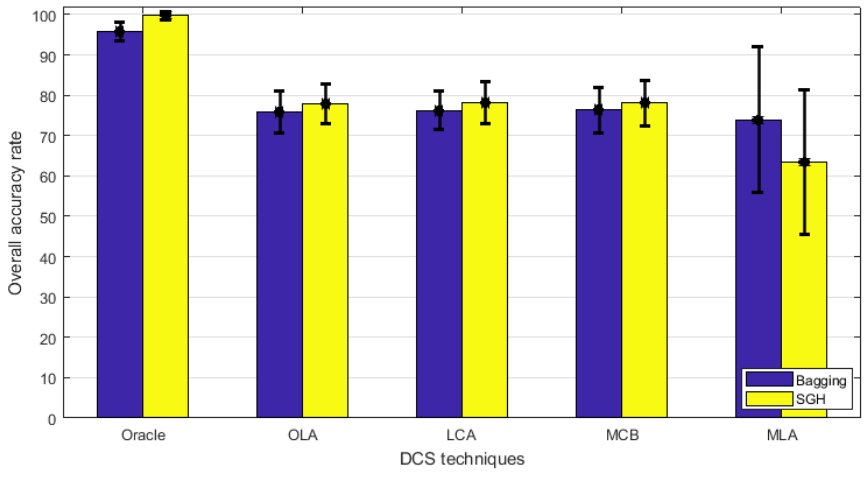


Figure 2.4: Mean and standard deviation of the accuracy rate of the Oracle model and the DCS techniques for Bagging-generated pool of 100 Perceptrons and a pool generated using the SGH method.

Figure 2.4 shows the mean accuracy rate of the Oracle model and the four analyzed DCS techniques using a Bagging-generated pool of 100 Perceptrons and a pool generated using the SGH method over all datasets. The mean pool size for the SGH method was $N = 3.80$ Perceptrons.

It can be observed that, though the SGH method obtained similar accuracy rates to a pool of size 100 generated by Bagging with considerably fewer classifiers for three of the four DCS techniques, it performed quite poorly for MLA. This was due to the local competence estimation in MLA, which is the sum of all the distances between the neighbors and the test instance, given that the classifier labels them as of the same class (Equation 2.1). Therefore, if a classifier labels a test instance and all its neighbors of the same class, it will be deemed more competent than another classifier that labels only part of the neighborhood as of the same class, regardless of the actual local accuracy rate of them both. Since MLA does not filter out non-applicable classifiers, that is, classifiers that do not recognize any of a test instance's neighbors, and the SGH generates only two-class classifiers, the selection, especially for multi-class problems, became based solely on the distance weighting, for usually each classifier in the pool labels most instances in the local region as of one of the two classes it recognizes.

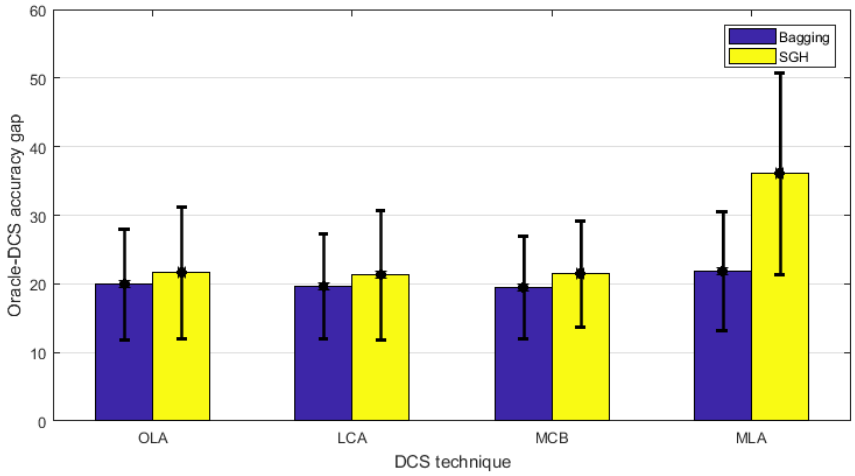


Figure 2.5: Mean and standard deviation of the performance gap between the accuracy rates of the Oracle model and the DCS techniques for Bagging-generated pool of 100 Perceptrons and a pool generated using the SGH method.

It can also be observed that, even though the SGH yielded a significant increase in the Oracle accuracy rate compared to Bagging, the accuracy rate of the DCS techniques did not follow this behavior. That is, the Oracle-DCS performance gap was not narrowed, as Figure 2.5 shows. Thus, this suggests that using the Oracle as a guide to generate the pool of classifiers do not necessarily improve the accuracy rate during the test, since guaranteeing an Oracle accuracy rate of 100% in the training set and almost 100% in the test set does not imply the DCS technique will be able to select the right classifier in the pool.

A further investigation was performed on the behavior of the DCS techniques with regards to the Oracle. Figure 2.6 shows the hit rate of the four DCS techniques, that is, the rate at which these methods selected the correct classifier, and their accuracy rate for the training data as well. The correct classifier of an instance in this context is the one that correctly predicted the label of this instance in the generation phase (step 10 of Algorithm 1). The hit rate was obtained by using the training set as test set with the pool generated by the SGH as input to the

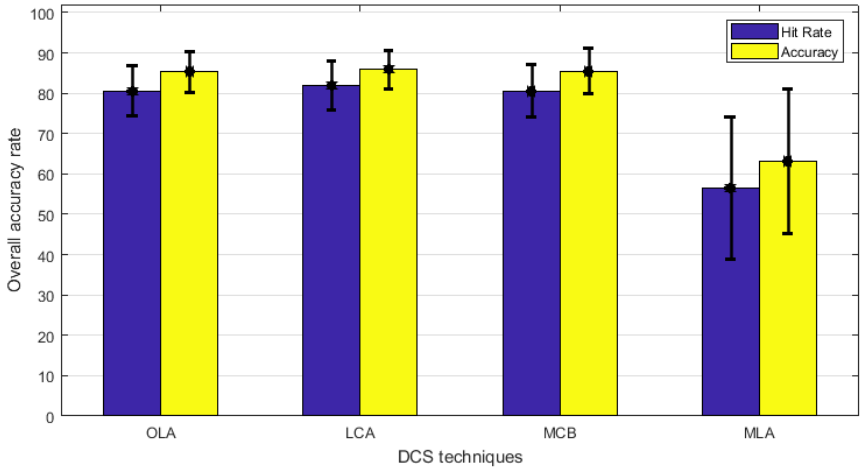


Figure 2.6: Mean and standard deviation of the hit rate and of the memorization accuracy rate of the DCS techniques using a pool generated by the SGH method.

DCS techniques. It is important to remember that the SGH guarantees an Oracle of 100%, in other words, each instance in the training set has at least one classifier that correctly classifies it. So, the hit rate presents a correlation with the classification accuracy in memorization of DCS techniques. It can be observed that, even though the Oracle for the training dataset is always 100%, the hit rate of the DCS techniques is in most cases more than 10 percentile points behind the model, which means that, although the presence of the correct classifier in the pool is guaranteed, the DCS techniques were not able to easily select it.

This shows a significant gap between the theoretical limit and the rate at which the DCS techniques actually select the most competent classifier in the pool, suggesting that the Oracle, as intuitive as it may be, is not the best guide for dynamically selecting a classifier. The reason behind this is that the Oracle perceives the classification problem *globally*, whereas DCS techniques rely on *local* information to select the best classifier for each test instance. Thus, the Oracle information as a guide to generating pool of classifiers may not be much relevant for dynamic selection schemes.

On the other hand, the hit rate captures in a better way the relationship between the theoretical limit and the actual recognition rate because, since the Oracle and the DCS techniques observe the problem from different perspectives, the hit rate measures the intersection between these two perspectives. Therefore, the hit rate may be a better guide than the Oracle in the process of generating a pool of classifiers.

2.5 Conclusion

In this chapter, an overall view on the main aspects regarding MCS was presented. The three stages of an MCS were introduced in Section 2.2, with a brief description on the many possible approaches for each one of them. The Oracle model and its importance in MCS literature were also presented.

In Section 2.3, the general process of dynamically selecting a set of classifiers was further explained. The main approaches for obtaining the RoC and estimating the competence of a classifier were introduced as well. Since this work focus on DCS techniques, the most relevant ones were also presented.

Section 2.4 presented a short analysis on the behavior of the Oracle in regards to DCS techniques. It was shown that the DCS techniques struggle to select the most competent classifier in the pool, even when its presence is assured by the Oracle. This analysis was accomplished with the help of the SGH method, which always generates a pool with an Oracle accuracy rate of 100%. It was concluded that, since the Oracle is performed globally, it may be unsuited to be used as a guide to generate a pool of classifiers for DCS techniques, which consider only its small RoC to estimate the competence and select the best classifier in the pool for a given query sample.

3

The Proposed Method

3.1 Introduction

It was shown in (SOUZA et al., 2017) (Section 2.4), that the Oracle model, which is performed over the entire dataset, was not the best guide in the search for a good pool for DCS. It was argued that this is due to the fact that the DCS techniques rely only on a small region, an instance's neighborhood, in order to select the most competent classifier for this instance. Therefore, difference in the perspectives of the Oracle model and the DCS techniques do not favor the use of the model in generation methods in the DCS context.

With that in mind, it is proposed the use of an Oracle-guided generation method on a local scope, so that the model's properties may be explored by the DCS techniques. The idea is to use a local pool (LP) consisted of specialized classifiers, each of which selected using a DCS technique from a local subpool that contains at least one competent classifier for each instance in class overlap regions of the feature space. A given test instance is in a class overlap area in this context if its RoC is composed of samples from more than one class. If the unknown instance's Region of Competence (RoC) is located in a region with class overlap, the LP is generated on the fly using its neighboring instances and then used to label the query sample. However, if the query instance is far from the classes' borders, no pool is generated and the output label is obtained using the K-NN rule.

The reasoning behind the proposed approach is that using locally generated classifiers for instances in class overlap areas may be of help to the DCS techniques due to their high accuracy in these regions. Moreover, most works regarding DCS use classical generation methods, which were designed for SS techniques (CRUZ; SABOURIN; CAVALCANTI, 2018) and therefore do not take into account the regional aspect of the competence estimation performed by the DS techniques. Thus, matching the perspectives of the generation and the selection stages may be advantageous for these techniques.

The proposed method is presented in more detail in this chapter. In Section 3.2, an overview of the proposed method is presented. A step-by-step analysis of the proposed method is then performed using a 2D toy problem in Section 3.3. Finally, the main aspects regarding the proposed technique are summarized in Section 3.4.

3.2 Overview

The proposed method can be divided into three phases:

1. The *RoC evaluation* phase, in which the query instance's RoC is obtained and evaluated in order to identify if the test sample is in an overlap area of the feature space. If the region contains instances from two or more classes, the local pool (*LP*) is generated in the *local pool generation* phase and later used to label the query sample. If not, the query sample is directly labelled by the K-NN rule in the *generalization* phase.

2. The *local pool generation* phase, in which *LP* is generated. The pool size *M* of *LP* is an input parameter of the method. Each classifier in LP is obtained iteratively by using the SGH method over the query sample's increasing neighborhood. In each iteration, the SGH method yields a subpool with an Oracle accuracy rate of 100% over that neighborhood. Then, a DCS technique is used to select the most competent classifier in the generated subpool, adding it to LP.

3. The *generalization* phase, in which the query sample is labelled by the K-NN rule, if all its neighbors share the same label, or by the majority voting of the classifiers in LP, otherwise.

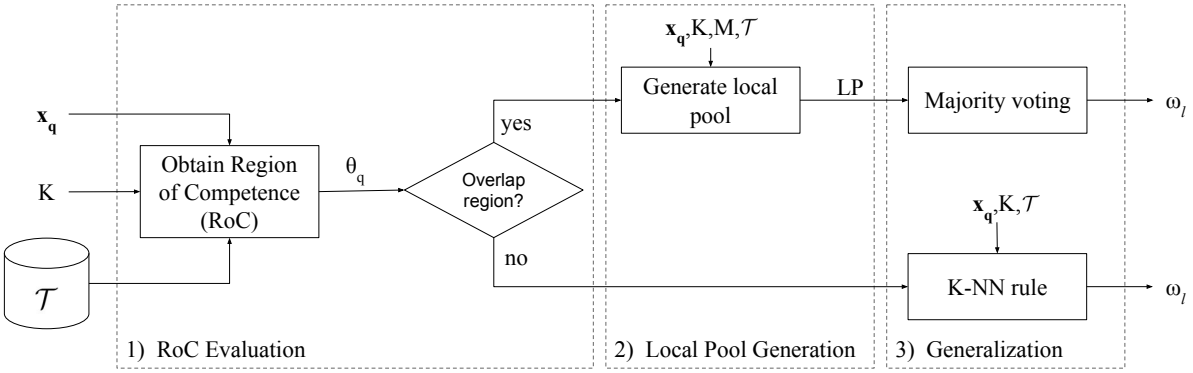


Figure 3.1: Overview of the proposed technique. $\mathcal{T}$ is the training set, $\mathbf{x_q}$ is the query sample, θ_q is its Region of Competence (RoC), K is the size of θ_q, LP is the local pool, M is the pool size of LP and ω_l is the output label of $\mathbf{x_q}$. In the first phase, θ_q is obtained and evaluated. If it only contains samples from the same class, the K-NN rule is used to label $\mathbf{x_q}$ in the third phase. Otherwise, the local pool is generated in the second phase, and $\mathbf{x_q}$ is labelled via majority voting of the classifiers in LP in the third phase.

An overview of the proposed method's three phases is depicted in Figure 3.1, in which $\mathcal{T}$ is the training set, $\mathbf{x_q}$ is the query sample, θ_q is the query sample's Region of Competence (RoC), K is the size of θ_q, LP is the local pool containing M classifiers and ω_l is the output label of the query sample. In the RoC evaluation phase, the K nearest neighbors in the training set $\mathcal{T}$ of the query sample $\mathbf{x_q}$ are selected to form the query sample's RoC θ_q. The DSEL dataset

was not used in the proposed method because the SGH method did not present overfitting when used for RoC definition (SOUZA et al., 2017). Then, the instances that compose θ_q are analyzed. If all of them belong to the same class, the method skips the local pool generation and goes directly to the generalization phase. The output class ω_l of $\mathbf{x_q}$ is then obtained using the K-NN rule with parameter K. However, if there are two or more instances in θ_q that belong to different classes, the query sample's RoC is considered to be in a class overlap region. Thus, the local pool LP containing M classifiers is generated in the second phase and used to label $\mathbf{x_q}$ in the generalization phase via majority voting. The local pool generation phase is explained next.

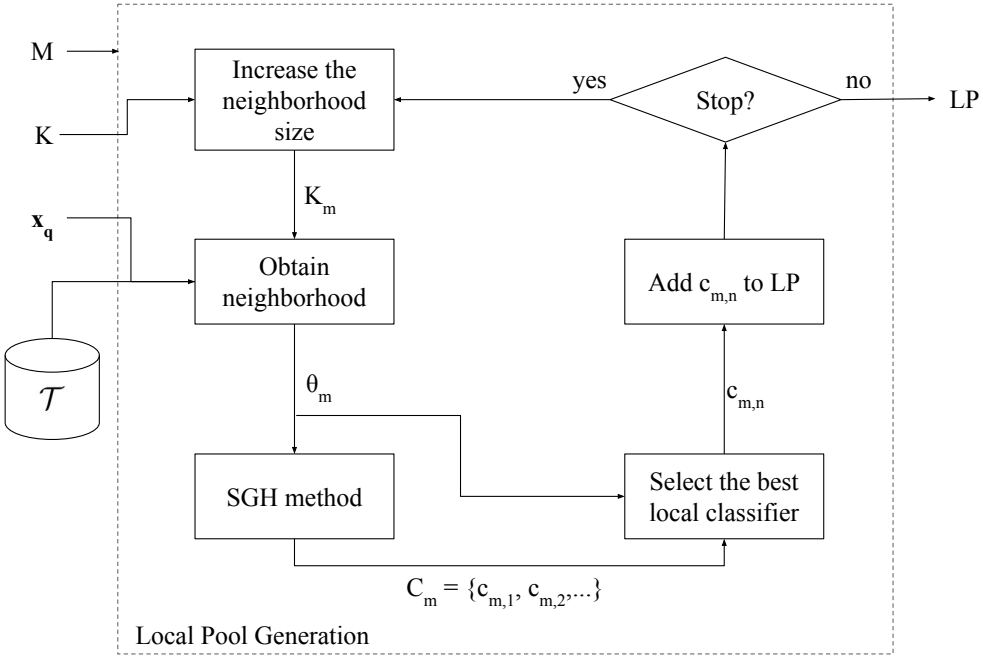


Figure 3.2: Local pool generation phase. The inputs to the generation scheme are the training set $\mathscr{T}$, the query sample $\mathbf{x_q}$, the size K of the query sample's RoC and the local pool size M. The output is the local pool LP. In the m-th iteration, the query sample's neighborhood θ_m of size K_m is obtained and used as input to the SGH method, which yields the subpool C_m. The classifiers from C_m are then evaluated over θ_m using a DCS technique. The classifiers' notation refers a classifier $c_{m,k}$ as the k-th classifier from the m-th subpool (C_m). The most competent classifier $c_{m,n}$ in subpool C_m is then selected and added to the local pool LP. This process is then repeated until LP contains M locally accurate classifiers.

Figure 3.2 shows the generation procedure of the local pool LP. The pool size M of the local pool is an input parameter. The other inputs are the training set ($\mathscr{T}$), the query sample ($\mathbf{x_q}$) and the query sample's RoC size (K). The LP is constructed iteratively. In the m-th iteration, the query sample's neighboring instances in the training set are obtained using any nearest neighbors method, with parameter K_m. These neighboring instances form the query sample's neighborhood θ_m, which is used as input to the SGH method. The SGH method then returns a local subpool C_m that fully covers the neighborhood θ_m. That is, the presence of at least one competent classifier $c_{m,k} \in C_m$ for each instance in θ_m is guaranteed. The indexes in the classifiers' notation indicates that the classifier $c_{m,k}$ is the k-th classifier from the m-th subpool (C_m). Then, the most competent

classifier $c_{m,n}$ from C_m in the region delimited by the neighborhood θ_q is selected by a DCS technique and added to the local pool. The same procedure is performed in iteration $m+1$ with the neighborhood size K_{m+1} increased by 2. This process is then repeated until the local pool contains M locally accurate classifiers.

Algorithm 2 Pseudocode of the proposed method.

Input: $\mathscr{T}, \mathbf{x_q}$ ▷ Training dataset and query instance
Input: K, M ▷ RoC size and pool size of local pool LP
Output: ω_l ▷ Output label of $\mathbf{x_q}$

1: $\theta_q \leftarrow obtainRoC(\mathbf{x_q}, K, \mathscr{T})$ ▷ Obtain the query instance's RoC — 1) RoC Evaluation
2: **if** $\{\exists \mathbf{x_i}, \mathbf{x_j} \in \theta_q | \omega_i \neq \omega_j\}$ **then** — 1) RoC Evaluation
3: $LP \leftarrow \{\}$ ▷ Local pool initially empty — 2) Local Pool Generation
4: **for** every m in $\{1, 2, ..., M\}$ **do** — 2) Local Pool Generation
5: $K_m \leftarrow K + 2 \times (m-1)$ ▷ Increase neighborhood size by 2 — 2) Local Pool Generation
6: $\theta_m \leftarrow obtainNeighborhood(\mathbf{x_q}, K_m, \mathscr{T})$ ▷ Obtain neighborhood of $\mathbf{x_q}$ — 2) Local Pool Generation
7: $C_m \leftarrow generatePool(\theta_m)$ ▷ Generate local subpool C_m — 2) Local Pool Generation
8: $c_{m,n} \leftarrow selectClassifier(\mathbf{x_q}, \theta_m, C_m)$ ▷ Select best classifier in C_m — 2) Local Pool Generation
9: $LP \leftarrow LP \cup \{c_{m,n}\}$ ▷ Add $c_{m,n}$ to LP — 2) Local Pool Generation
10: **end for** — 2) Local Pool Generation
11: $\omega_l \leftarrow majorityVoting(\mathbf{x_q}, LP)$ ▷ Label $\mathbf{x_i}$ with majority voting on LP — 3) Generalization
12: **else** — 3) Generalization
13: $\omega_l \leftarrow KNN(\mathbf{x_q}, K, \mathscr{T})$ ▷ Label query sample using K-NN rule — 3) Generalization
14: **end if** — 3) Generalization
15: **return** ω_l

The proposed method's pseudocode is shown in Algorithm 2. Its inputs are the training set $\mathscr{T}$, the query sample $\mathbf{x_q}$, the RoC size K and the local pool size M. In Step 1, the query sample's RoC θ_q is obtained by selecting the K closest samples to $\mathbf{x_q}$ in the training set. The RoC is then evaluated in Step 2. If all instances in θ_q belong to the same class, the method goes straight to Step 13 and the query sample's output label ω_l is obtained using the K-NN rule with parameter K and returned in Step 15.

However, if there are two instances $\mathbf{x_i}$ and $\mathbf{x_j}$ from θ_q that possess different labels ω_i and ω_j, then the method proceeds to Step 3. Each classifier in the local pool LP is obtained in the loop that iterates M times (Step 4 to Step 10). In each iteration, the neighborhood size K_m is calculated in Step 5. Then, the query sample's neighborhood θ_m is obtained using a nearest neighbors method in Step 6. The subpool C_m is then generated in Step 7 using the SGH method with θ_m as training set. In Step 8, a DCS technique is then used to select the most competent classifier $c_{m,n}$ in C_m. The classifier $c_{m,n}$ is added to LP in Step 9, and then the loop continues until the local pool is complete. Finally, the query sample's label ω_l is obtained using majority voting over the locally accurate classifiers in LP and returned in Step 15.

3.3 Step-by-step Analysis

In order to better understand the generation process by the proposed technique, the latter was executed over a 2D toy problem dataset. The P2 Problem (VALENTINI, 2005) was chosen

for its complex borders, and noise was added to the original problem by randomly changing the labels of the samples near the class borders. The toy problem used in this analysis contains 1000 instances, 75% of which were used for training and the rest for test. The P2 Problem training set used as input to the method is depicted in Figure 3.3, with its theoretical decision boundaries in grey. The pool size of the local pool for this demonstration was set to $M = 7$, and the RoC size K was set to 7. The method used for selecting the query instance's neighborhood in the local pool generation phase (Step 6 of Algorithm 2) for this example was the regular K-NN rule, and the DCS technique used to select the most competent classifier (Step 8 of Algorithm 2) was OLA.

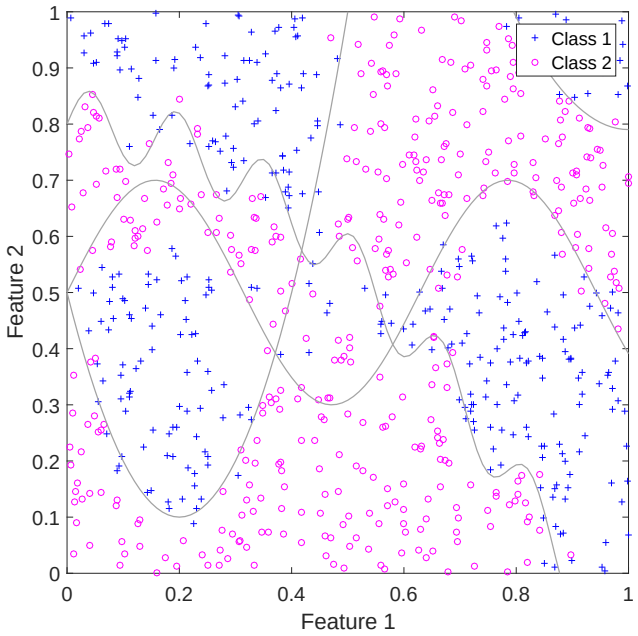


Figure 3.3: P2 Problem training dataset, with theoretical decision boundaries in grey.

Two scenarios of the proposed scheme can be observed in Figure 3.4. In Figure 3.4a, the input query instance $\mathbf{x_q}$ of Algorithm 2 belongs to Class 2. The query sample's RoC θ_q is obtained selecting its k-nearest neighbors over the training set $\mathscr{T}$ in Step 1. In this case, since all instances in θ_q belong to the same class, Class 2, $\mathbf{x_q}$ is not located in an overlap region of the feature space. Therefore, the procedure goes to Step 13, in order to obtain the output label ω_l of $\mathbf{x_q}$ using the K-NN rule over the training set with parameter K. Then, the query sample's label is returned in Step 15. In this case $\omega_l = 2$ since all K neighbors of $\mathbf{x_q}$ belong to this class.

In the second scenario, shown in Figure 3.4b, the query instance $\mathbf{x_q}$ of Algorithm 2 belongs to Class 1. Its RoC θ_q is obtained in Step 1, with more than half of its instances belonging to the opposite class. Thus, a simple K-NN rule would misclassify this query sample. The query instance's RoC θ_q is then analyzed in Step 2. Since there are instances in θ_q belonging to different classes, the region the query sample is located is identified as a class overlap area and the local pool LP will be generated and used from this step forward. Starting with an empty set (Step 3), each iteration from Step 4 to Step 10 adds a single classifier to LP.

In the first iteration, the neighborhood size K_1 is set to 7 in Step 5, and then the K_1

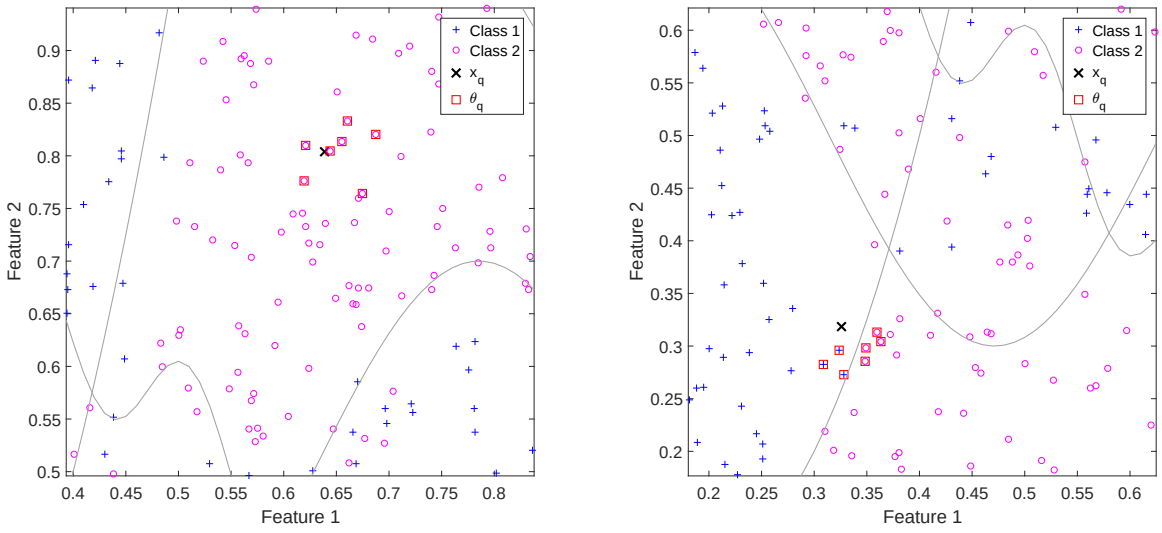


Figure 3.4: Two different scenarios of the proposed method. In (a), the query instance $\mathbf{x_q}$ belongs to Class 2. Since all instances in its neighborhood θ_q belong to the same class, the K-NN rule is used to label $\mathbf{x_q}$. On the other hand, the query sample's neighborhood θ_q in (b) contains both classes. Thus, the local pool *LP* will label the query instance $\mathbf{x_q}$, which belongs to Class 1.

nearest neighbors of $\mathbf{x_q}$ are selected to compose the query sample's neighborhood θ_1 in Step 6. The local subpool C_1 is then generated using θ_1 as the input dataset to the SGH method. The resulting pool, which guarantees an Oracle accuracy rate of 100% in θ_1, is shown in Figure 3.5a, containing only one classifier, $c_{1,1}$. Since there is only one classifier in C_1, $c_{1,1}$ is selected to compose *LP* in Step 8 and Step 9.

In the second iteration, the neighborhood parameter is increased by 2 in Step 5, and the resulting neighborhood θ_2 contains $K_2 = 9$ instances, as shown in Figure 3.5b. Then, the local subpool C_2 is generated in Step 7, with θ_2 as the input parameter of the SGH method. Since only one classifier was able to deliver an Oracle accuracy rate of 100% over θ_2, the resulting pool contains only $c_{2,1}$, which is selected in Step 8 to be added to *LP* in Step 9.

The neighborhood θ_3, obtained in Step 6 of the third iteration, contains $K_3 = 11$ instances, as Figure 3.5c shows. C_3 is then generated in Step 7 so that it fully covers θ_3, resulting in only one classifier ($c_{3,1}$), which is later added to *LP* in Step 9.

The fourth local subpool C_4, depicted in Figure 3.5d, is generated in Step 7 of the fourth iteration, with neighborhood θ_4 of size $K_4 = 13$ as input to the SGH method. The only classifier generated, $c_{4,1}$, is then added to *LP* in Step 9.

In the fifth iteration, the neighborhood θ_5 is obtained with parameter $K_5 = 15$ in Step 6. In Step 7, the SGH method yields the local subpool C_5, depicted in Figure 3.5e. Afterwards, the single classifier $c_{5,1}$ in C_5 is added to *LP*.

The neighborhood θ_6 of the sixth iteration is obtained with $K_6 = 17$ in Step 6. Then, the local subpool C_6 is generated in Step 7, resulting in two classifiers, $c_{6,1}$ and $c_{6,2}$, as shown in Figure 3.5f. In Step 8, both classifiers are evaluated over θ_6 using a DCS technique, OLA in this

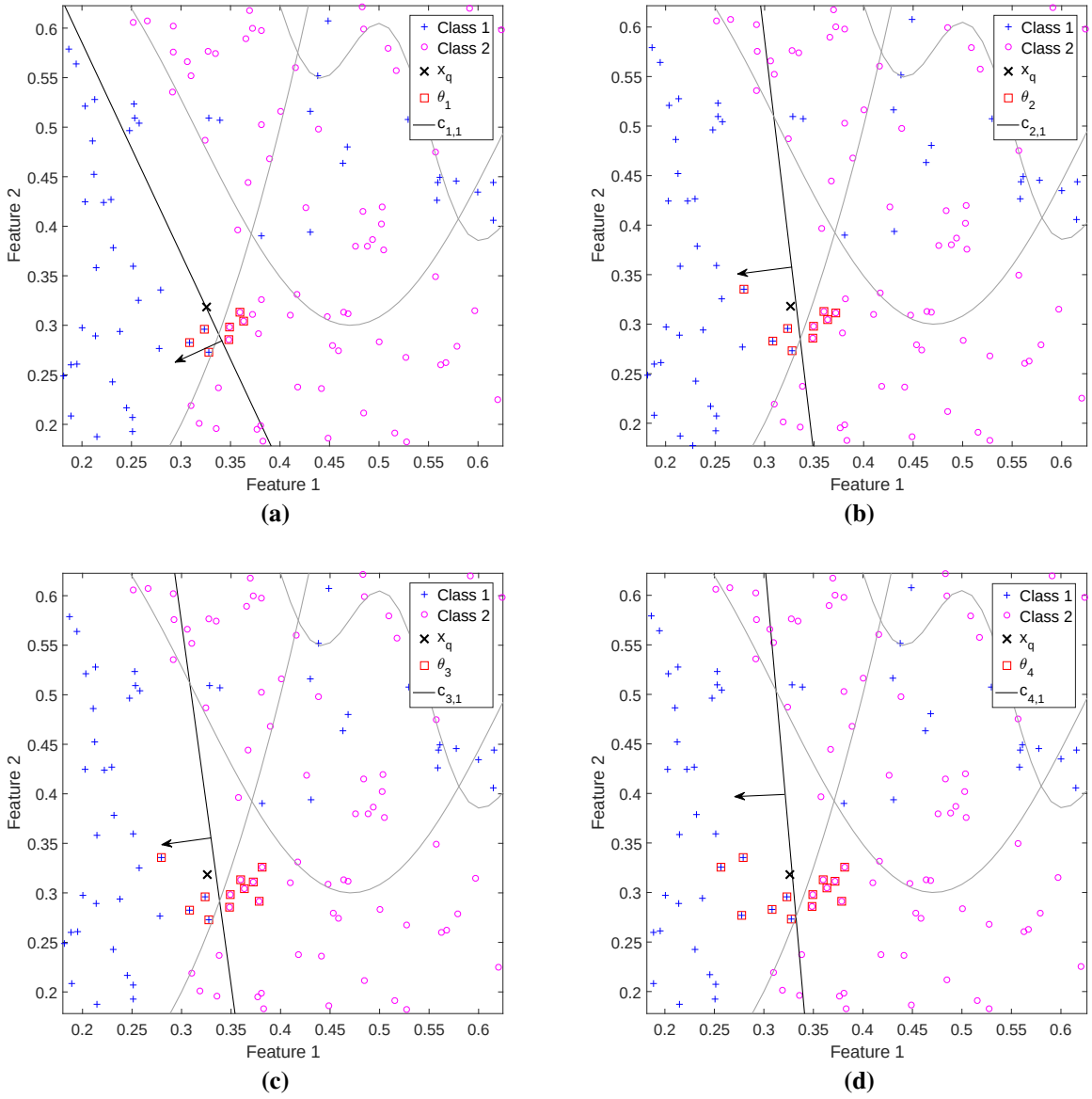


Figure 3.5: Local pool generation. (a) First, (b) second, (c) third, (d) fourth, (e) fifth, (f) sixth and (g) seventh iteration of the method, with its respective neighborhoods (θ_m) and generated local subpools C_m formed by the depicted classifiers ($c_{m,k}$). The arrows indicate in which part of the feature space the classifiers label as Class 1. Each local subpool C_m is obtained using the SGH method with its respective neighborhood θ_m, which increases in each iteration, as input. The final local pool LP, formed by the best classifiers in each subpool C_m, is shown in (h).

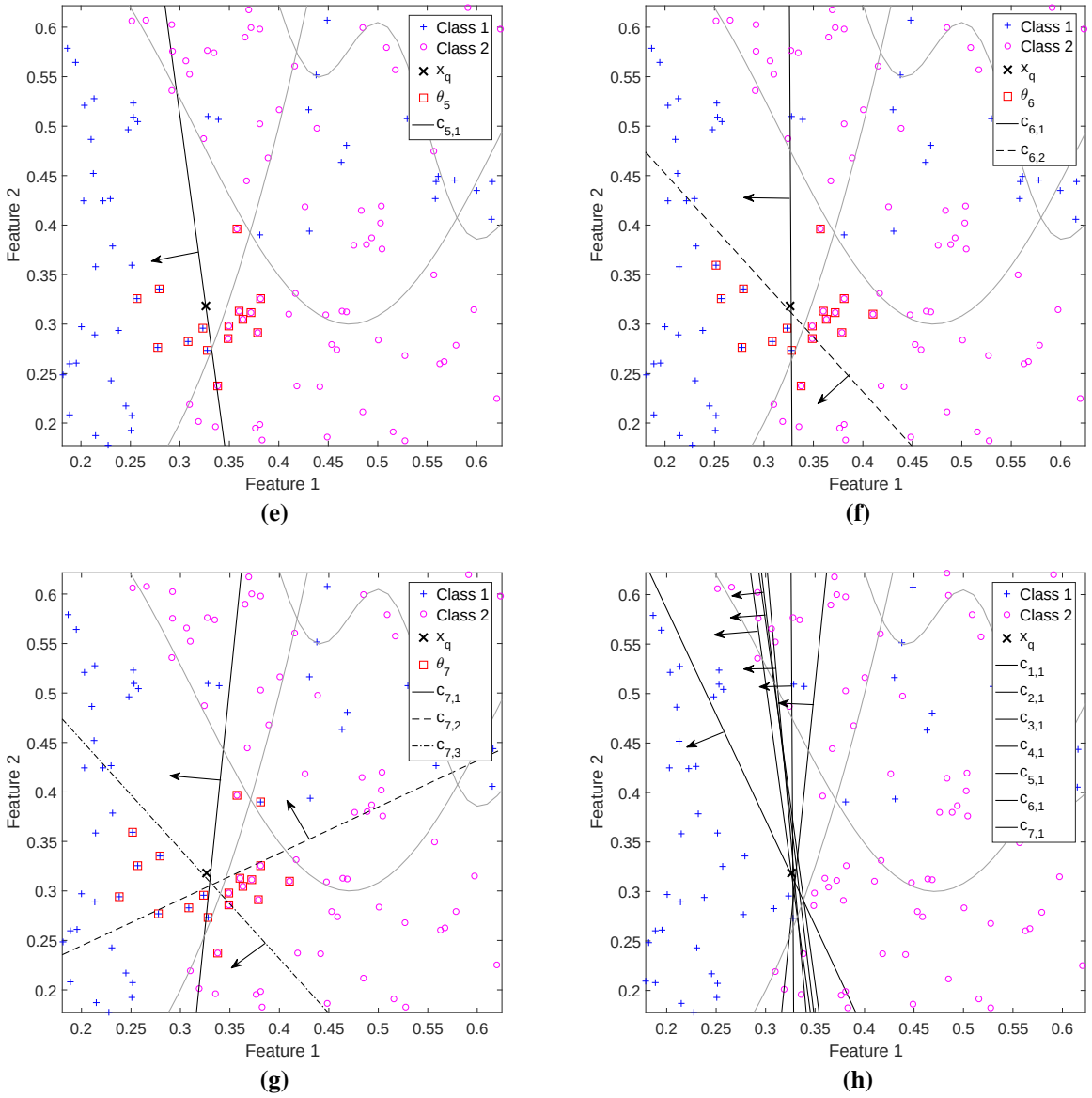


Figure 3.5: Local pool generation. (a) First, (b) second, (c) third, (d) fourth, (e) fifth, (f) sixth and (g) seventh iteration of the method, with its respective neighborhoods (θ_m) and generated local subpools C_m formed by the depicted classifiers ($c_{m,k}$). The arrows indicate in which part of the feature space the classifiers label as Class 1. Each local subpool C_m is obtained using the SGH method with its respective neighborhood θ_m, which increases in each iteration, as input. The final local pool LP, formed by the best classifiers in each subpool C_m, is shown in (h).

Table 3.1: Majority voting of the classifiers from *LP* for the query instance from Figure 3.4b.

	$c_{1,1}$	$c_{2,1}$	$c_{3,1}$	$c_{4,1}$	$c_{5,1}$	$c_{6,1}$	$c_{7,1}$	**Total**
Class 1		x	x	x		x	x	5
Class 2	x				x			2

case. The most accurate one ($c_{6,1}$) in C_6 is returned and added to *LP* in Step 9.

In the last iteration, the local subpool C_7 is generated in Step 7 using the neighborhood θ_7 with $K_7 = 19$ instances. Then, the local subpool C_7 is generated, yielding three classifiers that fully cover θ_7. Each classifier in C_7, shown in Figure 3.5g, is then evaluated using OLA, and the one that performs best over θ_7 is selected. The selected classifier, $c_{7,1}$ in this case, is then added to the local pool, completing the generation process of *LP*, depicted in Figure 3.5h.

After the generation process of the local pool, each classifier in it labels the query instance $\mathbf{x_q}$, and the final label is obtained by majority vote in Step 11. Table 3.1 shows the vote of each classifier in *LP*. The final label returned in Step 11 by the local pool is $\omega_l = 1$, which is the true class of $\mathbf{x_q}$.

3.4 Conclusion

In this chapter, an online pool generation method for DCS techniques was presented. The proposed technique generates classifiers on a local scope, so that the DCS techniques may be able to select the most competent one for a given instance more frequently, since the generation and the selection are performed from the same perspective.

The proposed technique generates a pool comprised of locally accurate classifiers each time a query sample is located in a class overlap region of the feature space. In such cases, the local pool is obtained sequentially. In each iteration of the generation scheme, a local subpool is generated over the query sample's neighborhood, and the most competent classifier in the former is selected by a DCS technique to compose the final local pool. The latter is then used to label the query sample using majority voting. For instances located far from the classes' borders, a simple K-NN rule is applied instead.

An overview of the proposed method was presented in Section 3.2. The three phases of the method were introduced and the pseudocode of the proposed scheme was presented and explained in detail. In addition to that, the proposed method's workings was further illustrated in a step-by-step analysis using a toy problem in Section 3.3.

4

Experiments

4.1 Introduction

In Chapter 2, the main concepts of MCS were presented, with special attention to the area of DCS. It was also shown that the DCS techniques have difficulty in selecting the best classifier even when the pool contains it, as the Oracle indicates. Then, in Chapter 3, an online local pool generation method was proposed in order to aid the selection performed by these techniques. The idea behind it was that, by using locally generated classifiers, it could be easier for the DCS techniques to select the best among them, since the generation and the selection perspectives match in this case.

In this chapter, an experimental analysis on the proposed method is performed. The aim of these experiments is to observe whether the DCS techniques are more prone to selecting the best classifier in the pool when said pool is generated locally and whether the use of such pools increase classification rates, in comparison to globally generated pools.

This chapter is organized as follows. In Section 4.2, the experimental protocol for the experimental analysis is outlined. Section 4.3 presents information regarding the proposed method's operation, as well as an analysis on the sensitivity of the proposed method's parameter. Lastly, a comparative study on the performance of DCS techniques using locally generated pools, by means of the proposed technique, and globally generated pools is done in Section 4.4. The main conclusions derived from the experiments are then summarized in Section 4.5.

4.2 Experimental Protocol

Experiments were conducted over the 20 datasets described in Table 4.1. All of them are public datasets. Eleven from the UCI machine learning repository (BACHE; LICHMAN, 2013), three from the Ludmila Kuncheva Collection (KUNCHEVA, 2004) of real medical data, three from the STATLOG project (KING; FENG; SUTHERLAND, 1995), two from the Knowledge Extraction based on Evolutionary Learning (KEEL) repository (ALCALÁ et al., 2011) and one from the Enhanced Learning for Evolutive Neural Architectures (ELENA) project (JUTTEN, 2002).

The DCS techniques chosen to evaluate the methods in the experiments were OLA,

Table 4.1: Main characteristics of the datasets used in the experiments.

Dataset	No. of Instances	No. of Features	No. of Classes	Class Sizes	Source
Adult	48842	14	2	383;307	UCI
Blood Transfusion	748	4	2	570;178	UCI
Cardiotocography (CTG)	2126	21	3	1655;295;176	UCI
Steel Plate Faults	1941	27	7	158;190;391;72;55;402;673	UCI
German credit	1000	20	2	700;300	STATLOG
Glass	214	9	6	70;76;17;13;9;29	UCI
Haberman's Survival	306	3	2	225;81	UCI
Heart	270	13	2	150;120	STATLOG
Ionosphere	315	34	2	126;225	UCI
Laryngeal1	213	16	2	81;132	LKC
Laryngeal3	353	16	3	53;218;82	LKC
Liver Disorders	345	6	2	145;200	UCI
Mammographic	961	5	2	427;403	KEEL
Monk2	4322	6	2	204;228	KEEL
Phoneme	5404	6	2	3818;1586	ELENA
Pima	768	8	2	500;268	UCI
Sonar	208	60	2	97;111	UCI
Vehicle	846	18	4	199;212;217;218	STATLOG
Vertebral Column	310	6	2	204;96	UCI
Weaning	302	17	2	151;151	LKC

LCA and MCB, since they outperformed the other evaluated DCS techniques in (BRITTO; SABOURIN; OLIVEIRA, 2014) and have no issue with two-class classifiers, as MLA does (Section 2.4.2). The RoC size K for each of the DCS techniques is set to 7, since it yielded the best results in (BRITTO; SABOURIN; OLIVEIRA, 2014). Moreover, the proposed method was tested with different parameter configurations, with the best overall configuration featuring in the comparative study, while the baseline generation method (Bagging) was tested with the pool size set to 100 classifiers, as it is often done in DS works (CRUZ et al., 2015; CRUZ; SABOURIN; CAVALCANTI, 2015).

All configurations were evaluated using 20 replications of each dataset from Table 4.1. For the configurations that used pools generated by the SGH method, each replication was randomly split into two parts: 75% for training and 25% for test. Since the SGH method did not present overfitting, both in global (SOUZA et al., 2017) and local scope, it was chosen not to use a dynamic selection validation set (DSEL). In the comparative study, however, two configurations use a pool of 100 classifiers obtained using Bagging. For these configurations, the DSEL was used in order to avoid overfitting, and each replication was thus randomly split: 50% for training, 25% for validation and the remaining 25% for test.

4.3 Proposed Method Analysis

The proposed method was evaluated over the test set, and the results regarding its operation are presented and analyzed in this section in three parts. In Section 4.3.1, an evaluation on the method's frequency at identifying an instance in class overlap areas is presented. Moreover, Section 4.3.2 presents an analysis on the parameter sensitivity of the proposed method.

4.3.1 RoC Evaluation

The mean percentage of test instances truly located in class overlap regions is depicted in the *True* bars of Figure 4.1 for all datasets. This percentage was obtained observing the neighborhood of each test instance over the entire dataset. The mean percentage of test instances the proposed method identified as in a class overlap region is also depicted in Figure 4.1 (*Estimated* bars). That is, the *Estimated* bars show the frequency at which the proposed method generated and used the local pool, whilst the *True* bars show the actual proportion of instances close to the class borders for each problem. It can be observed that, though the proportion of instances in overlap regions varies greatly from problem to problem, the proposed method was mostly able to identify in which cases the query sample was truly located in one and thus generated a local pool to handle them.

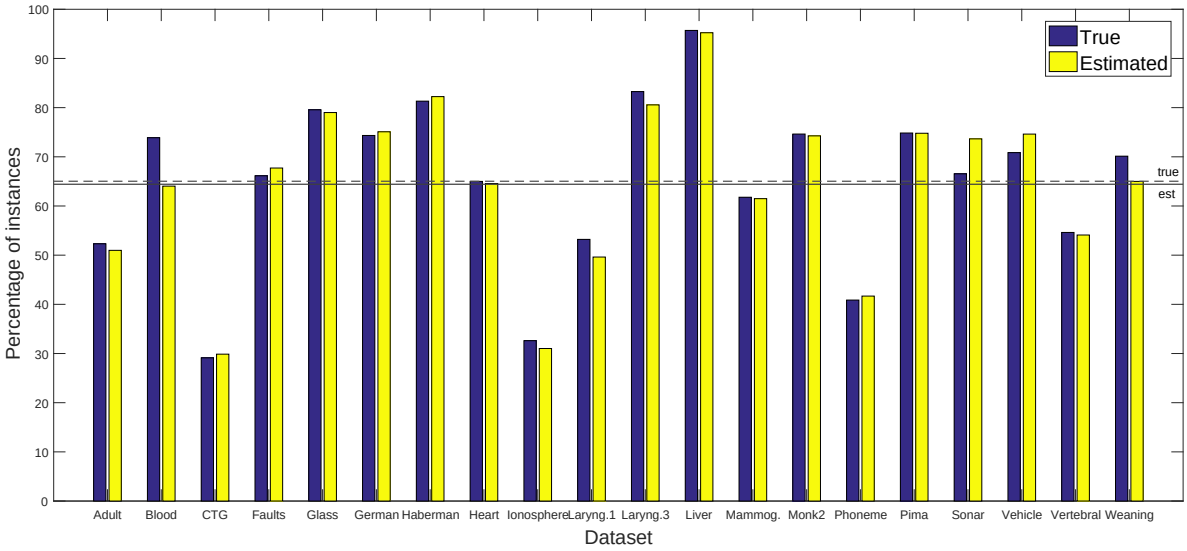


Figure 4.1: Mean percentage of test instances in overlap regions for all datasets from Table 4.1. The *Estimated* bar indicates the times the local pool was used to classify an instance, while the *True* bar indicates the true percentage of test instances in overlap regions considering the entire dataser. The lines $true$ and *est* indicate the averaged values of all datasets for the estimated and true percentage of test instances, respectively.

The averaged value of the true and estimated percentage of instances in those regions is also indicated in Figure 4.1 by the $true$ and *est* lines, respectively. It can be observed that the mean percentage of test instances truly located near the borders was 65.03%, while the proposed method generated local pools for 64.48% of the test instances, on average.

4.3.2 Parameter Sensitivity

The proposed method was evaluated with different parameter configurations in order to analyze the parameters' effect on performance. Two neighborhood acquisition methods ($obtainNeighborhood()$ from Step 6 of Algorithm 2) were tested: the regular K-NN rule, and a a version of the K-NNE method, in which the returned neighborhood contains an equal amount

Table 4.2: Mean and standard deviation of the accuracy rate of the proposed technique using (a) OLA, (b) LCA and (c) MCB. The local pool in configuration LP_m uses K-NN in the generation process and contains $M = m$ classifiers and are grouped together. Configurations referenced as LP_m^e are also grouped and use K-NNE to generate $M = m$ classifiers. The row *Avg rank* shows the resulting mean ranks of a Friedman test with $alpha = 0.05$ on each group. Best results are in bold.

(a)

Dataset	LP_1	LP_3	LP_5	LP_7	LP_1^e	LP_3^e	LP_5^e	LP_7^e
Adult	84.68 (3.27)	84.36 (3.09)	**84.80 (2.85)**	83.61 (3.69)	86.10 (2.14)	86.99 (2.91)	87.69 (2.87)	**87.75 (2.17)**
Blood	70.08 (2.71)	72.02 (1.69)	**72.50 (1.53)**	74.02 (2.69)	75.72 (2.57)	76.84 (0.80)	76.68 (1.50)	**77.31 (1.61)**
CTG	91.80 (0.61)	**91.95 (0.70)**	91.89 (1.01)	92.20 (1.10)	**91.22 (0.34)**	90.74 (0.28)	90.42 (0.44)	89.98 (0.80)
Faults	71.74 (0.94)	72.37 (1.74)	**72.65 (1.69)**	72.40 (1.29)	**68.91 (1.07)**	67.30 (1.39)	66.21 (1.04)	65.93 (1.29)
German	71.20 (2.59)	72.32 (2.63)	**72.96 (2.81)**	72.24 (2.02)	72.94 (1.49)	74.28 (2.12)	**74.76 (1.87)**	74.08 (1.79)
Glass	69.43 (2.97)	69.25 (3.41)	**69.62 (4.93)**	68.11 (3.87)	**67.45 (2.44)**	64.81 (5.28)	63.87 (4.79)	62.36 (2.84)
Haberman	68.95 (3.90)	**69.80 (3.81)**	69.67 (5.02)	69.61 (4.07)	68.95 (2.02)	69.80 (2.07)	70.99 (2.32)	**72.83 (2.64)**
Heart	81.25 (3.37)	80.29 (4.54)	81.54 (5.06)	**82.06 (4.86)**	80.37 (4.06)	83.53 (3.29)	**83.68 (3.66)**	**83.68 (3.27)**
Ionosphere	88.81 (2.28)	89.32 (2.69)	90.45 (2.03)	**91.59 (1.82)**	89.72 (1.30)	90.74 (1.53)	91.02 (1.42)	**91.31 (1.66)**
Laryngeal1	78.40 (5.69)	**80.00 (4.91)**	78.87 (5.56)	77.74 (5.53)	**80.94 (3.62)**	79.81 (5.41)	80.57 (5.13)	80.57 (5.87)
Laryngeal3	68.99 (3.01)	69.83 (2.56)	71.74 (1.90)	**72.13 (2.69)**	66.57 (0.72)	**67.42 (1.71)**	66.69 (1.38)	65.96 (1.42)
Liver	58.95 (4.12)	60.87 (5.15)	**61.22 (5.99)**	60.06 (5.04)	61.92 (3.52)	63.26 (3.34)	64.01 (3.03)	**67.50 (1.53)**
Mammographic	75.70 (2.50)	76.47 (3.36)	**76.51 (2.56)**	76.32 (2.54)	81.80 (2.15)	82.26 (1.88)	**82.38 (2.32)**	**82.38 (1.98)**
Monk2	94.12 (1.68)	**95.93 (1.14)**	95.05 (0.81)	94.91 (0.97)	**95.19 (1.46)**	94.44 (1.12)	94.17 (0.74)	94.07 (0.76)
Phoneme	**89.03 (0.65)**	88.81 (0.37)	88.76 (0.45)	88.65 (0.55)	87.30 (0.25)	**87.36 (0.31)**	87.06 (0.41)	86.95 (0.55)
Pima	70.52 (1.80)	71.30 (1.71)	**72.42 (2.11)**	72.08 (1.75)	73.07 (1.39)	74.90 (1.26)	76.15 (1.38)	**76.82 (2.21)**
Sonar	83.56 (3.50)	82.88 (6.51)	**84.23 (4.57)**	83.46 (5.66)	**78.85 (4.99)**	77.69 (4.06)	76.73 (3.47)	75.19 (4.09)
Vehicle	72.10 (1.61)	73.56 (2.11)	73.61 (2.27)	**74.15 (2.08)**	72.76 (2.13)	**72.81 (2.02)**	72.08 (1.84)	70.73 (1.72)
Vertebral	80.64 (4.24)	80.83 (3.42)	83.33 (2.32)	**85.06 (2.13)**	84.68 (3.47)	86.15 (3.26)	**87.44 (3.57)**	86.47 (2.65)
Weaning	84.14 (2.43)	85.86 (2.56)	**86.32 (1.73)**	**86.32 (1.83)**	**86.32 (2.19)**	86.05 (2.19)	85.66 (1.86)	**86.32 (2.43)**
Average	77.70	78.40	**78.91**	78.84	78.54	78.86	**78.91**	**78.91**
Avg rank	3.45	2.55	**1.675**	2.325	2.725	2.4	**2.375**	2.5
p-value	2.16×10^{-4}				0.81			

of instances from all classes, given that these classes are present in the query instance's RoC θ_q (Step 1 of Algorithm2). Moreover, the parameter M was varied in the set $\{1,3,5,7\}$ for each of the two variations of the method. The configurations that use K-NN and $M = m$ are referenced as LP_m, while the ones with K-NNE and $M = m$ are referenced as LP_m^e in the experiments.

The mean accuracy of each configuration can be observed in Table 4.2 for OLA, LCA and MCB. The results are grouped by neighborhood acquisition method and DCS technique, and the best values from each group are in bold. Moreover, the rows *Avg rank* and *p-value* show the result of a Friedman test with a significance level of $\alpha = 0.5$ on each group.

It can be observed from Table 4.2 that the best pool size value for a given problem and a given neighborhood method varies little in regards to the DCS techniques. However, two main aspects stand out in the results. The first is related to the neighborhood acquisition method. It can be observed that on most multi-class datasets the use of K-NN greatly outperforms the configurations that use K-NNE. The reason for this is further discussed afterwards.

The second aspect regards the best value of the pool size, which is clearly problem dependent. It can be observed, for instance, that the recognition rate increase for the "Liver" problem from configuration LP_5^e to configuration LP_7^e with the addition of only two classifiers in the pool. The opposite occurs for the "Sonar" problem from configuration LP_1^e to configuration LP_3^e. Thus, fine tuning of the method's parameters is required in order to obtain the best performance for each individual problem.

(b)

Dataset	LP_1	LP_3	LP_5	LP_7	LP^e_1	LP^e_3	LP^e_5	LP^e_7
Adult	85.58 (3.26)	85.29 (3.41)	**85.84 (3.52)**	84.65 (3.69)	85.81 (2.73)	86.59 (3.34)	**87.37 (3.39)**	87.11 (2.40)
Blood	72.39 (2.64)	75.56 (1.99)	77.21 (2.02)	**77.93 (2.50)**	75.24 (2.21)	76.57 (0.84)	76.49 (1.49)	**76.94 (1.67)**
CTG	91.95 (0.49)	92.05 (0.68)	**92.23 (0.98)**	92.22 (1.10)	**91.25 (0.39)**	91.00 (0.56)	90.69 (0.39)	90.58 (0.39)
Faults	72.73 (1.20)	73.30 (1.42)	**73.64 (1.67)**	73.20 (1.22)	**69.10 (1.15)**	68.22 (1.49)	66.85 (1.35)	66.28 (1.15)
German	71.30 (2.68)	72.22 (2.65)	**73.20 (2.55)**	72.86 (2.32)	72.94 (1.29)	74.32 (1.99)	**74.88 (2.04)**	74.12 (1.75)
Glass	70.38 (4.02)	**70.75 (3.15)**	70.28 (3.62)	66.98 (3.03)	66.42 (2.49)	69.06 (4.03)	62.92 (2.14)	**63.58 (3.47)**
Haberman	70.20 (3.12)	**71.12 (2.72)**	70.59 (3.50)	70.79 (3.74)	67.63 (2.11)	69.67 (2.72)	71.12 (2.20)	**72.70 (2.52)**
Heart	81.76 (4.06)	80.29 (4.54)	82.06 (5.39)	**82.57 (5.42)**	80.00 (3.83)	83.46 (3.44)	**83.68 (3.66)**	83.09 (3.32)
Ionosphere	88.86 (2.17)	89.94 (1.82)	91.08 (1.62)	**91.36 (1.24)**	90.11 (1.28)	91.48 (1.82)	91.59 (1.89)	**91.82 (2.01)**
Laryngeal1	79.06 (5.77)	**79.62 (4.96)**	**79.62 (5.60)**	79.25 (5.05)	80.38 (4.08)	79.62 (5.53)	**80.57 (5.13)**	**80.57 (5.87)**
Laryngeal3	69.55 (3.36)	70.28 (2.86)	72.92 (2.39)	**73.54 (2.35)**	66.46 (1.83)	67.19 (1.44)	66.80 (1.18)	**67.81 (1.72)**
Liver	58.95 (3.99)	60.99 (5.12)	**62.09 (6.42)**	62.03 (5.44)	62.56 (3.34)	63.37 (3.30)	63.90 (2.93)	**67.09 (1.69)**
Mammographic	77.62 (2.76)	79.52 (2.19)	**80.10 (2.02)**	**80.10 (2.02)**	81.75 (2.75)	82.31 (2.16)	82.38 (2.58)	**82.40 (2.01)**
Monk2	94.07 (1.60)	**95.74 (1.22)**	95.00 (0.87)	94.91 (0.97)	**95.14 (1.47)**	94.40 (1.14)	94.12 (0.75)	94.07 (0.76)
Phoneme	**89.60 (0.41)**	89.22 (0.36)	89.13 (0.56)	89.15 (0.48)	**87.37 (0.29)**	87.35 (0.36)	87.06 (0.46)	86.97 (0.53)
Pima	71.04 (1.98)	72.08 (1.74)	**73.54 (1.69)**	73.41 (0.98)	72.94 (1.72)	74.74 (1.20)	76.04 (1.26)	**76.80 (2.19)**
Sonar	83.37 (3.38)	82.79 (6.35)	**84.13 (4.41)**	83.27 (5.30)	**80.00 (3.33)**	78.08 (3.44)	76.92 (3.18)	76.54 (3.72)
Vehicle	72.74 (1.42)	72.36 (0.96)	**73.92 (2.39)**	73.47 (1.66)	**73.14 (1.91)**	73.09 (1.77)	72.36 (1.59)	72.12 (1.15)
Vertebral	81.86 (3.46)	82.50 (2.61)	83.46 (2.66)	**85.38 (2.64)**	85.38 (3.36)	86.15 (3.26)	**87.44 (3.57)**	86.41 (2.71)
Weaning	84.08 (2.37)	85.92 (2.49)	86.32 (1.73)	**86.45 (2.01)**	**86.45 (2.26)**	86.05 (2.19)	85.66 (1.86)	86.32 (2.43)
Average	78.35	79.07	**79.82**	79.68	78.50	79.14	78.94	**79.17**
Avg rank	3.45	2.675	**1.75**	2.125	2.8	2.45	**2.375**	**2.375**
p-value	1.84×10^{-4}				0.68			

(c)

Dataset	LP_1	LP_3	LP_5	LP_7	LP^e_1	LP^e_3	LP^e_5	LP^e_7
Adult	85.38 (3.22)	84.91 (3.24)	**85.46 (3.15)**	84.65 (3.90)	85.43 (2.32)	86.65 (2.96)	87.63 (2.81)	**87.75 (2.18)**
Blood	72.07 (2.55)	74.26 (1.30)	74.84 (1.12)	**76.89 (2.34)**	74.95 (2.65)	76.14 (1.11)	76.44 (1.65)	**76.49 (1.98)**
CTG	91.90 (0.62)	92.17 (0.74)	**92.19 (1.04)**	92.10 (1.20)	**91.45 (0.56)**	90.92 (0.58)	90.63 (0.52)	90.23 (0.36)
Faults	72.43 (1.09)	72.73 (1.42)	72.77 (1.47)	**72.80 (1.29)**	**68.70 (1.16)**	67.54 (1.26)	66.31 (0.70)	66.24 (0.86)
German	71.44 (2.72)	72.24 (2.63)	**73.36 (2.78)**	72.82 (2.36)	72.90 (1.24)	74.32 (1.91)	**74.86 (2.04)**	74.12 (1.75)
Glass	69.15 (3.48)	**69.81 (3.06)**	68.49 (4.16)	66.51 (4.45)	**65.57 (3.05)**	64.34 (5.93)	64.62 (4.78)	62.08 (2.36)
Haberman	68.49 (2.91)	70.20 (3.34)	**70.33 (4.49)**	69.21 (3.23)	68.29 (2.09)	69.80 (3.29)	71.12 (3.21)	**72.50 (3.08)**
Heart	81.69 (4.01)	80.29 (4.54)	81.32 (4.91)	**82.13 (4.73)**	80.07 (3.83)	**83.38 (3.57)**	83.09 (3.95)	83.09 (3.32)
Ionosphere	88.98 (2.25)	89.77 (1.77)	91.02 (1.60)	**91.53 (1.40)**	90.23 (1.40)	91.25 (1.73)	91.25 (1.48)	**91.31 (1.93)**
Laryngeal1	78.68 (6.19)	**79.72 (4.97)**	79.06 (5.71)	78.30 (5.18)	80.00 (3.74)	79.43 (5.44)	**80.66 (5.26)**	80.38 (5.56)
Laryngeal3	69.61 (3.27)	70.00 (2.61)	71.91 (1.71)	**72.02 (2.44)**	66.57 (1.58)	67.08 (1.75)	**67.47 (1.12)**	66.35 (1.12)
Liver	59.30 (3.72)	60.58 (4.82)	**61.45 (5.67)**	61.22 (5.30)	61.98 (3.56)	63.14 (3.27)	64.30 (3.23)	**67.33 (1.25)**
Mammographic	76.71 (3.16)	78.10 (2.23)	78.44 (1.99)	**78.89 (2.44)**	81.71 (2.92)	**82.45 (1.52)**	82.26 (2.40)	82.28 (1.73)
Monk2	94.07 (1.60)	**95.74 (1.22)**	95.00 (0.87)	94.91 (0.97)	**95.14 (1.47)**	94.40 (1.14)	94.07 (0.76)	94.07 (0.76)
Phoneme	**89.52 (0.50)**	89.14 (0.40)	89.01 (0.60)	88.94 (0.45)	87.22 (0.33)	**87.32 (0.36)**	87.08 (0.42)	86.97 (0.58)
Pima	71.02 (1.76)	71.77 (1.71)	**73.02 (1.69)**	72.86 (1.61)	72.34 (1.66)	74.40 (1.15)	75.68 (1.03)	**76.80 (2.23)**
Sonar	83.85 (3.33)	83.56 (5.60)	**84.23 (4.35)**	83.27 (5.30)	**80.00 (3.08)**	77.98 (3.39)	76.92 (3.18)	76.35 (3.42)
Vehicle	72.85 (1.53)	74.34 (2.24)	74.62 (2.60)	**74.72 (1.82)**	73.00 (2.21)	**73.25 (1.79)**	72.48 (1.75)	71.63 (1.55)
Vertebral	81.73 (3.35)	82.56 (2.77)	83.65 (2.46)	**85.45 (2.40)**	85.58 (3.32)	86.28 (3.28)	**87.31 (3.79)**	86.41 (2.74)
Weaning	84.08 (2.37)	85.92 (2.49)	86.32 (1.73)	**86.38 (1.72)**	86.25 (2.15)	85.79 (1.98)	85.39 (1.75)	**86.32 (2.43)**
Average	78.15	78.89	**79.32**	79.28	78.37	78.79	**78.98**	78.93
Avg rank	3.4	2.6	**1.8**	2.2	2.85	**2.325**	**2.325**	2.5
p-value	7.77×10^{-4}				0.52			

The Friedman tests indicate there is no significant difference between the configurations that use K-NNE, and though in the K-NN cases there is a significant difference, as their *p-value* suggests, the difference lies in the LP_1 configuration, which obtained a much worse average rank in the tests. Since for most DCS techniques and configurations the pool size $M = 5$ yielded the greatest mean accuracy rate and the best average rank, the parameter is thus set in the comparative study.

Discussion

From Table 4.2, it can be observed that the two neighborhood acquisition methods used in the proposed technique yielded quite distinct results: the LP_m configurations always surpassed, by far most of the times, the LP^e_m configurations for the multi-class problems for all three DCS techniques. The reason for this difference in performance lies in the neighborhood selection schemes used in the proposed method, as it can be observed in Figure 4.2, in which two multi-class toy problems are depicted.

In Figure 4.2a, the neighborhood θ_1 of the query instance $\mathbf{x_q}$ was obtained selecting que sample's K nearest neighbors. It can be observed that, since the border contains only two classes (Class 1 and Class 2), this is also the case for all two-class problems. Therefore, the SGH method, which generates only two-class classifiers, returns a pool with only one classifier ($c_{1,1}$) that cover the entire neighborhood θ_1. Figure 4.2b shows the same scenario, but with θ_1 being obtained using the version of K-NNE used in this work, which returns the same amount of neighboring instances for all classes in the original K-NN neighborhood. That is, the instances from classes too far from the query sample are not included in this method, as Figure 4.2b shows. The generated pool also contains only one classifier ($c_{1,1}$) that cover the instances in θ_1. In both presented cases, the DCS technique would select the correct classifier for this query sample, which belongs to Class 1, though the classifier from Figure 4.2b seems better adjusted than the one from Figure 4.2a.

On the other hand, Figure 4.2c shows a similar situation, but with Class 3 much closer to the other two classes. In this case, the neighborhood θ_1 returned by K-NN contains instances from the three classes in the problem. Since the SGH method only generates two-class classifiers, the coverage of θ_1 is incomplete. This is due to the fact that the most distant class in the input set is selected more frequently to draw the hyperplanes. It can be observed in Figure 4.2c that Class 3, which is the farthest class and thus the least relevant one, is much better covered, with all classifiers recognizing it, than the other two classes. In fact, there is not one classifier that separates Class 1 from Class 2 in the generated pool. However, since the DCS technique evaluates the classifiers competence over θ_1 in the proposed technique, Class 3 only possesses one instance, therefore its weight is much smaller than the remaining two classes in the classifiers' score. That way, the classifier $c_{1,3}$ would be selected by OLA, for instance, which would yield the correct label of $\mathbf{x_q}$.

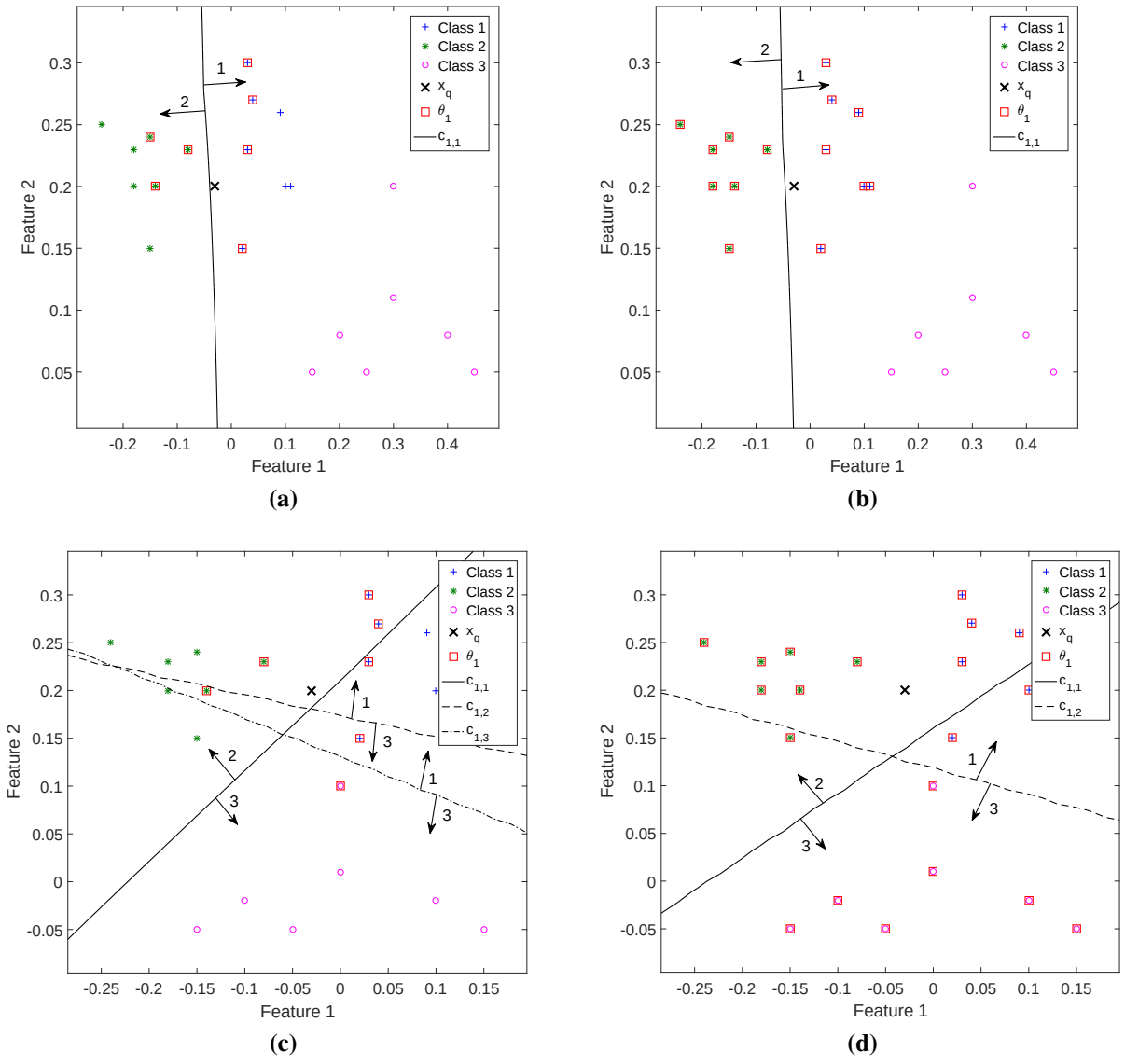


Figure 4.2: Example of pool generation for multi-class problems. In all scenarios, $\mathbf{x_q}$ belongs to Class 1. In (a) and (c), the query instance's ($\mathbf{x_q}$) neighborhood θ_1 was obtained using K-NN with $K_1 = 7$. In (b) and (d), θ_1 was obtained using a version of K-NNE with $K_1 = 7$ as well. These neighborhoods were used as input to the SGH method, which yielded the corresponding subpool of classifiers depicted in the images.

Figure 4.2d depicts the same scenario from Figure 4.2c, but with θ_1 obtained using K-NNE. Since the original K-NN neighborhood already contained an instance from Class 3, this class is also included in θ_1. This leads to the neighborhood containing $K_1 = 7$ instances of each of the three classes of the problem. The SGH method generates then two classifiers ($c_{1,1}$ and $c_{1,2}$), and, as in the previous case, the most distant and least relevant class (Class 3) is favoured by the method, since all classifiers recognize it. The other two classes, which are closer to $\mathbf{x_q}$, do not have a classifier in this subpool to distinguish among themselves. However, as opposed to the previous case, the amount of instances of the farthest class is the same as the other two classes, which makes its as relevant as the closer classes for the DCS techniques, since the classifiers are evaluated over the entire θ_1. In this example, as both classifiers correctly label two out of three classes in the neighborhood, the DCS technique would choose one of them randomly, which would in turn fairly degrade the performance of the system.

Therefore, a better approach for multi-class problems is to use the LP^e_m, which evaluates over the original neighborhood and is likely to give less weight to less relevant classes in the border region. Therefore, the combined configurations of the proposed method, LP_m for binary problems and LP^e_m for multi-class problems, is referenced as LP^{mc}_m and is used from this point forward in this work.

4.4 Comparative Study

In this section, a comparative study on the performances of the proposed method and related approaches are presented. For simplicity, the proposed method's configuration used in this study is the LP^{mc}_5, which generates $M = 5$ classifiers and uses K-NN for multi-class problems and K-NNE for two-class ones, since it performed the best in average rank and mean accuracy rate (Section 4.3.2. The baseline method used in the comparison is Bagging with a pool size set to 100 classifiers. Moreover, the SGH method over the entire training set is also tested and compared, since it provides another global approach for generating classifiers. The pool generated by this technique is referenced as the global pool (GP).

Another related method, though it is not a generation one, is the Frienemy Indecision Region Dynamic Ensemble Selection (FIRE-DES) framework (OLIVEIRA; CAVALCANTI; SABOURIN, 2017). In the FIRE-DES framework, when a query sample is in an *indecision region*, that is, a neighborhood that contains more than one class, the classifiers that correctly label instances from different classes in the query sample's region of competence are pre-selected to form the pool used in the DS technique. That is, if a border is detected in the query sample's RoC, the selection scheme searches only among the classifiers that cross this border. This is performed using the Dynamic Frienemy Pruning (DFP), an online pruning method for DS techniques. The FIRE-DES framework is designed for two-class problems, and it obtained a significant increase in accuracy for most DS techniques, specially for highly imbalanced datasets, in which cases the DFP method provided a considerable improvement in performance for those

techniques.

As shown in (OLIVEIRA; CAVALCANTI; SABOURIN, 2017) for two-class problems, it is advantageous to use locally accurate classifiers for query samples close to a class border. Due to the proposed local subpool generation in this work (Figure 3.2), it is guaranteed that all classifiers in the final local pool (*LP*) cross the query sample's RoC. Thus, the same idea of using only locally accurate classifiers for difficult regions indirectly applies to the proposed method. Therefore, the FIRE-DES framework is also included in the comparative study that follows. The pool used in this framework is the same as the one from the Bagging configuration, which contains 100 classifiers.

The performance of these configurations is evaluated in memorization, using the hit rate measure, and in generalization, using the accuracy rate over the datasets from Table 4.1. The analysis on the hit rate is performed in Section 4.4.1, whilst the accuracy rates are compared in Section 4.4.2.

4.4.1 Hit Rate

The hit rate (SOUZA et al., 2017) is a metric derived from the SGH method (Section 2.4.1) which indicates how well the generated pool integrates with the DCS techniques. In the SGH method, since the Oracle accuracy rate over the training set is 100%, each training instance is assigned to a classifier in the pool that correctly labels it. The hit rate is then obtained using the training set as test set, and comparing the chosen classifier to the correct classifier indicated by the SGH method for each training instance. Thus, the hit rate is the rate at which the DCS technique selects the correct classifier for a given known instance.

Since the hit rate is defined specifically for pools generated using the SGH method, the hit rate of the proposed method is only compared with the *GP* configuration, which uses a pool generated by the SGH method with the entire training set as input. The hit rate of the proposed configuration is calculated the same way as the *GP* configuration, with the only difference being for instances not in class overlap regions. In this case, the accuracy rate is used to compute the measure. The comparison between the *GP* and the *LP* configurations is relevant because it provides the answer to whether or not the generation over a local region instead of over the entire problem is useful in the selection process of a DCS technique.

Table 4.1 shows the mean hit rates of the *GP* and the LP_5^{mc} configurations. It can be observed that, for the majority of problems, the rate at which the DCS selects the correct classifier is greater when using a locally generated pool than using a globally generated one. A Wilcoxon signed rank test was performed on the difference between the mean hit rate of both configurations with a significance level of $\alpha = 0.5$. The results (row Wilcoxon) show that there is a significant difference on the frequency at which the classifiers are correctly selected in the local pool in comparison with the global pool for all three DCS techniques. This suggests that using the same perspective in generation and selection may indeed help the DCS techniques in the selection

Table 4.1: Mean and standard deviation of the hit rate, i.e., the rate at which the right Perceptron is chosen by (a) OLA,(b) LCA and (c) MCB using the *GP* and the LP_5^{mc} configurations. The row *Wilcoxon* shows the result of a Wilcoxon signed rank test for the null hypothesis that the difference between the hit rates of the proposed configuration and the GP configuration comes from a distribution with zero median. The significance level was $\alpha = 0.05$, and the symbols $+$, $-$ and $\sim$ indicate whether the if the compared method is significantly superior, inferior or not significantly different from the proposed method, respectively. Best results are in bold.

(a)

Dataset	GP	LP_5^{mc}
Adult	86.91 (0.87)	**89.03 (0.88)**
Blood	**79.59 (0.51)**	78.95 (1.10)
CTG	92.50 (0.59)	**94.70 (0.23)**
Faults	76.88 (1.26)	**80.81 (0.56)**
German	71.05 (1.44)	**85.66 (1.03)**
Glass	**76.21 (1.98)**	70.90 (0.71)
Haberman	**76.26 (1.10)**	74.24 (0.55)
Heart	84.06 (1.92)	**88.32 (0.86)**
Ionosphere	86.46 (1.48)	**92.62 (0.91)**
Laryngeal1	84.75 (2.07)	**87.16 (1.29)**
Laryngeal3	74.81 (2.95)	**86.09 (0.81)**
Liver	67.22 (1.40)	**77.43 (1.25)**
Mammographic	**82.72 (0.64)**	82.16 (0.87)
Monk2	85.77 (3.60)	**95.83 (0.34)**
Phoneme	87.40 (0.46)	**89.89 (0.16)**
Pima	75.64 (1.55)	**83.22 (0.51)**
Sonar	80.00 (3.62)	**90.94 (0.99)**
Vehicle	76.14 (1.49)	**83.49 (0.86)**
Vertebral	82.39 (2.14)	**87.98 (1.00)**
Weaning	83.45 (1.33)	**93.80 (0.70)**
Average	80.51	**85.66**
Wilcoxon	-	n/a

(b)

Dataset	GP	LP_5^{mc}
Adult	86.77 (0.92)	**89.42 (0.98)**
Blood	**80.20 (0.35)**	78.70 (1.24)
CTG	92.63 (0.44)	**95.23 (0.19)**
Faults	76.84 (1.01)	**81.47 (0.57)**
German	75.75 (1.35)	**86.16 (0.92)**
Glass	**77.95 (1.92)**	76.11 (1.48)
Haberman	**76.61 (1.46)**	73.55 (0.79)
Heart	83.86 (2.40)	**88.95 (0.96)**
Ionosphere	87.34 (1.53)	**93.90 (1.19)**
Laryngeal1	84.81 (2.38)	**87.46 (1.28)**
Laryngeal3	73.98 (1.99)	**87.16 (0.90)**
Liver	70.62 (2.91)	**77.72 (1.36)**
Mammographic	**82.83 (1.54)**	80.75 (0.67)
Monk2	91.82 (3.61)	**95.69 (0.34)**
Phoneme	89.48 (0.44)	**90.00 (0.16)**
Pima	76.02 (1.67)	**83.52 (0.50)**
Sonar	83.46 (3.45)	**91.85 (0.95)**
Vehicle	77.98 (1.57)	**82.84 (0.74)**
Vertebral	84.33 (2.32)	**88.28 (1.00)**
Weaning	84.38 (1.72)	**94.19 (0.92)**
Average	81.88	**86.15**
Wilcoxon	-	n/a

(c)

Dataset	GP	LP_5^{mc}
Adult	87.14 (0.73)	**89.32 (0.90)**
Blood	**79.61 (0.51)**	78.76 (1.21)
CTG	92.49 (0.63)	**95.16 (0.20)**
Faults	76.87 (1.26)	**81.50 (0.49)**
German	71.23 (1.47)	**86.24 (0.99)**
Glass	**76.27 (1.99)**	73.36 (1.07)
Haberman	**76.35 (1.10)**	74.02 (0.83)
Heart	83.96 (1.72)	**89.18 (0.76)**
Ionosphere	86.43 (1.43)	**94.79 (1.00)**
Laryngeal1	84.75 (1.93)	**87.41 (1.39)**
Laryngeal3	74.85 (2.90)	**87.11 (1.10)**
Liver	67.34 (1.28)	**77.53 (1.33)**
Mammographic	**82.68 (0.73)**	82.14 (0.74)
Monk2	86.67 (4.48)	**95.83 (0.35)**
Phoneme	87.40 (0.47)	**90.08 (0.18)**
Pima	75.82 (1.83)	**83.52 (0.49)**
Sonar	80.19 (3.63)	**92.17 (0.98)**
Vehicle	76.20 (1.51)	**83.64 (0.75)**
Vertebral	82.39 (2.19)	**88.30 (1.02)**
Weaning	83.36 (1.20)	**94.29 (0.83)**
Average	80.60	**86.22**
Wilcoxon	-	n/a

process.

4.4.2 Accuracy Rate

The methods described in Section 4.4 were evaluated over the test set and Table 4.2 shows the mean accuracy rates for OLA, LCA and MCB. It can be observed that the proposed method obtained a greater overall performance in comparison with all other methods for the three DCS techniques. Generally, problems that contain a higher percentage of instances near the class borders, such as "Monk2", "German", "Pima" and "Weaning" (Figure 4.1) considerably benefited from the use of the local pool. However, this does not determine when it is best to use the local pool. For instance, for the "Ionosphere" dataset, which possesses only about 30% of instances in overlap areas, the use of locally generated classifiers largely increased the recognition rates, whist for a dataset such as "Sonar", with more than 70% of borderline samples, the accuracy rates were quite impaired. Overall, the proposed configuration yielded greater accuracy rates for at least half of the datasets used in the experiments, for all three DCS techniques.

A Wilcoxon signed rank test with significance level of $\alpha = 0.05$ was also performed on the mean accuracy rates of the proposed method and each one of the three compared methods. The results are depicted in the *Wilcoxon* row. It can be observed that the proposed configuration is significantly superior to the other three methods for all DCS techniques. This suggests that integrating locally generated pools to DCS techniques may be advantageous performance-wise compared to globally generated ones, in most cases.

The Friedman test was also used to compare the performance of all configurations from Table 4.2 for each of the three DCS techniques evaluated. The level of significance of the test was set to $\alpha = 0.05$, and the average rank of each configuration and the p-value of the test are shown in the *Avg rank* row and the *p-value* row, respectively. It can be observed that the proposed method obtained the highest average rank for all DCS techniques.

Since the resulting p-values indicate that there is a significant difference between the performances of the evaluated configurations for all three DCS techniques, a post-hoc Bonferroni-Dunn test was performed afterwards to obtain a pairwise comparison between the configurations. Two configurations are significantly different if the difference between their average rank is greater than the critical difference *CD*. The critical difference diagrams (DEMŠAR, 2006) depicted in Figure 4.3, show the results of the post-hoc tests for each DCS technique. The configurations with no significant difference are connected by a bar, whilst significantly different ones are not intersected in the diagram.

The critical difference value obtained by the Bonferroni-Dunn post-hoc test was $CD = 1.0488$. It can be observed from Figure 4.3 that the proposed configuration yielded a significantly superior performance in comparison with the FIRE-DES framework for all DCS techniques, which suggests that generating locally accurate classifiers is a better strategy than pruning a

Table 4.2: Mean and standard deviation of the accuracy rate of using (a) OLA, (b) LCA and (c) MCB for a pool with 100 Perceptrons generated using Bagging (column Bagging), a pool of 100 Perceptrons generated using Bagging and pruned with the DFP method (column FIRE-DES), the *GP* configuration and the LP_5^{mc} configuration. The row *Wilcoxon* shows the result of a Wilcoxon signed rank test for the null hypothesis that the difference between the mean accuracy rates of the proposed configuration and each of the remaining methods comes from a distribution with zero median. The significance level was $\alpha = 0.05$, and the symbols $+$, $-$ and $\sim$ indicate if the compared method is significantly superior, inferior or not significantly different from the proposed method, respectively. The row *Avg rank* shows the resulting mean ranks of a Friedman test with a significance level of $\alpha = 0.05$, and the p-value of the test is shown in row *p-value*. Best results are in bold.

(a)

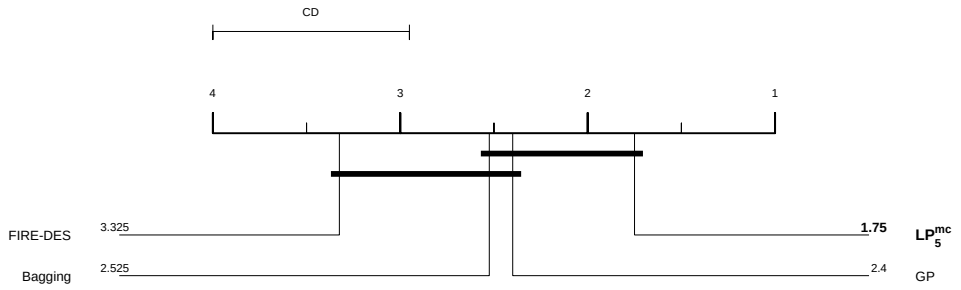
Dataset	**Bagging**	**FIRE-DES**	**GP**	**LP_5^{mc}**
Adult	85.58 (3.43)	84.34 (3.41)	**88.15 (2.93)**	87.69 (2.87)
Blood	75.05 (2.24)	68.94 (2.91)	75.53 (1.14)	**76.68 (1.50)**
CTG	88.53 (1.62)	88.36 (1.72)	90.24 (0.77)	**91.89 (1.01)**
Faults	66.52 (1.65)	65.33 (1.95)	71.91 (1.60)	**72.65 (1.69)**
German	70.96 (2.50)	68.68 (2.48)	70.04 (2.35)	**74.76 (1.87)**
Glass	60.00 (6.97)	59.81 (7.04)	66.79 (4.17)	**69.62 (4.93)**
Haberman	**72.17 (5.22)**	66.91 (4.64)	71.58 (5.24)	70.99 (2.32)
Heart	80.74 (4.45)	80.59 (4.59)	**86.62 (2.18)**	83.68 (3.66)
Ionosphere	86.70 (2.95)	86.53 (3.00)	87.16 (2.76)	**91.02 (1.42)**
Laryngeal1	**82.17 (4.04)**	81.70 (5.16)	80.38 (4.26)	80.57 (5.13)
Laryngeal3	71.52 (5.97)	68.54 (5.32)	**72.25 (1.71)**	71.74 (1.90)
Liver	**64.71 (4.64)**	64.48 (5.19)	58.37 (3.53)	64.01 (3.03)
Mammographic	82.07 (1.77)	78.75 (3.56)	**82.60 (2.47)**	82.38 (2.32)
Monk2	87.82 (3.60)	87.45 (3.59)	86.20 (3.74)	**94.17 (0.74)**
Phoneme	80.25 (0.69)	75.89 (0.95)	86.74 (0.73)	**87.06 (0.41)**
Pima	72.27 (2.61)	69.04 (2.84)	72.29 (2.39)	**76.15 (1.38)**
Sonar	**81.44 (2.36)**	**81.44 (2.59)**	80.00 (3.33)	76.73 (3.47)
Vehicle	74.74 (2.11)	**75.14 (2.25)**	70.09 (2.57)	73.61 (2.27)
Vertebral	84.68 (3.52)	84.87 (3.69)	81.41 (2.06)	**87.44 (3.57)**
Weaning	76.05 (4.00)	76.18 (3.91)	78.68 (3.71)	**85.66 (1.86)**
Average	77.19	75.65	77.85	**79.92**
Wilcoxon	-	-	-	n/a
Avg rank	2.525	3.325	2.4	**1.75**
p-value	0.0017			

(b)

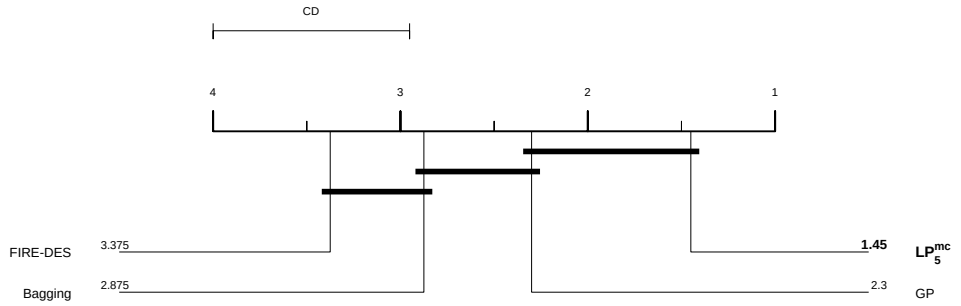
Dataset	Bagging	FIRE-DES	GP	LP_5^{mc}
Adult	86.76 (3.55)	86.01 (3.34)	**87.40 (2.82)**	87.37 (3.39)
Blood	75.69 (2.13)	70.64 (2.41)	75.74 (1.04)	**76.49 (1.49)**
CTG	88.36 (1.24)	88.34 (1.44)	90.30 (0.84)	**92.23 (0.98)**
Faults	66.00 (1.69)	65.67 (2.23)	71.99 (1.53)	**73.64 (1.67)**
German	71.62 (1.61)	70.60 (1.70)	70.84 (1.87)	**74.88 (2.04)**
Glass	57.64 (4.56)	57.74 (4.76)	69.43 (3.33)	**70.28 (3.62)**
Haberman	**71.97 (4.21)**	70.13 (4.56)	71.05 (1.91)	71.12 (2.20)
Heart	81.25 (4.72)	81.25 (4.72)	**86.47 (2.85)**	83.68 (3.66)
Ionosphere	86.14 (4.28)	85.97 (4.21)	87.27 (3.21)	**91.59 (1.89)**
Laryngeal1	80.38 (3.26)	79.91 (3.31)	**80.94 (4.70)**	80.57 (5.13)
Laryngeal3	70.62 (5.43)	67.64 (6.66)	72.58 (2.14)	**72.92 (2.39)**
Liver	65.41 (4.84)	**66.28 (4.48)**	58.37 (2.81)	63.90 (2.93)
Mammographic	81.59 (3.05)	78.97 (4.24)	81.63 (3.06)	**82.38 (2.58)**
Monk2	85.60 (4.30)	85.42 (4.42)	90.28 (2.18)	**94.12 (0.75)**
Phoneme	80.84 (0.57)	77.09 (0.91)	87.01 (0.77)	**87.06 (0.46)**
Pima	74.92 (2.81)	73.67 (2.88)	73.23 (3.39)	**76.04 (1.26)**
Sonar	77.50 (4.42)	77.79 (4.43)	**78.08 (5.01)**	76.92 (3.18)
Vehicle	72.52 (1.38)	72.88 (1.38)	70.75 (2.22)	**73.92 (2.39)**
Vertebral	84.74 (2.97)	84.87 (2.75)	82.31 (1.93)	**87.44 (3.57)**
Weaning	73.16 (3.61)	73.36 (3.49)	78.82 (3.05)	**85.66 (1.86)**
Average	76.63	75.71	78.22	**80.11**
Wilcoxon	-	-	-	n/a
Avg rank	2.875	3.375	2.3	**1.45**
p-value	1.78×10^5			

(c)

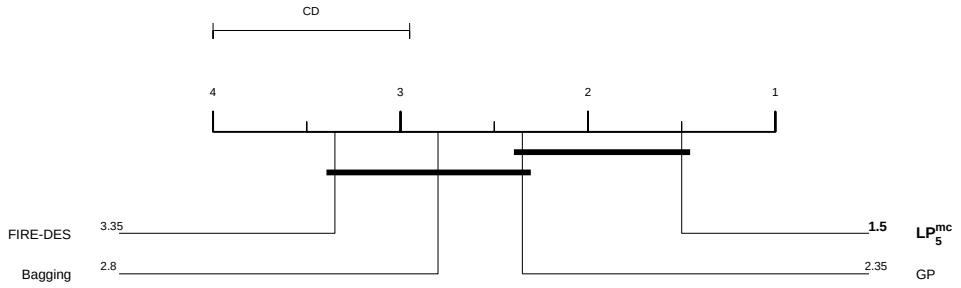
Dataset	Bagging	FIRE-DES	GP	LP_5^{mc}
Adult	85.28 (2.92)	83.41 (2.61)	**88.15 (2.93)**	87.63 (2.81)
Blood	75.34 (1.89)	68.80 (3.73)	75.53 (1.14)	**76.44 (1.65)**
CTG	88.42 (1.53)	88.36 (1.64)	90.24 (0.77)	**92.19 (1.04)**
Faults	66.58 (1.37)	65.77 (2.32)	71.91 (1.60)	**72.77 (1.47)**
German	70.54 (2.02)	68.62 (2.20)	70.52 (2.08)	**74.86 (2.04)**
Glass	60.00 (7.04)	59.71 (6.87)	66.79 (4.17)	**68.49 (4.16)**
Haberman	70.32 (4.78)	66.97 (3.33)	**71.71 (4.91)**	71.12 (3.21)
Heart	81.91 (5.44)	83.38 (3.50)	**86.18 (2.36)**	83.09 (3.95)
Ionosphere	87.61 (2.32)	86.70 (2.63)	87.16 (2.71)	**91.25 (1.48)**
Laryngeal1	81.88 (4.25)	**82.26 (5.09)**	80.57 (4.59)	80.66 (5.26)
Laryngeal3	70.22 (6.48)	68.20 (4.72)	71.80 (1.58)	**71.91 (1.71)**
Liver	64.01 (5.11)	64.18 (4.88)	58.37 (3.49)	**64.30 (3.23)**
Mammographic	82.16 (1.86)	79.18 (3.52)	**82.60 (2.47)**	82.26 (2.40)
Monk2	87.51 (3.92)	87.36 (3.75)	87.96 (3.80)	**94.07 (0.76)**
Phoneme	80.53 (0.61)	76.58 (0.94)	86.73 (0.73)	**87.08 (0.42)**
Pima	72.08 (2.99)	68.54 (2.79)	72.71 (2.67)	**75.68 (1.03)**
Sonar	80.19 (3.64)	**82.30 (3.27)**	79.81 (3.09)	76.92 (3.18)
Vehicle	73.89 (1.97)	73.82 (2.76)	70.14 (2.52)	**74.62 (2.60)**
Vertebral	84.61 (4.15)	84.80 (3.60)	82.69 (2.22)	**87.31 (3.79)**
Weaning	77.10 (3.94)	77.03 (3.65)	79.21 (3.30)	**85.39 (1.75)**
Average	77.01	75.80	78.03	**79.90**
Wilcoxon	-	-	-	n/a
Avg rank	2.8	3.35	2.35	**1.5**
p-value	6.46×10^5			



(a)



(b)



(c)

Figure 4.3: Critical difference diagram representing the results of a post-hoc Bonferroni-Dunn test on the accuracy rates of the methods from Table 4.2 for (a) OLA, (b) LCA and (c) MCB. The calculated critical difference value was $CD = 1.0488$. The values near the methods' labels indicate their average rank. Statistically similar methods are connected by an horizontal line, while statistically different ones are disconnected.

large pool in search of such classifiers for instances in overlap regions, at least for balanced and moderately imbalanced problems, as used in the experiments. The proposed configuration also performed significantly better than Bagging when using LCA and MCB. Thus, using locally generated pools may indeed be an advantageous alternative to using globally generated ones for DCS techniques.

4.5 Conclusion

In this chapter, a performance analysis of the proposed method was presented. Experiments were conducted over 20 datasets according to the experimental protocol presented in Section 4.2.

An analysis on the proposed method alone was performed in Section 4.3. It was shown that the proposed method generates local pools for instances truly in class overlap regions of the feature space, in most cases. A parameter sensitivity analysis was also performed, with variations on the neighborhood acquisition method and the pool size. It was observed that each problem benefits more from a specific number of classifiers and a neighborhood acquisition method, so in order to achieve the best possible results, a fine tuning of the method's parameters would be necessary. Moreover, it was observed that, for multi-class problems, the use of K-NNE to obtain the neighborhoods in the generation process of the method degrades the performance of the latter. This was due to the nature of the SGH method, which generates only two-class classifiers. Thus, a combined method that uses K-NNE for two-class problems and K-NN for multi-class ones was proposed at the end of the proposed method's analysis.

The performance in memorization and in generalization of the proposed method and other related techniques was compared in Section 4.4. It was shown that the rate at which the DCS techniques select the correct classifier in memorization was significantly increased when using locally generated pools, in comparison with globally generated ones. Therefore, the hit rate of the proposed method suggests that the DCS techniques seem to benefit from using locally generated classifiers in the selection process.

Furthermore, the performances of the proposed method and three related methods were evaluated in generalization. It was observed that the proposed method yields an overall accuracy rate greater than all other compared methods. The use of local pools was considerably advantageous for problems with high percentages of borderline samples. Moreover, a Wilcoxon signed rank test yielded that the proposed method was significantly superior than the other three methods for all DCS techniques.

5

Conclusion

In this work, an overview of the MCS field was presented. The stages of an MCS were introduced, as well as the Oracle model and its importance. The process of dynamically selecting classifiers was also further explained, and the most relevant DCS techniques were introduced. It was also shown on a short analysis that the DCS techniques had difficulty in selecting a competent classifier even though the presence of such a classifier in the pool was assured by the Oracle model. It was reasoned that the Oracle, being performed globally, did not help in the search for a good pool of classifiers for DCS techniques, because the latter use only local data to select a competent classifier for any given instance.

Based on that observation, an online local pool generation method was proposed in this work. The proposed technique involved generating subpools for each unknown instance in class overlap regions of the feature space, so that a more locally accurate pool could be used, in hopes that, by fully covering these regions with a locally specialist pool, it would be easier for the DCS techniques to select the most competent classifiers for these instances. On the other hand, instances surrounded by only one class would be labelled using a nearest neighbors rule. The reasoning behind the proposed method is that, by generating the classifiers in the same perspective as they are selected by the DCS techniques, the latter could better integrate to the generated pool and thus select the most competent classifier more often, which in turn would result in higher recognition rates.

Experiments were conducted over 20 public datasets. A parameter sensitivity analysis on the proposed method was performed and it was concluded that the best parameter setting varied from problem to problem. Moreover, it was shown that, due to limitations in the SGH method, the K-NNE was not suited to be used in the proposed method for multi-class problems. Thus, it was suggested the use of the regular K-NN rule for these cases and K-NNE for two-class datasets. This combined approach was used in the comparative study, which evaluated the proposed method and three other related methods.

The proposed method and a globally generated pool were tested over the training set, and it was shown that the hit rate, that is, the rate at which the DCS techniques select the correct classifier, was significantly increased for the proposed method. Thus, it was concluded that the integrating locally generated pools to DCS techniques may actually help the latter in the

selection of the best classifier. Moreover, the proposed method and other three related methods were evaluated in generalization and it was shown that the former outperforms the latter, on average, specially for problems with a higher proportion of instances near the borders. It was also shown that the proposed method yielded a significantly superior accuracy rate in comparison with the other three methods. Thus, it was concluded that, not only do the DCS techniques select the best classifier more frequently, but also the recognition rates of the DCS techniques indeed increase when using the same local perspective in the pool generation stage.

Future works may include developing an automatic scheme for defining the parameters of the proposed method, for, as it was shown, the proposed method requires fine tuning in order to obtain the best performance for each specific problem. Furthermore, the generation process may also be adapted to better accommodate multi-class problems, since the proposed method, as the SGH method, generates only two-class classifiers, and this may hinder the performance of the proposed technique.

References

ALCALÁ, J. et al. KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. **Journal of Multiple-Valued Logic and Soft Computing**, [S.l.], v.17, n.2-3, p.255–287, 2011.

ALKOOT, F.; KITTLER, J. Experimental evaluation of expert fusion strategies. **Pattern Recognition Letters**, [S.l.], v.20, n.11, p.1361 – 1369, 1999.

BACHE, K.; LICHMAN, M. **UCI machine learning repository**. [Online], Available: `http://archive.ics.uci.edu/ml`.

BREIMAN, L. Bagging predictors. **Machine Learning**, [S.l.], v.24, n.2, p.123–140, 1996.

BRITTO, A.; SABOURIN, R.; OLIVEIRA, L. Dynamic selection of classifiers - A comprehensive review. **Pattern Recognition**, [S.l.], v.47, n.11, p.3665–3680, 2014.

BROWN, G. et al. Diversity creation methods: a survey and categorisation. **Information Fusion**, [S.l.], v.6, n.1, p.5–20, 2005.

BRUN, A. L. et al. Contribution of data complexity features on dynamic classifier selection. In: INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS (IJCNN), 2016. **Anais...** [S.l.: s.n.], 2016. p.4396–4403.

CRUZ, R. M.; CAVALCANTI, G. D.; REN, T. I. Handwritten digit recognition using multiple feature extraction techniques and classifier ensemble. In: INTERNATIONAL CONFERENCE ON SYSTEMS, SIGNALS AND IMAGE PROCESSING, 17. **Anais...** [S.l.: s.n.], 2010. p.215–218.

CRUZ, R. M. O. et al. META-DES: a dynamic ensemble selection framework using meta-learning. **Pattern Recognition**, [S.l.], v.48, n.5, p.1925–1935, 2015.

CRUZ, R. M.; SABOURIN, R.; CAVALCANTI, G. D. META-DES.H: a dynamic ensemble selection technique using meta-learning and a dynamic weighting approach. In: INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS, 2015. **Anais...** IEEE, 2015. p.1–8.

CRUZ, R. M.; SABOURIN, R.; CAVALCANTI, G. D. Prototype selection for dynamic classifier and ensemble selection. **Neural Computing and Applications**, [S.l.], p.1–11, 2016.

CRUZ, R. M.; SABOURIN, R.; CAVALCANTI, G. D. Dynamic classifier selection: recent advances and perspectives. **Information Fusion**, [S.l.], v.41, p.195–216, 2018.

DEMŠAR, J. Statistical comparisons of classifiers over multiple data sets. **Journal of Machine learning research**, [S.l.], v.7, n.Jan, p.1–30, 2006.

DIDACI, L. et al. A study on the performances of dynamic classifier selection based on local accuracy estimation. **Pattern Recognition**, [S.l.], v.38, n.11, p.2188–2191, 2005.

DIDACI, L.; GIACINTO, G. Dynamic classifier selection by adaptive k-nearest-neighbourhood rule. **Multiple Classifier Systems**, [S.l.], p.174–183, 2004.

DIETTERICH, T. G.; BAKIRI, G. Solving multiclass learning problems via error-correcting output codes. **Journal of Artificial Intelligence Research**, [S.l.], v.2, p.263–286, 1995.

DOS SANTOS, E. M.; SABOURIN, R.; MAUPIN, P. A dynamic overproduce-and-choose strategy for the selection of classifier ensembles. **Pattern Recognition**, [S.l.], v.41, n.10, p.2993–3009, 2008.

DOS SANTOS, E. M.; SABOURIN, R.; MAUPIN, P. Overfitting cautious selection of classifier ensembles with genetic algorithms. **Information Fusion**, [S.l.], v.10, n.2, p.150–162, 2009.

FERNáNDEZ-DELGADO, M. et al. Do We Need Hundreds of Classifiers to Solve Real World Classification Problems? **Journal of Machine Learning Research**, [S.l.], v.15, n.1, p.3133–3181, 2014.

GIACINTO, G.; ROLI, F. Methods for dynamic classifier selection. In: INTERNATIONAL CONFERENCE ON IMAGE ANALYSIS AND PROCESSING, 10. **Proceedings...** [S.l.: s.n.], 1999. p.659–664.

GIACINTO, G.; ROLI, F.; DIDACI, L. Fusion of multiple classifiers for intrusion detection in computer networks. **Pattern Recognition Letters**, [S.l.], v.24, n.12, p.1795–1803, 2003.

HO, T. K. The random subspace method for constructing decision forests. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, [S.l.], v.20, n.8, p.832–844, 1998.

HUANG, Y. S.; SUEN, C. Y. A method of combining multiple experts for the recognition of unconstrained handwritten numerals. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, [S.l.], v.17, n.1, p.90–94, 1995.

JAHRER, M.; TÖSCHER, A.; LEGENSTEIN, R. Combining predictions for accurate recommender systems. In: ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, 16. **Proceedings...** [S.l.: s.n.], 2010. p.693–702.

JUTTEN, C. **The Enhanced Learning for Evolutive Neural Architectures Project**. [Online], Available: `https://www.elen.ucl.ac.be/neural-nets/Research/Projects/ELENA/elena.htm`.

KING, R. D.; FENG, C.; SUTHERLAND, A. Statlog: comparison of classification algorithms on large real-world problems. **Applied Artificial Intelligence**, [S.l.], v.9, n.3, p.289–333, 1995.

KITTLER, J. et al. On combining classifiers. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, [S.l.], v.20, p.226–239, 1998.

KO, A. H. R.; SABOURIN, R.; JR., A. S. B. From dynamic classifier selection to dynamic ensemble selection. **Pattern Recognition**, [S.l.], v.41, n.5, p.1718–1731, 2008.

KO, A. H.-R.; SABOURIN, R.; SOUZA BRITTO JR, A. de. A new dynamic ensemble selection method for numeral recognition. In: INTERNATIONAL CONFERENCE ON MULTIPLE CLASSIFIER SYSTEMS, 7. **Anais...** Springer-Verlag, 2007. p.431–439.

KUNCHEVA, L. **Ludmila Kuncheva Collection**. [Online], Available: `http://pages.bangor.ac.uk/~mas00a/activities/real_data.htm`.

KUNCHEVA, L. **Combining pattern classifiers**: methods and algorithms. [S.l.]: J. Wiley, 2014.

KUNCHEVA, L. I. A theoretical study on six classifier fusion strategies. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, [S.l.], v.24, n.2, p.281–286, 2002.

KUNCHEVA, L. I.; RODRIGUEZ, J. J. Classifier ensembles with a random linear oracle. **IEEE Transactions on Knowledge and Data Engineering**, [S.l.], v.19, n.4, p.500–508, 2007.

KUNCHEVA, L. I.; WHITAKER, C. J. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. **Machine Learning**, [S.l.], v.51, n.2, p.181–207, 2003.

KURZYNSKI, M.; TRAJDOS, P. On a New Competence Measure Applied to the Dynamic Selection of Classifiers Ensemble. In: INTERNATIONAL CONFERENCE ON DISCOVERY SCIENCE. **Anais...** [S.l.: s.n.], 2017. p.93–107.

LIMA, T. P. F. de; SERGIO, A. T.; LUDERMIR, T. B. Improving Classifiers and Regions of Competence in Dynamic Ensemble Selection. In: BRAZILIAN CONFERENCE ON INTELLIGENT SYSTEMS (BRACIS), 2014. **Anais...** [S.l.: s.n.], 2014. p.13–18.

OLIVEIRA, D. V.; CAVALCANTI, G. D.; SABOURIN, R. Online pruning of base classifiers for Dynamic Ensemble Selection. **Pattern Recognition**, [S.l.], v.72, p.44 – 58, 2017.

OPITZ, D. W.; MACLIN, R. Popular ensemble methods: an empirical study. **Journal of Artificial Intelligence Research**, [S.l.], v.11, p.169–198, 1999.

PARTALAS, I.; TSOUMAKAS, G.; VLAHAVAS, I. P. Focused Ensemble Selection: a diversity-based method for greedy ensemble selection. In: ECAI. **Anais...** [S.l.: s.n.], 2008. p.117–121.

RAUDYS, Š. Trainable fusion rules. I. Large sample size case. **Neural Networks**, [S.l.], v.19, n.10, p.1506–1516, 2006.

SABOURIN, M. et al. Classifier combination for hand-printed digit recognition. In: SECOND INTERNATIONAL CONFERENCE ON DOCUMENT ANALYSIS AND RECOGNITION. **Proceedings...** [S.l.: s.n.], 1993. p.163–166.

SCHAPIRE, R. E. et al. Boosting the margin: a new explanation for the effectiveness of voting methods. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 14. **Anais...** Morgan Kaufmann Publishers Inc., 1997. p.322–330.

SIERRA, B. et al. K Nearest Neighbor Equality: giving equal chance to all existing classes. **Information Sciences**, [S.l.], v.181, n.23, p.5158–5168, 2011.

SMITS, P. C. Multiple classifier systems for supervised remote sensing image classification based on dynamic classifier selection. **IEEE Transactions on Geoscience and Remote Sensing**, [S.l.], v.40, n.4, p.801–813, 2002.

SOUZA, M. A. et al. On the characterization of the Oracle for dynamic classifier selection. In: INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS. **Anais...** [S.l.: s.n.], 2017. p.332–339.

TORRE, M. De-la et al. An adaptive ensemble-based system for face recognition in person re-identification. **Machine Vision and Applications**, [S.l.], v.26, n.6, p.741–773, 2015.

VALENTINI, G. An experimental bias-variance analysis of svm ensembles based on resampling techniques. **IEEE Transactions on Systems, Man, and Cybernetics**, [S.l.], v.Part B 35, p.1252–1271, 2005.

WANG, J.; NESKOVIC, P.; COOPER, L. N. Improving nearest neighbor rule with a simple adaptive distance measure. **Pattern Recognition Letters**, [S.l.], v.28, n.2, p.207–213, 2007.

WOLPERT, D.; MACREADY, W. No free lunch theorems for optimization. **IEEE Transactions on Evolutionary Computation**, [S.l.], v.1, p.67–82, 1997.

WOODS, K.; KEGELMEYER JR, W. P.; BOWYER, K. Combination of multiple classifiers using local accuracy estimates. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, [S.l.], v.19, n.4, p.405–410, 1997.

WOŹNIAK, M.; GRAÑA, M.; CORCHADO, E. A survey of multiple classifier systems as hybrid systems. **Information Fusion**, [S.l.], v.16, p.3–17, 2014.

XIAO, H.; XIAO, Z.; WANG, Y. Ensemble classification based on supervised clustering for credit scoring. **Applied Soft Computing**, [S.l.], v.43, p.73–86, 2016.

XIAO, J. et al. Dynamic classifier ensemble model for customer classification with imbalanced class distribution. **Expert Systems with Applications**, [S.l.], v.39, n.3, p.3668–3675, 2012.

ZHOU, Z. **Ensemble methods**. [S.l.]: Taylor & Francis, 2012.

Appendix

A

The Self-Generating Hyperplanes Method

Since the SGH generation method possesses some interesting properties, as discussed in Section 2.4.1, and for that reason is widely used in this work, it is presented in more detail with a step-by-step analysis here. Algorithm 3 shows the procedure for generating the hyperplanes in the SGH method. The toy problem used in the step-by-step analysis is shown in Figure A.1. The training set of the toy problem contains $N = 350$ instances and $L = 5$ classes. The step-by-step execution of the algorithm for this example happens as follows.

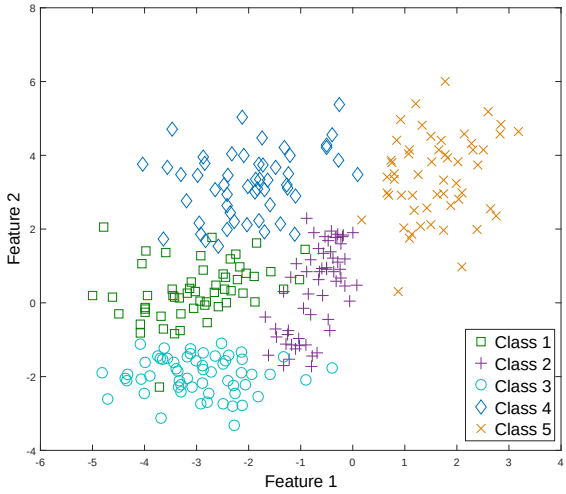


Figure A.1: Training set $\mathscr{T}$ of the toy problem, containing $N = 350$ instances and $L = 5$ classes.

Step 1 of Algorithm 3 consist of assigning to Ω the five possible labels of the problem ($\{1,2,3,4,5\}$). Step 2 assigns the Pool to an empty set, and Step 3 the classifier count is started.

In the first iteration of the algorithm's outer loop, the centroids of each of the five classes are calculated and stored in the set $\mathscr{R}$ (Step 5 to Step 7). All five centroids are represented by asterisks (*) in Figure A.2a. Then, in Step 8 and Step 9, the two most distant points in $\mathscr{R}$ are chosen, in this case $\mathbf{r_3}$ and $\mathbf{r_5}$, and their classes $i,j = 3,5$ identified. In Figure A.2a, Class 3 and Class 5 centroids are the large asterisks in red.

From Step 10 to step 13, the weights $\mathbf{w_1}$ and bias b_1 of classifier c_1 are calculated, using Equation A.1 and Equation A.2 with $i,j = 3,5$. The weights $\mathbf{w_1}$ of classifier c_1 are the coordinates of the normalized distance vector between the centroids ($\hat{n_{3,5}}$), while the bias b_1 is obtained by

Algorithm 3 Self-generating Hyperplanes (SGH) method.

Input: $\mathscr{T} = \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_N}\}$ ▷ Training dataset
Output: C ▷ Final pool
1: $\Omega \leftarrow \{\omega_1, \omega_2, ..., \omega_L\}$ ▷ Set of problem classes
2: $C \leftarrow \{\}$ ▷ Pool initially empty
3: $m = 1$ ▷ Classifier count
4: **while** $\mathscr{T} \neq \{\}$ **do**
5: **for** every ω_l in Ω **do**
6: $\mathbf{r_l} \leftarrow mean(\{\forall \mathbf{x_n} \in \mathscr{T} | \mathbf{x_n} \in \omega_l\})$ ▷ Calculate centroid of class ω_l
7: **end for**
8: $d \leftarrow max(pairwiseDistance(\mathscr{R}))$ ▷ Maximum distance between the classes' centroids in $\mathscr{R}$
9: $i, j \leftarrow findIndex(d)$
10: $p_{i,j} \leftarrow (\mathbf{r_i} + \mathbf{r_j})/2$ ▷ Calculate midpoint $p_{i,j}$ between the centroids
11: $\hat{\mathbf{n}_{\mathbf{i,j}}} \leftarrow (\mathbf{r_i} - \mathbf{r_j})/d$ ▷ Calculate normal vector
12: $\mathbf{w_m} \leftarrow \{\hat{\mathbf{n}_{\mathbf{i,j}}}\}$ ▷ Calculate weights of classifier c_m
13: $b_m \leftarrow -p_{i,j} \cdot \hat{\mathbf{n}_{\mathbf{i,j}}}$ ▷ Calculate bias of classifier c_m
14: $c_m \leftarrow constructPerceptron(\mathbf{w_m}, b_m)$
15: **for** every $\mathbf{x_n}$ in $\mathscr{T}$ **do**
16: $\omega \leftarrow c_m(\mathbf{x_n})$ ▷ Test c_m over training instance
17: **if** $\omega = y_n$ **then**
18: $\mathscr{T} \leftarrow \mathscr{T} - \{\mathbf{x_n}\}$ ▷ Remove from $\mathscr{T}$ correctly classified instance
19: **end if**
20: **end for**
21: $C \leftarrow C \cup \{c_m\}$ ▷ Add c_m to pool
22: $m = m + 1$
23: **end while**
24: **return** C

applying the scalar product between the midpoint $p_{3,5}$ between the classes and the normalized distance vector between the centroids. These two parameters are calculated so that c_1 separates $\mathbf{r_3}$ and $\mathbf{r_5}$ halfway between them, as can be observed in Figure A.2a.

$$\mathbf{w_m} = \frac{\mathbf{r_i} - \mathbf{r_j}}{\|\mathbf{r_i} - \mathbf{r_j}\|} \tag{A.1}$$

$$b_m = -\mathbf{w_m} \cdot \frac{\mathbf{r_i} + \mathbf{r_j}}{2} \tag{A.2}$$

In steps 15 to 20, each instance in $\mathscr{T}$ is tested with c_1, and the instances correctly classified are then excluded from $\mathscr{T}$. Since c_1 correctly classifies all instances of Class 3 and Class 5, as can be seen in Figure A.2a, by the end of that iteration $\mathscr{T}$ no longer contains instances from both classes, though it still contains all instances of the other classes of the problem. In step 21, the classifier c_1 is then added to the pool C.

In the second iteration of the outer loop, $\mathscr{T}$ contains all instances of Class 1, Class 2, and Class 4, so the centroids of these classes are calculated from Step 5 to Step 7 and stored in $\mathscr{R}$. These centroids are represented by the three asterisks in Figure A.2b. The centroids chosen in Step 8 and Step 9 are $\mathbf{r_2}$ and $\mathbf{r_4}$, since they are the most distant to each other, as can be noticed in Figure A.2b, in which they are the large asterisks in red.

The weights $\mathbf{w_2}$ and bias b_2 of classifier c_2 are then calculated from Step 10 to Step 14, dividing the space between $\mathbf{r_2}$ and $\mathbf{r_4}$ right in the middle, as Figure A.2b shows. Classifier c_2 is

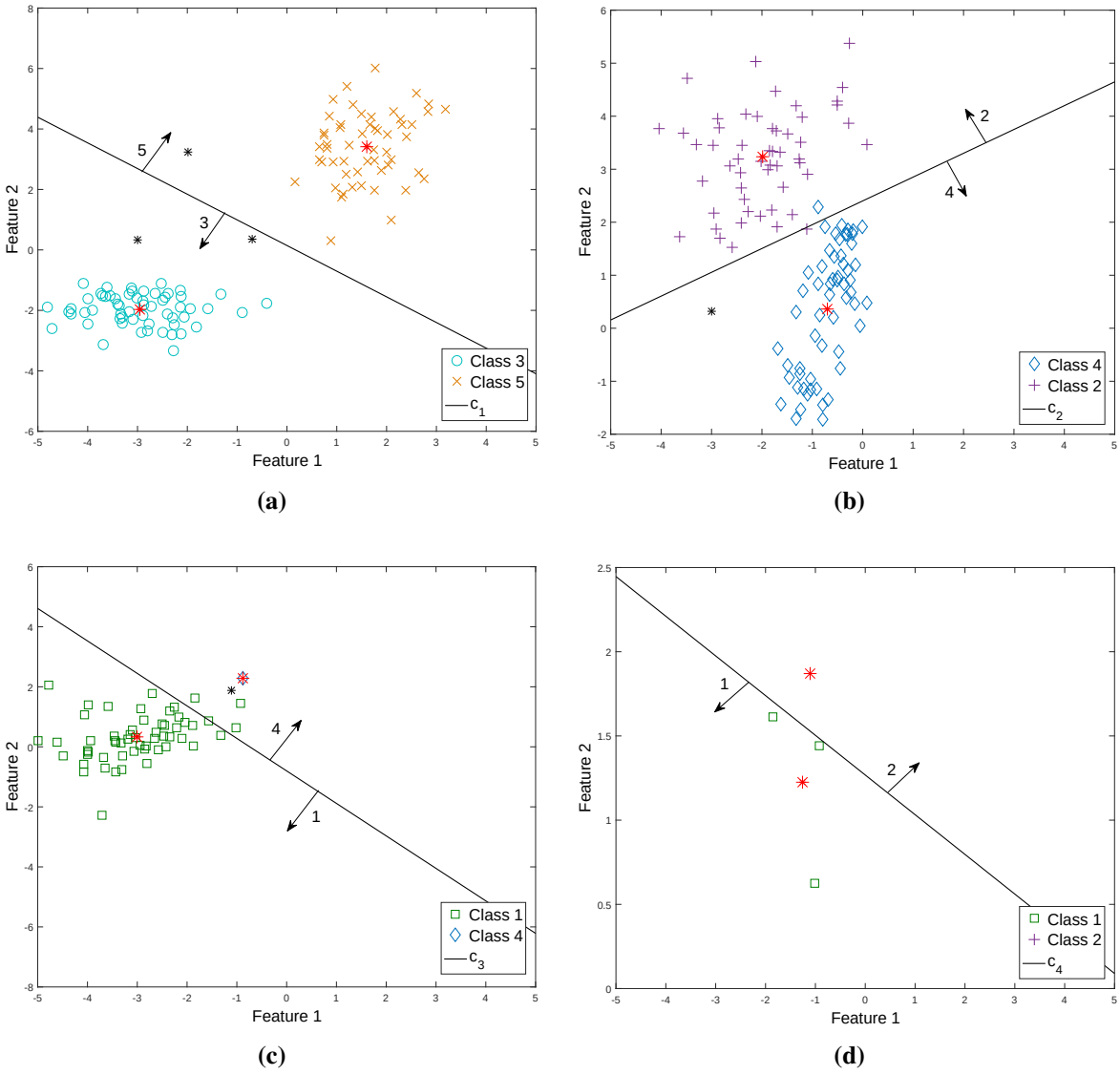


Figure A.2: Generation of hyperplanes using the SGH method over the toy problem. (a) First iteration. (b) Second iteration. (c) Third iteration. (d) Last iteration.

then used to test each instance in $\mathscr{T}$ from Step 15 to Step 20, and the instances that remain in $\mathscr{T}$ are the ones c_2 classifies incorrectly. Since Class 2 and Class 4 are not linearly separable, c_2 is not able to eliminate all instances from these classes. Classifier c_2 is added to C in step 21.

In the third iteration, $\mathscr{T}$ still has instances of Class 1, Class 2, and Class 4, so their centroids $\mathbf{r_1}$, $\mathbf{r_2}$ and $\mathbf{r_4}$ are calculated and added to $\mathscr{R}$ from Step 5 to Step 7. It can be observed that, since most of Class 2 and Class 4 instances were eliminated in the previous iteration, their centroids changed. It did not happen to the centroid of Class 1, as neither c_1 nor c_2 were able to classify Class 1 instances.

The large asterisks in Figure A.2c show that centroids $\mathbf{r_1}$ and $\mathbf{r_4}$ are the most distant ones in this iteration, with centroid $\mathbf{r_2}$ in black. Classifier c_3 is then created from Step 10 to Step 14 so that it splits the plane in a half. From Steps 15 to Step 20, each instance remaining in $\mathscr{T}$ is then tested with c_3, and the instances it correctly classifies are further eliminated from $\mathscr{T}$. It can be observed that the remaining Class 4 instance is correctly classified by c_3, so $\mathscr{T}$ only possesses Class 1 and Class 2 instances after the third iteration. In step 21, the classifier c_3 is then added to the pool.

In the fourth and last iteration, $\mathscr{T}$ contains only 4 instances, 3 of Class 1 and 1 of Class 2, as showed in Figure A.2d. Centroids $\mathbf{r_1}$ and $\mathbf{r_2}$ are calculated from Step 5 to Step 7 and chosen to calculate the weights $\mathbf{w_m}$ and bias b_4 of classifier c_4 from Step 10 to 14. Each instance in $\mathscr{T}$ is tested with c_4 in Step 15 to 20, and since it correctly classifies all 4 remaining instances, they are eliminated and $\mathscr{T}$ turns into an empty set. Classifier c_4 is then added to the pool C in step 21, and the algorithm leaves the outer loop, returning C, which contains four classifiers, in step 24.

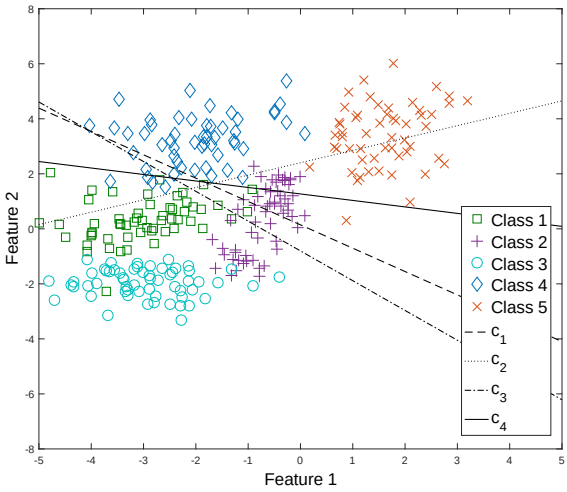


Figure A.3: Generated pool $C = \{c_1, c_2, c_3, c_4\}$ over the training set $\mathscr{T}$ of the toy problem.

Figure A.3 shows the entire training dataset with all four classifiers generated by the proposed method. The spatial disposition of the only four hyperplanes necessary to "cover" the entire dataset can be observed. It is clear, from this example, that the SGH method generates at least one competent classifier for each instance in the training set. Thus, it always guarantees an Oracle accuracy rate of 100% over the input set, as discussed in Chapter 2. This design

also allows the definition of the hit rate, for in the generation process, each training instance is assigned to a classifier, which is the one responsible for its elimination from the training set (Step 18). Moreover, it can be observed that the resulting pool contains only two-class classifiers, independently of the number of classes in the problem.