Pattern Recognition 85 (2019) 132–148

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Online local pool generation for dynamic classifier selection

Mariana A. Souza[a,*], George D.C. Cavalcanti[a], Rafael M.O. Cruz[b], Robert Sabourin[b]

[a] Centro de Informática, Universidade Federal de Pernambuco, Recife, PE, Brazil
[b] École de Technologie Supérieure, Université du Québec, Montreal, Quebec, Canada

ARTICLE INFO

Article history:
Received 14 May 2018
Accepted 1 August 2018
Available online 10 August 2018

Keywords:
Multiple classifier systems
Instance hardness
Pool generation
Dynamic classifier selection

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 classifiers 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 difficult regions of the feature space. The difficulty of a given area is determined by the estimated classification difficulty of the instances in it. 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 query samples surrounded by easy instances, a simple nearest neighbors rule is used in the proposed method. In order to identify in which cases the local pool is used in the proposed scheme, an analysis on the correlation between instance hardness and DCS techniques is performed in this work, and it is proposed the use of an instance hardness measure that conveys the degree of local class overlap near a given sample. Experimental results show that the DCS techniques were more able to select the most competent classifier for difficult instances when using the proposed local pool than when using a globally generated pool. Moreover, the proposed technique yielded significantly greater recognition rates in comparison to a Bagging-generated pool and two other global generation schemes for all DCS techniques evaluated. The performance of the proposed technique was also significantly superior to three state-of-the-art classification models and was statistically equivalent to five of them.

1. Introduction

Multiple Classifier Systems (MCS) aim to improve the overall performance of a pattern recognition system by combining numerous base classifiers [1–3]. An MCS contains three phases [4]: (1) Generation, (2) Selection and (3) Integration. In the first phase, a pool of classifiers is generated using the training data. In the second phase, a non-empty subset of classifiers from the pool is selected to perform the classification task. In the third and last phase, the selected classifiers' predictions are combined to form the final system's output. There are two possible approaches in the Selection phase: 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 static ensembles, specially on ill-defined problems [4,5], 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 [4]. The amount of classifiers singled out to label a given sample separates the DS schemes in two groups [6]: 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 [7] 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

* Corresponding author.
E-mail addresses: mas2@cin.ufpe.br (M.A. Souza), gdcc@cin.ufpe.br (G.D.C. Cavalcanti), cruz@livia.etsmtl.ca (R.M.O. Cruz), robert.sabourin@etsmtl.ca (R. Sabourin).

https://doi.org/10.1016/j.patcog.2018.08.004
0031-3203/

pool contains such classifier. Thus, the Oracle accuracy rate is the theoretical limit for DCS techniques.

The behavior of the Oracle regarding pool generation for DCS techniques was characterized in a previous work [8]. 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 is performed globally, while DCS techniques use only local data to select the most competent classifier for each instance. Thus, the difference in perspectives between the generation and the selection may hinder the DCS techniques in the selection of a competent classifier, even when the latter is guaranteed to be in the pool.

In addition to that, most works regarding DS use classical generation methods, which were designed for static ensembles [9] and therefore do not take into account the regional aspect of the competence estimation performed by the DS techniques. Thus, since local information is not considered during the generation process, the presence of local experts is not guaranteed in the final pool.

Based on these observations, it is proposed in this work an online 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 Oracle 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 in [8], 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 a pool specialized on the local region where a given unknown sample is, if that region is deemed difficult. In this context, a region is considered *difficult* if it contains an instance likely to be misclassified, as indicated by an instance hardness measure. Therefore, if a query sample is located in a difficult region of the feature space, a local pool (LP) is generated on the fly so that its classifiers fully cover the surrounding area of that specific instance. Otherwise, a simple k-Nearest Neighbors (k-NN) rule is used to label the query sample, since the classification task is less complex. Hence, whenever an unknown sample is located in a difficult region, the proposed method uses Oracle information in that area to generate locally accurate classifiers for that instance, in hopes that the best classifier among them will be more easily selected by a DCS technique than if the classifiers were generated with a global perspective.

To the best of our knowledge, there is no ensemble method designed to generate local experts for dynamic selection techniques. However, a local learning algorithm with a similar strategy to that of the proposed technique is presented in [10]. The learning algorithm consists of generating a linear classifier for each unknown instance using its surrounding training samples, and then labelling that instance with it. This model was used to analyze the trade-off between capacity and locality of the learning algorithms and its impact on their recognition rates. Although the learning algorithm provides a local perspective on the classification problem, its concept was not used in the context of producing a pool of locally accurate classifiers for DS techniques.

Other related works, such as the Mixture of Random Prototype-based Local Experts [11] and the Forest of Local Trees [12] techniques, explore the divide-to-conquer approach of MCS by locally training their base classifiers in different regions of the feature space and weighting the classifiers' votes based on the distance between the query sample and their assigned region. As opposed to these works, in which the pool generation is paired to a selection based on dynamic distance weighting, our approach consists of producing on the fly a locally accurate pool to be coupled with a DCS technique. Furthermore, the generation process of these approaches do not guarantee the presence of local experts in the vicinity of each borderline unknown sample, as the proposed method does.

Thus, with our proposed approach, we aim to find out in this work whether the presence 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 performances of the proposed scheme and of different ensemble methods that yield globally-generated pools are assessed using DCS techniques over 20 public datasets, and the results compared and analyzed. A comparative study with several state-of-the-art classification models is also performed afterwards.

This work is organized as follows: in Section 2, an analysis on instance hardness for DCS techniques is performed in order to observe the correlation between an instance hardness measure and the mistakes made by these techniques. Once this relationship is established, Section 3 presents the proposed generation method, which explores the instance hardness information obtained in the previous section. In Section 4 the proposed method is evaluated, and it is analyzed whether the use of specialist subpools in difficult regions is beneficial for DCS techniques. A comparative study with state-of-the-art classification models is also performed in Section 4. Lastly, in Section 5 the results are summarized and future works are suggested.

2. Instance hardness analysis

Hardness is an aspect inherent to a problem that hinders a classifier, or a set of classifiers, in the classification task. *Instance hardness* is then a characteristic of a sample's problem that conveys the likelihood of such sample being mislabelled by a classifier [13]. Hardness measures attempt to quantify this characteristic based on different sources of difficulty associated with data, as well as provide insights as to why some instances or problems are difficult for most learning algorithms. Many data hardness measures were proposed and also used to improve a vast number of methods in the literature [14–16].

In [17], the authors introduce a set of hardness measures obtained over the entire training set. They also identify aspects that lead to a problem, as a whole, being difficult for a classifier. The authors in [18] propose a set of hardness measures, also over the whole dataset, and use them, together with the ones from [17], to identify and remove noisy data in the training set.

In addition to characterizing the hardness of an entire set, efforts have been made to estimate the difficulty of classifying each individual instance from a dataset. In [19], the authors propose several instance hardness measures and use a subset of them in the construction of noise filter which removes potentially noisy instances among the hard ones. A hardness analysis on an instance level was done in [13], in which the authors identify the most influential causes for an instance to be hard for many diverse classification models. They also introduce more instance hardness measures and show the correlation between them and the misclassification of the classifiers analysed. Moreover, they suggest the integration of the error information of these classifiers in two different scenarios: in the training of a neural network, so that the weight of the hard instances are smaller, and in a noise filter, based on the same idea as the previous one. In [20], the authors propose two instance hardness measures that take into account misclassification costs. These measures are further used to define a measure of similarity between algorithms.

Although in [19] a set of classifiers was used to evaluate the correlation between the hardness measures and the instance hardness itself, the authors did not investigate it for DCS techniques. Though an analysis on instance hardness regarding DS techniques was performed in [21], its focus was on the comparison between these techniques and the k-NN classifier. Thus, an analysis on instance hardness in DCS context is done in this section. The purpose of such analysis is to understand the correlation between instance hardness measures and the errors made by DCS techniques, in order to identify in which cases the DCS techniques fail to choose a competent classifier for a given instance. This information will later be used to generate subpools specialized in the difficult regions of the training set.

The chosen instance hardness measure to be analyzed is the k-Disagreeing Neighbors (kDN) [13], which is defined in Eq. 1, where $\mathbf{x_i}$ is the instance being evaluated, $\mathcal{T}$ is the dataset that contains it, $kNN()$ is the k-Nearest Neighbors (k-NN) rule and k_h is the neighborhood size of the hardness measure. The kDN measure is the percentage of instances in an example's neighborhood that do not share the same label as itself. Therefore, a high kDN value means the instance is in a local overlap region, making it harder to label it. The reason for using this measure is because it denotes the most relevant source of instance hardness (overlap), according to [13]. Moreover, the kDN measure was the most correlated with instance hardness according to the same article.

$$kDN(\mathbf{x_i}, \mathcal{T}, k_h) = \frac{|\mathbf{x_j} : \mathbf{x_j} \in kNN(\mathbf{x_i}, \mathcal{T}, k_h) \wedge label(\mathbf{x_j}) \neq label(\mathbf{x_i})|}{k_h} \tag{1}$$

The pool generation technique used in this analysis was the Self-generating Hyperplanes (SGH) method [8], a simple generation scheme which yielded a similar performance as Bagging [22] for most DCS techniques. The SGH generation method is described in Algorithm 1. The input to the SGH method is only the training set

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

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

$\mathcal{T}$, and its output is the generated pool of classifiers (C). In each iteration (Step 3 to Step 15), the centroids of all classes in $\mathcal{T}$ are obtained in Step 4 and stored in $\mathcal{R}$. The two centroids in $\mathcal{R}$ 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 two-class linear classifier c_m is then tested over the training set, and the instances it correctly labels are removed from $\mathcal{T}$ (Step 7 to Step 12). Then, c_m is added to C in Step 13, and the loop is repeated until $\mathcal{T}$ 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%.

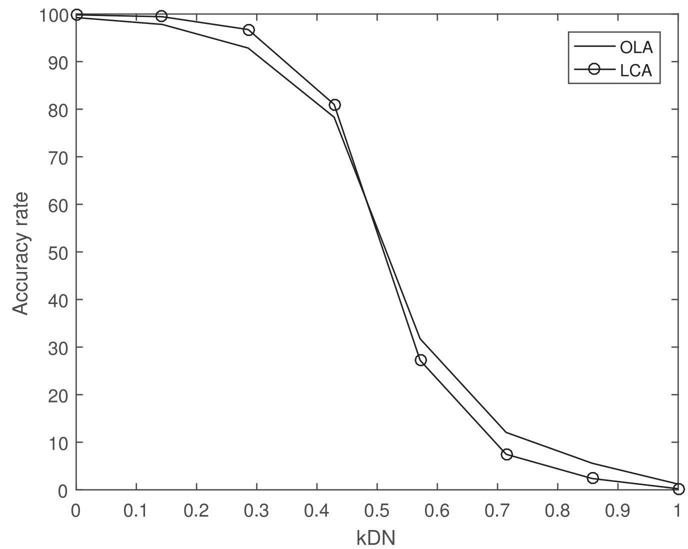


Fig. 1. Mean accuracy rate of OLA and LCA for each group of kDN value, for all datasets from Table 1. The neighborhood sizes of the DCS techniques and the kDN measure are $k_s = k_h = 7$.

In order to evaluate the correlation between the instance hardness measures and the accuracy of the DCS techniques, the hardness of each instance was computed using the entire dataset, thereby obtaining the *true* hardness value of each instance. Afterwards, the accuracy of the DCS techniques was obtained using 10 times 10-fold cross validation. The previous knowledge regarding each instance's hardness is then used to draw a relationship between this measure and the frequency at which the DCS techniques misclassifies it. That way, an evaluation of the relationship between the kDN measure and the error rate of the DCS techniques can be performed.

The datasets used in this analysis are shown in Table 1. All of them are public datasets. Eleven from the UCI machine learning repository [23], three from the Ludmila Kuncheva Collection [24] of real medical data, three from the STATLOG project [25], two from the Knowledge Extraction based on Evolutionary Learning (KEEL) repository [26] and one from the Enhanced Learning for Evolutive Neural Architectures (ELENA) project [27]. The DCS techniques used in this analysis were Overall Local Accuracy (OLA) [28] and Local Class Accuracy (LCA) [28], which were the two best performing DCS techniques in a recent survey on dynamic selection of classifiers [9]. For simplicity, the neighborhood sizes of both the kDN measure and the DCS techniques were set to $k_h = k_s = 7$.

In order to characterize the relationship between the kDN measure and the DCS techniques, all samples from each dataset were grouped by their true hardness value and the mean accuracy rate of both DCS techniques on each group was calculated. The results are summarized in Fig. 1.

It can be observed in Fig. 1 that the two DCS techniques misclassify the majority of all instances with kDN above 0.5, on average. It can also be seen a great difference between the accuracy rates of instances with kDN $\in$[0.4, 0.5] and kDN $\in$[0.5, 0.6], which is reasonable since kDN values above 0.5 mean the majority

Table 1
Main characteristics of the datasets used in the experiments.

Dataset	No. of samples	No. of features	No. of classes	Class sizes	Source
Adult	48,842	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

of an instance's neighbors belong to a different class than its own class. This result shows the correlation between the kDN measure and the classification difficulty by the DCS techniques, which was somewhat expected since the measure and the techniques operate locally using the same mechanism of selecting the k nearest neighbors. Moreover, Fig. 1 shows that LCA correctly classifies a greater amount of the easiest instances ($kDN \leq 0.5$) than OLA, though it struggles more to correctly label the hardest instances ($kDN \geq 0.5$), on average.

3. The proposed method

In the previous section, it was shown the DCS techniques struggle to select a competent classifier for instances in regions with overlap between the problem's classes. Moreover, since the DCS techniques rely only on a small region, an instance's neighborhood, in order to select the most competent classifier for this instance, a global perspective in the search for a promising pool for DCS could be inadequate in such cases [8].

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. If the unknown instance's Region of Competence (RoC) is located in a difficult region, 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 a simple nearest neighbors 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 static ensembles [9] 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.

An overview of the proposed method is described in more detail in Section 3.1. Then, a step-by-step analysis of the proposed method is presented using a 2D toy problem in Section 3.2.

3.1. Overview

The proposed technique is divided into two phases:

1. The *offline* phase, in which the hardness estimation of the training instances is performed. The hardness value of the training samples is used to identify the difficult regions of the feature space.
2. The *online* phase, in which the RoC of the query sample is evaluated using a hardness measure in order to identify if the local area is difficult. If it is not, the sample is labelled using a nearest neighbors rule. Otherwise, a pool composed of the most locally accurate classifiers in that region, as indicated by a DCS technique, is generated and used to label the unknown instance.

An overview of the proposed techniques' phases is depicted in Fig. 2, in which $\mathcal{T}$ is the training set, H is the set of hardness estimates, $\mathbf{x_q}$ is the query sample, θ_q is its RoC, k_s 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 offline phase (Fig. 2a), the hardness of each instance $\mathbf{x_n} \in \mathcal{T}$ is estimated using a hardness measure, and its value is stored in H, to be later used in the online phase. The online phase, in which the query sample $\mathbf{x_q}$ is labelled, is performed in three steps: RoC evaluation, local pool generation and generalization, as shown in Fig. 2b.

In the RoC evaluation step, the k_s 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 dynamic selection dataset (DSEL), which is a set of labelled samples used for RoC definition in DS techniques [9], was not used in the proposed method because the SGH method did not present overfitting when used for RoC definition [8]. Then, the instances that compose θ_q are analyzed. If all of them are not in an overlap region, that is, they all have hardness estimate $H = 0$, 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_s. However, if there is at least one instance in θ_q located in an identified class overlap area, the query sample's RoC is considered to be a difficult region. Thus, the local pool LP containing M classifiers is generated in the second step and used to label $\mathbf{x_q}$ in the generalization step via majority voting. The local pool generation step is explained next.

Fig. 3 shows the generation procedure of the local pool LP. The pool size M of the local pool is an input parameter. The other

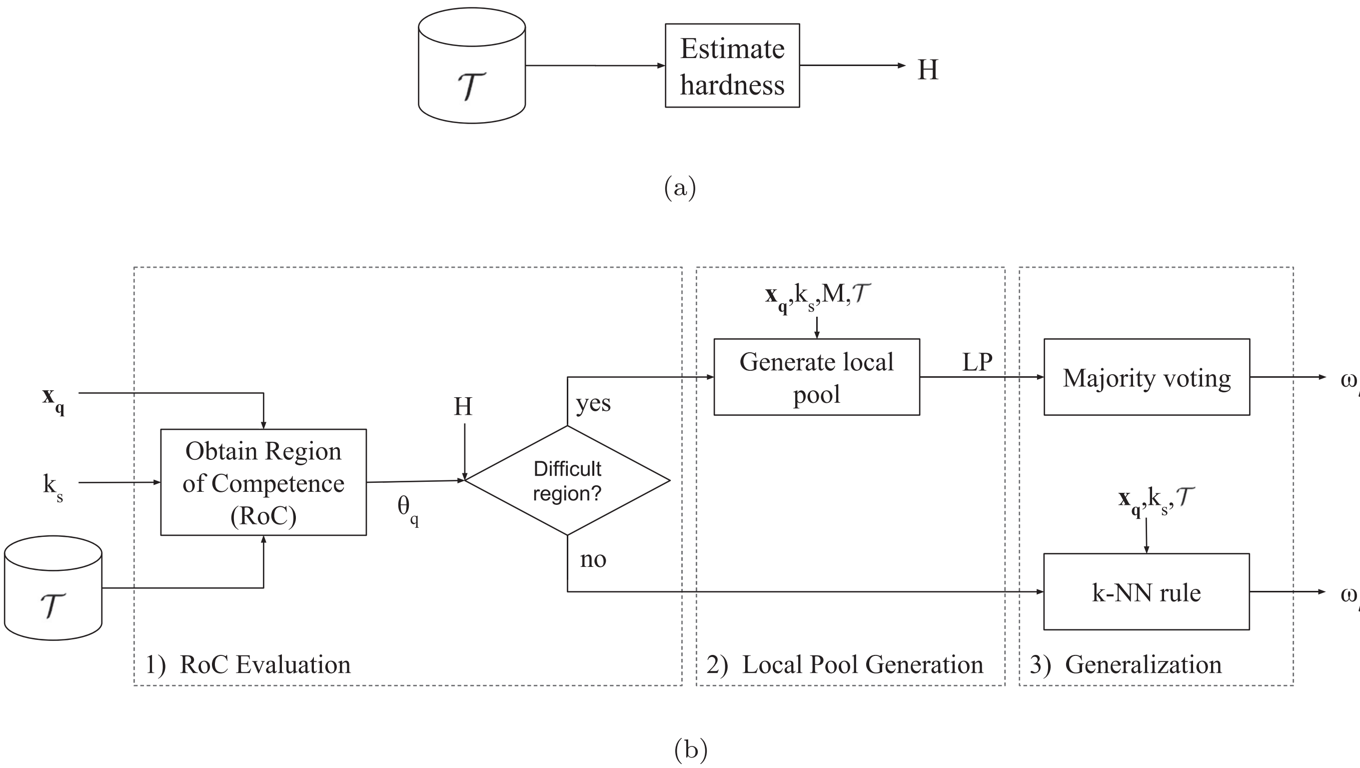


Fig. 2. Overview of the (a) offline and (b) online phases of the proposed method. $\mathcal{T}$ is the training set, H is the set of hardness estimates, $\mathbf{x_q}$ is the query sample, θ_q is its Region of Competence (RoC), k_s is the size of θ_q, LP is the local pool, M is the pool size of LP and ω_l is the output label of $\mathbf{x_q}$. In the offline phase, the hardness value of all instances in $\mathcal{T}$ is estimated and stored in H. In the online phase, θ_q is first obtained and evaluated based on the hardness values in H. If it only contains easy instances, the k-NN rule is used to label $\mathbf{x_q}$ in the last step. Otherwise, the local pool is generated in the second step, and $\mathbf{x_q}$ is labelled via majority voting of the classifiers in LP in the third step.

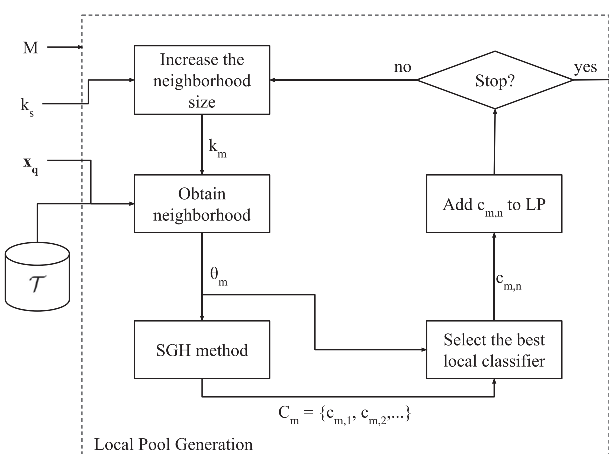


Fig. 3. Local pool generation step. The inputs to the generation scheme are the training set $\mathcal{T}$, the query sample $\mathbf{x_q}$, the size k_s of the query sample's RoC and the local pool size M. The output is the local pool LP. In the m-th iteration, the query sample's neighborhood θ_m of 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.

inputs are the training set ($\mathcal{T}$), the query sample ($\mathbf{x_q}$) and the query sample's RoC size (k_s). 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 (Algorithm 1 from Section 2). 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.

The pseudocode of the offline phase of the proposed method is shown in Algorithm 2. Its inputs are the training set $\mathcal{T}$ and the

Algorithm 2 Offline phase (proposed technique).

Input: $\mathcal{T}, k_h$ ▷ Training dataset and kDN neighborhood size
Output: H ▷ Estimated hardness values
1: **for** every $\mathbf{x_i}$ in $\mathcal{T}$ **do**
2: $\quad H(i) \leftarrow kDN(\mathbf{x_i}, \mathcal{T}, k_h)$ ▷ Calculate hardness (Equation 1)
3: **end for**
4: **return** H

kDN parameter k_h, which denotes the neighborhood size of the hardness estimate. From Step 1 to Step 3, the hardness of each instance $\mathbf{x_i} \in \mathcal{T}$ is calculated and stored in H, which is then returned in Step 4.

The online phase, on the other hand, is described in more detail in Algorithm 3. Its inputs are the query sample $\mathbf{x_q}$, training set $\mathcal{T}$, the set of hardness estimates H, the RoC size k_s and the local pool size M. In Step 1, the query sample's RoC θ_q is obtained by selecting the k_s closest samples to $\mathbf{x_q}$ in the training set. The RoC

Algorithm 3 Online phase (proposed technique).

Input: $\mathbf{x_q}, \mathcal{T}, H$ ▷ Query sample, training set and hardness estimates
Input: k_s, M ▷ RoC size and pool size of local pool LP
Output: ω_l ▷ Output label of $\mathbf{x_q}$

1) RoC Evaluation
1: $\theta_q \leftarrow obtainRoC(\mathbf{x_q}, k_s, \mathcal{T})$ ▷ Obtain the query instance's RoC
2: **if** $\{\exists \mathbf{x_i} \in \theta_q | H(i) > 0\}$ **then**

2) Local Pool Generation
3: $LP \leftarrow \{\}$ ▷ Local pool initially empty
4: **for** every m in $\{1, 2, ..., M\}$ **do**
5: $k_m \leftarrow k_s + 2 \times (m-1)$ ▷ Increase neighborhood size by 2
6: $\theta_m \leftarrow obtainNeighborhood(\mathbf{x_q}, k_m, \mathcal{T})$ ▷ Obtain neighborhood of $\mathbf{x_q}$
7: $C_m \leftarrow generatePool(\theta_m)$ ▷ Generate local subpool C_m
8: $c_{m,n} \leftarrow selectClassifier(\mathbf{x_q}, \theta_m, C_m)$ ▷ Select best classifier in C_m
9: $LP \leftarrow LP \cup \{c_{m,n}\}$ ▷ Add $c_{m,n}$ to LP
10: **end for**

3) Generalization
11: $\omega_l \leftarrow majorityVoting(\mathbf{x_q}, LP)$ ▷ Label $\mathbf{x_i}$ with majority voting on LP
12: **else**
13: $\omega_l \leftarrow kNN(\mathbf{x_q}, k_s, \mathcal{T})$ ▷ Label query sample using k-NN rule
14: **end if**
15: **return** ω_l

is then evaluated in Step 2. If all instances in θ_q are not located in a difficult region, that is, their hardness value is zero, 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_s and returned in Step 15.

However, if there is one instance $\mathbf{x_i}$ from θ_q whose hardness estimate $H(i)$ is above zero, the region is considered a difficult one and 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.

3.2. 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 [29] was chosen for its complex borders. Since the P2 problem has no overlap between the classes, noise was added to the original distribution by randomly changing the labels of the samples near the class borders. The dataset used in this analysis contains 1000 instances, 75% of which were used for training and the rest for test. The parameters used in this demonstration were $k_h = k_s = 7$ and $M = 7$. The method used for selecting the query instance's neighborhood in *getNeighborhood*() (Step 6 of Algorithm 3) for this example was the regular k-NN, and the DCS technique used to select the most competent classifier (Step 8 of Algorithm 3) was OLA.

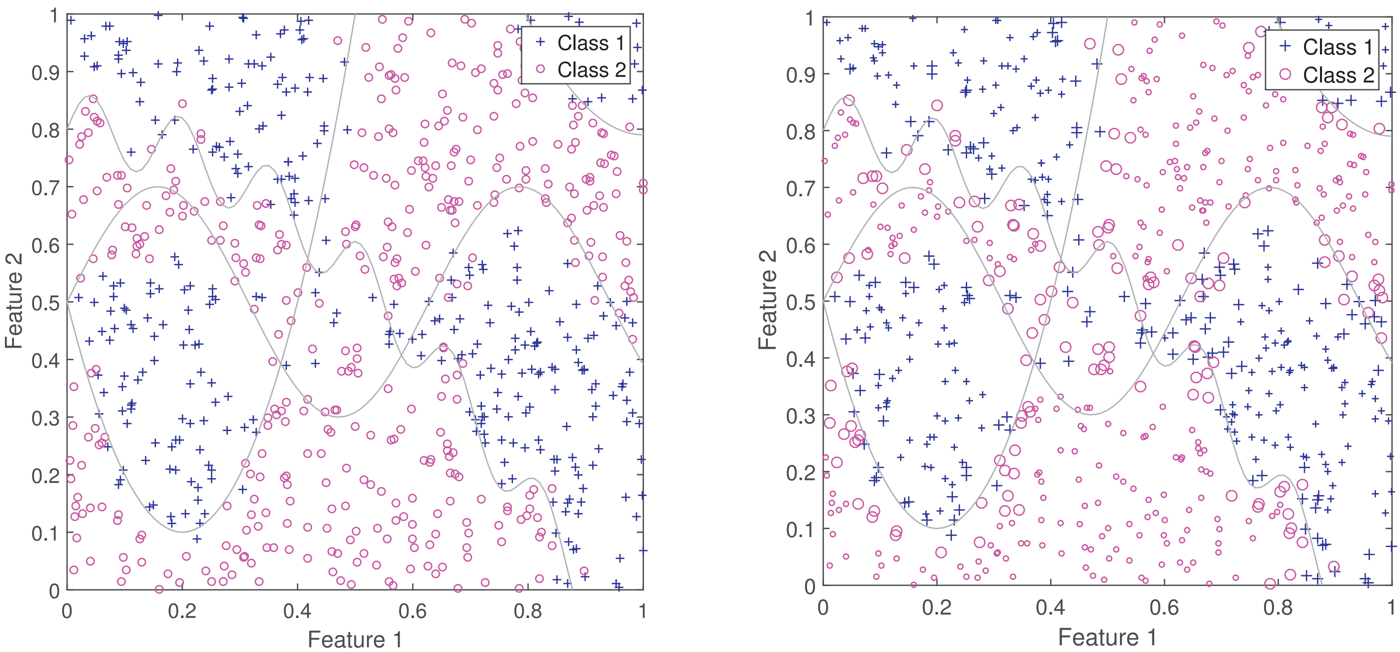
The P2 Problem training set used in this analysis is shown in Fig. 4a, with its theoretical decision boundaries in grey. The hardness estimation in Step 1 to Step 3 of Algorithm 2 separates the training instances with estimated hardness above zero, that is, the instances closer to the border between classes, and the remaining ones. Fig. 4b shows the training instances closer to the borders (large markers) and the instances with $H(i) = 0$ (small markers).

Two scenarios of the proposed scheme's online phase can be observed in Fig. 5a and Fig. 5b. In the first, the input query instance $\mathbf{x_q}$ of Algorithm 3 belongs to Class 2. The query sample's RoC θ_q is obtained selecting its k-nearest neighbors over the training set $\mathcal{T}$ in Step 1. In this case, since all instances in θ_q have estimated hardness $H_i = 0$, represented in Fig. 5a by small markers, $\mathbf{x_q}$ is considered to be in an easy 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_s. Then, the query sample's label is returned in Step 15. In this case $\omega_l = 2$ since all k_s neighbors of $\mathbf{x_q}$ belong to this class.

In the second scenario, shown in Fig. 5b, the query instance $\mathbf{x_q}$ of Algorithm 3 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. The hardness estimate H_i of each neighbor is verified in Step 2 of Algorithm 3, and since at least one of them is above zero, depicted in Fig. 5b in large markers, 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.

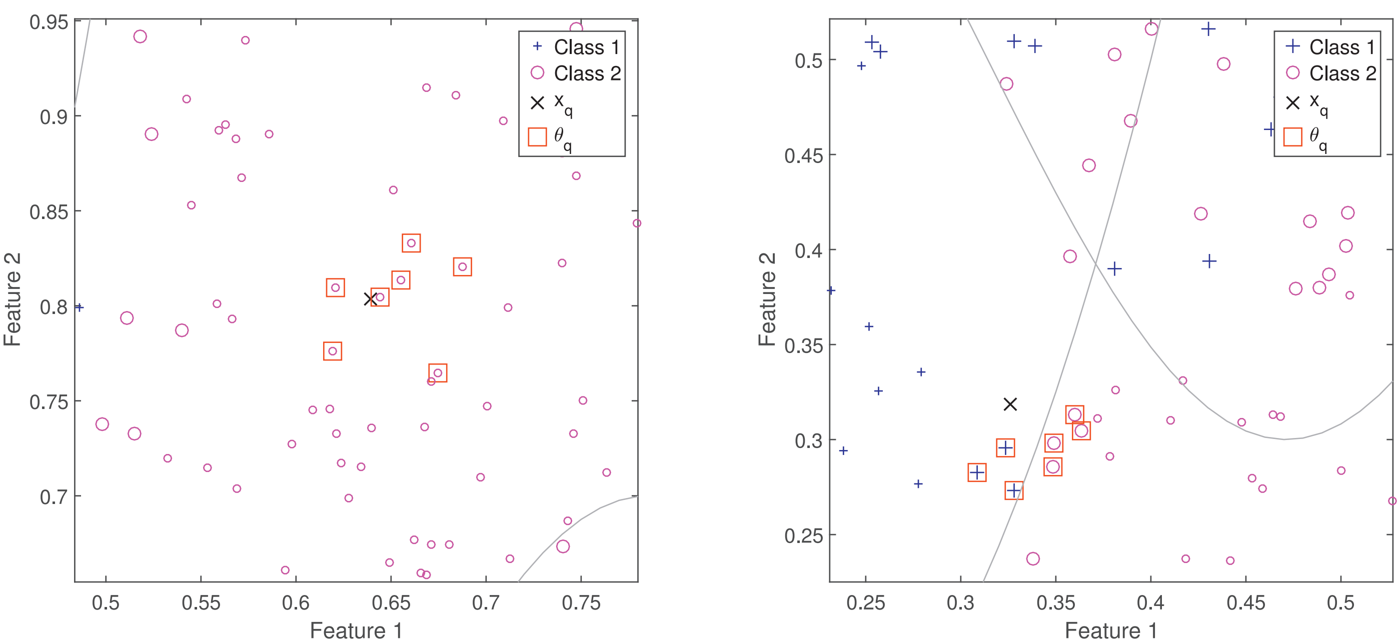
In the first iteration, the neighborhood size k_1 is set to 7 in Step 5, and then the k_1 nearest neighbors of $\mathbf{x_q}$ are selected to compose the query sample's neighborhood θ_1 in Step 6. The local subpool C_1 is then generated using θ_1 as the input dataset to the SGH method. The resulting pool, which guarantees an Oracle accuracy rate of 100% in θ_1, is shown in Fig. 6a, 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



(a) (b)

Fig. 4. P2 Problem dataset, with theoretical decision boundaries in grey. The training set is depicted in (a), and in (b) the same set is shown with hard instances in large markers and easy instances in small ones.



(a) (b)

Fig. 5. Two different scenarios of the online phase. In (a), the query instance $\mathbf{x_q}$ belongs to Class 2. Since all instances in its neighborhood θ_q are easy (small markers), the k-NN rule is used to label $\mathbf{x_q}$. On the other hand, all instances in the query sample's neighborhood θ_q in (b) are deemed hard (large markers). Thus, the local pool *LP* will label the query instance $\mathbf{x_q}$, which belongs to Class 1.

$k_2 = 9$ instances, as shown in Fig. 6b. 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 Fig. 6c 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 Fig. 6d, 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.

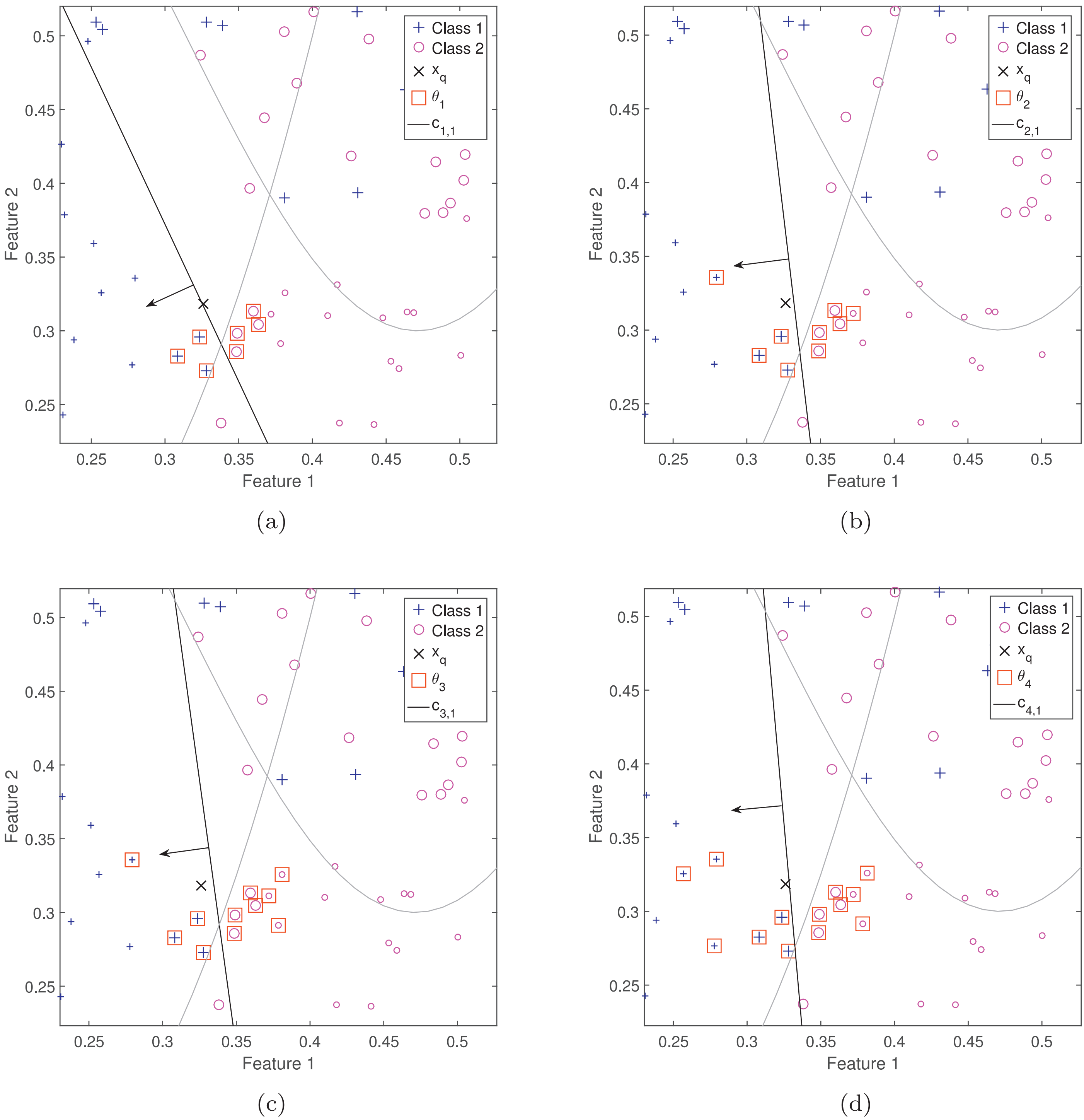


Fig. 6. Local pool generation. (a) First, (b) second, (c) third, (d) fourth, (e) fifth, (f) sixth and (g) seventh iteration of the method, with its respective neighborhoods (θ_m) and generated local subpools C_m formed by the depicted classifiers ($c_{m,\,k}$). The arrows indicate in which part of the 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).

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 Fig. 6e. 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 Fig. 6f. In Step 8, both classifiers are evaluated over θ_6 using a DCS technique, OLA in this 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 Fig. 6g, 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

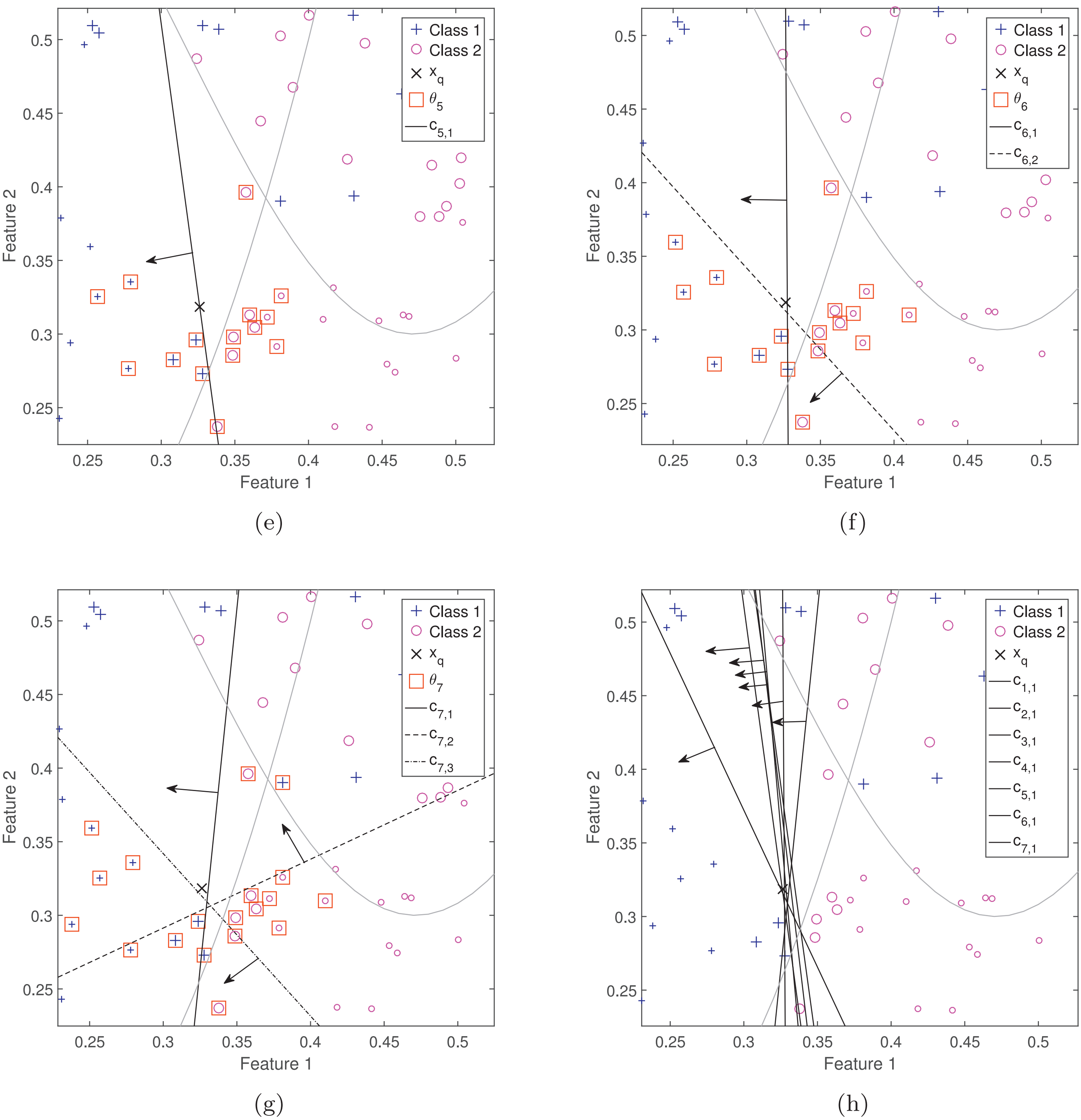


Fig. 6. Continued

the local pool, completing the generation process of *LP*, depicted in Fig. 6h.

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 2 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 $\mathbf{x_q}$.

Table 2
Majority voting of the classifiers from LP for the query instance from Fig. 5b.

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

4. Experiments

In order to analyse and evaluate the performance of the proposed method, experiments were conducted over the 20 datasets described in Table 1. All methods in the comparative study were evaluated using 20 replications of each dataset. 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 [8] and local scope, the training set was used as the DSEL set. In the comparative study, however, the methods that use a DS technique were tested using a pool of 100 Perceptrons obtained using Bagging [22], as it is often done in DS works [5,30]. For these configurations, the validation set was used as the DSEL in order to avoid overfitting, so one third of the training set was randomly selected to compose the DSEL set.

This section is organized as follows. A comparative study with regards to DCS techniques is performed in Section 4.1, with the purpose of analyzing whether the use of locally generated pools is in fact advantageous in this context. In Section 4.2, the performances of the proposed method and state-of-the-art models, including single models, static ensembles and DS techniques, are also compared and analyzed. Lastly, the computational complexity of the proposed method and the compared models are discussed in Section 4.3.

4.1. Comparison with DCS techniques

In this section, 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.

The DCS techniques chosen to evaluate the methods in these experiments were OLA, LCA and MCB, since they outperformed the other evaluated DCS techniques in [9]. The RoC size k_s for each of the DCS techniques is set to 7, since it yielded the best results in [4].

The parameters of the proposed method were set to $k_s = k_h = 7$ and $M = 7$. Moreover, the proposed scheme was tested with two neighborhood acquisition methods: the *LP* configuration and the LP^e configuration. In the first, the *getNeighborhood*() method from Algorithm 3 (Step 6) used was the regular k-NN. In the second configuration, the *getNeighborhood*() procedure used was a version of the k-Nearest Neighbor Equality (k-NNE) [31] in which the returned neighborhood contains an equal amount of instances from all classes, given that these classes are present in the query instance's RoC θ_q from Algorithm 3 (Step 1).

The performance of the proposed method with regards to the DCS techniques is compared to three globally generated pool configurations. The baseline method used in the comparison is a Bagging-generated pool composed of 100 classifiers. The SGH method over the entire training set is also included in the comparative study, since it provides another global approach for generating classifiers. The pool generated by this technique is referenced as the global pool (GP). Lastly, another related method, though it is not a generation one, is used in the comparison with DCS techniques: the Frienemy Indecision Region Dynamic Ensemble Selection (FIRE-DES) framework [32].

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 RoC 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.

In the FIRE-DES context, an unknown sample in an indecision region has, by definition, a hardness value greater than zero, since at least one of its neighbors belongs to a different class, regardless of its label. In the proposed method, such an instance is labelled using the local pool, which is guaranteed to contain classifiers that cross the query sample's RoC due to its generation procedure (Fig. 3). Thus, the same idea of using only locally accurate classifiers for instances in overlap regions from the FIRE-DES framework indirectly applies to the proposed method as well. Therefore, the FIRE-DES framework, coupled with the chosen DCS techniques, is also included in the comparative study that follows. The pool used by this framework in the experiments is the same as the one from the Bagging configuration, which contains 100 globally generated classifiers.

The performance of the chosen configurations with regards to DCS techniques is evaluated in memorization, using the hit rate measure, in Section 4.1.1, and in generalization, using the accuracy rate over the datasets from Table 1, in Section 4.1.2.

4.1.1. Performance in memorization

The proposed method was evaluated in memorization using the hit rate [8], which is a metric derived from the SGH method that 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 configurations are calculated the same way as the *GP* configuration, with the only difference being for instances not in difficult regions. In this case, the accuracy rate of the k-NN rule 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 3 shows the mean hit rate for OLA, LCA and MCB of the three configurations that use the SHG method. In comparison with the *GP* configuration, both *LP* and LP^e configurations obtained a greater overall hit rate for both DCS techniques. More specifically, for the LP^e configuration, nearly half of the datasets yielded a hit rate above 90%, whilst for the *GP* only two of them at most obtained a similar hit rate for the three DCS techniques.

Moreover, a Wilcoxon signed rank test with a significance level of $\alpha = 0.05$ was performed between the hit rate results for the *GP* and the two proposed configurations. It can be observed, from the Wilcoxon rows, that the proposed configurations yielded a significantly greater hit rate than the global configuration for LCA, and, in particular, the LP^e configuration obtained a significant increase in the hit rate for OLA and MCB as well. This suggests that the use of the local pools indeed facilitates the DCS technique in choosing the correct classifier for instances in difficult regions.

Table 3
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, LP* and *LP^e^* configurations. The row Wilcoxon shows the result of a Wilcoxon signed rank test over the mean hit rates of the GP configuration and the two proposed configurations. The significance level was $\alpha = 0.05$, and the symbols + , - and $\sim$ indicate the method is significantly superior, inferior or not significantly different, respectively. Best results are in bold.

Dataset	GP	LP	LP^e
(a)			
Adult	86.91 (0.87)	86.10 (1.13)	**91.41 (0.87)**
Blood	79.59 (0.51)	73.11 (1.42)	**80.96 (0.88)**
CTG	92.50 (0.59)	92.66 (0.83)	**92.95 (0.92)**
Faults	**76.88 (1.26)**	74.85 (0.70)	70.92 (0.95)
German	71.05 (1.44)	89.25 (0.56)	**91.86 (0.34)**
Glass	**76.21 (1.98)**	63.07 (0.75)	69.71 (1.74)
Haberman	76.26 (1.10)	69.14 (2.45)	**77.55 (0.93)**
Heart	84.06 (1.92)	90.06 (1.24)	**92.64 (0.54)**
Ionosphere	86.46 (1.48)	87.26 (1.59)	**87.97 (1.19)**
Laryngeal1	84.75 (2.07)	87.16 (0.89)	**89.82 (1.22)**
Laryngeal3	74.81 (2.95)	**79.63 (1.32)**	76.44 (1.20)
Liver	67.22 (1.40)	79.16 (1.28)	**83.92 (0.89)**
Mammographic	82.72 (0.64)	70.32 (1.61)	**84.02 (0.84)**
Monk2	85.77 (3.60)	95.85 (0.32)	**96.47 (0.33)**
Phoneme	87.40 (0.46)	88.56 (0.23)	**90.09 (0.29)**
Pima	75.64 (1.55)	83.00 (0.63)	**87.65 (0.28)**
Sonar	80.00 (3.62)	92.48 (1.11)	**93.81 (1.13)**
Vehicle	76.14 (1.49)	**78.25 (1.00)**	77.32 (0.70)
Vertebral	82.39 (2.14)	87.42 (1.27)	**89.38 (1.02)**
Weaning	83.45 (1.33)	94.82 (0.45)	**94.97 (0.35)**
Average	80.51	83.11	**86.00**
Wilcoxon	n/a	$\sim$	+
(b)			
Adult	86.77 (0.92)	89.99 (0.60)	**91.27 (0.89)**
Blood	80.20 (0.35)	79.43 (1.35)	**80.27 (1.20)**
CTG	92.63 (0.44)	**94.03 (0.27)**	93.30 (0.34)
Faults	**76.84 (1.01)**	74.85 (0.39)	71.25 (0.61)
German	75.75 (1.35)	90.01 (0.63)	**91.90 (0.32)**
Glass	**77.95 (1.92)**	67.73 (1.37)	69.17 (2.00)
Haberman	**76.61 (1.46)**	74.76 (1.84)	76.60 (1.04)
Heart	83.86 (2.40)	91.85 (0.79)	**92.77 (0.68)**
Ionosphere	87.34 (1.53)	**92.32 (0.77)**	92.11 (0.75)
Laryngeal1	84.81 (2.38)	88.55 (0.94)	**89.83 (1.14)**
Laryngeal3	73.98 (1.99)	**80.28 (1.72)**	78.37 (2.01)
Liver	70.62 (2.91)	79.69 (1.26)	**84.10 (1.01)**
Mammographic	**82.83 (1.54)**	80.82 (1.15)	82.07 (0.85)
Monk2	91.82 (3.61)	**96.63 (0.27)**	96.44 (0.32)
Phoneme	89.48 (0.44)	**91.93 (0.32)**	91.31 (0.23)
Pima	76.02 (1.67)	84.07 (0.48)	**87.68 (0.27)**
Sonar	83.46 (3.45)	93.37 (1.08)	**94.27 (0.96)**
Vehicle	**77.98 (1.57)**	77.11 (0.78)	76.60 (0.77)
Vertebral	84.33 (2.32)	**89.87 (1.02)**	89.85 (1.00)
Weaning	84.38 (1.72)	**95.17 (0.51)**	95.07 (0.36)
Average	81.88	85.63	**86.21**
Wilcoxon	n/a	+	+
(c)			
Adult	87.14 (0.73)	87.35 (1.19)	**89.76 (0.59)**
Blood	79.61 (0.51)	74.17 (1.48)	**79.67 (0.81)**
CTG	**92.49 (0.63)**	92.14 (0.44)	92.00 (0.37)
Faults	76.87 (1.26)	**77.16 (0.84)**	76.73 (0.49)
German	71.23 (1.47)	90.90 (0.61)	**91.93 (0.58)**
Glass	**76.27 (1.99)**	66.54 (1.44)	69.98 (1.61)
Haberman	76.35 (1.10)	69.03 (1.65)	**76.77 (1.59)**
Heart	83.96 (1.72)	89.10 (1.31)	**91.85 (1.17)**
Ionosphere	86.43 (1.43)	88.65 (0.77)	**92.14 (0.75)**
Laryngeal1	84.75 (1.93)	86.72 (1.11)	**88.71 (0.90)**
Laryngeal3	74.85 (2.90)	78.97 (1.48)	**80.79 (1.37)**
Liver	67.34 (1.28)	79.60 (1.24)	**83.68 (1.10)**
Mammographic	**82.68 (0.73)**	71.21 (1.48)	82.56 (1.04)
Monk2	86.67 (4.48)	**95.69 (0.28)**	95.35 (0.31)
Phoneme	87.40 (0.47)	89.12 (0.27)	**90.10 (0.18)**
Pima	75.82 (1.83)	83.52 (0.79)	**87.54 (0.31)**
Sonar	80.19 (3.63)	92.51 (0.91)	**93.89 (1.03)**
Vehicle	76.20 (1.51)	**77.90 (0.78)**	76.05 (0.76)
Vertebral	82.39 (2.19)	88.06 (1.27)	**89.63 (1.24)**
Weaning	83.36 (1.20)	**94.06 (0.51)**	93.88 (0.72)
Average	80.60	83.62	**86.15**
Wilcoxon	n/a	$\sim$	+

4.1.2. Performance in generalization

The mean percentage of test instances with true hardness value above zero is depicted in the *True* bars of Fig. 7 for all datasets. The true hardness value is obtained observing the neighborhood of each test instance over the entire dataset. The mean percentage of test instances deemed hard by the proposed method is also depicted in Fig. 7 (*Estimated* bars). That is, the *Estimated* bars show the frequency at which the proposed method generated and used local pools, whilst the *True* bars show the actual proportion of instances in difficult regions for each problem. It can be observed that, though the proportion of instances in difficult 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 a difficult region and thus generated a local pool to handle them.

The averaged value of the true and estimated percentage of hard instances is also indicated in Fig. 7 by the *true* and *est* lines, respectively. It can be observed that the mean percentage of test instances truly located near the borders was 65.04%, while the proposed method generated local pools for 64.46% of the test instances, on average.

The accuracy rate of Bagging, FIRE-DES, the GP configuration and the proposed configurations were evaluated with OLA, LCA and MCB, and the results are presented in Table 4. It can be observed that the proposed configurations (*LP* and *LP^e^*) obtained an average accuracy rate greater than Bagging, FIRE-DES and the *GP* configuration for all DCS techniques. A Wilcoxon signed-rank test with a significance level of $\alpha = 0.05$ was also performed for comparing the accuracy of the evaluated techniques. We chose the Wilcoxon signed-rank test due to its robustness, as its result do not depend on the algorithms originally included in the comparison [33]. It can be observed from the Wilcoxon rows of Table 4 a that both proposed configurations were significantly superior to Bagging, and the *LP* configuration significantly outperformed the *GP* configuration using LCA.

Also, it can be observed in Table 4 that, for two-class problems with high percentage of difficult instances such as German, Liver, Monk2, Pima and Sonar (Fig. 7), the use of local pools fairly increased the accuracy rate in comparison with the other configurations for the three DCS techniques, further suggesting the advantage of such pools over the global one for instances in difficult regions.

4.1.3. Discussion

From Table 4, it can be observed that the two evaluated configurations of the proposed method yielded quite distinct results: the *LP* configuration always surpassed, by far most of the times, the *LP^e^* configuration for the multi-class problems for both DCS techniques. The reason for this difference in performance lies in the neighborhood selection schemes used in the online phase of the proposed method, as it can be observed in Fig. 8, in which two multi-class toy problems are depicted.

In Fig. 8a, the neighborhood θ_1 of the query instance $\mathbf{x_q}$ was obtained using the regular k-NN rule. 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. Fig. 8b 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 Fig. 8b 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,

Table 4
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, the *LP* configuration and the LP^e configuration. The row *Wilcoxon (Bagging)* shows the result of a Wilcoxon signed rank test over the mean accuracy rates of Bagging and each remaining method. The same test was performed in comparison with the FIRE-DES configuration and the *GP* configuration (rows *Wilcoxon (FIRE)* and *Wilcoxon (GP)*, respectively). The significance level was $\alpha = 0.05$, and the symbols + , - and ∼ indicate the method is significantly superior, inferior or not significantly different, respectively. In LP^{mc}, the LP^e is used for 2-class problems, while the *LP* is used for multi-class ones. Best results are in bold.

Dataset	Bagging [22]	FIRE-DES [32]	GP	LP	LP^e	LP^{mc}
(a)						
Adult	84.97 (2.50)	83.84 (3.37)	**88.15 (2.93)**	83.32 (3.63)	87.75 (2.17)	87.75 (2.17)
Blood	75.48 (2.31)	69.28 (3.90)	75.53 (1.14)	73.78 (2.80)	**77.21 (1.57)**	**77.21 (1.57)**
CTG	88.83 (1.26)	88.50 (1.36)	90.24 (0.77)	**92.20 (1.10)**	89.98 (0.80)	**92.20 (1.10)**
Faults	66.52 (1.65)	65.33 (1.95)	71.91 (1.60)	**72.40 (1.29)**	65.93 (1.29)	**72.40 (1.29)**
German	70.34 (1.88)	68.56 (1.89)	70.04 (2.35)	72.16 (2.12)	**74.04 (1.88)**	**74.04 (1.88)**
Glass	61.42 (4.22)	59.43 (5.66)	66.79 (4.17)	**67.83 (3.94)**	60.75 (2.17)	**67.83 (3.94)**
Haberman	70.79 (5.12)	66.78 (4.81)	71.58 (5.24)	68.95 (4.19)	**72.43 (2.24)**	72.43 (2.24)
Heart	82.35 (3.44)	82.21 (4.32)	**86.62 (2.18)**	81.99 (4.79)	83.68 (3.27)	83.68 (3.27)
Ionosphere	86.70 (3.04)	86.53 (2.84)	87.16 (2.76)	91.76 (1.95)	**91.99 (2.16)**	**91.99 (2.16)**
Laryngeal1	**82.92 (3.54)**	81.89 (5.62)	80.38 (4.26)	77.74 (5.53)	80.57 (5.87)	80.57 (5.87)
Laryngeal3	70.73 (5.79)	66.46 (4.73)	**72.25 (1.71)**	71.74 (2.73)	64.66 (1.34)	71.74 (2.73)
Liver	64.59 (4.18)	65.00 (3.85)	58.37 (3.53)	60.12 (4.99)	**67.21 (1.67)**	**67.21 (1.67)**
Mammographic	82.57 (2.02)	79.06 (3.62)	**82.60 (2.47)**	75.53 (2.60)	82.36 (1.81)	82.36 (1.81)
Monk2	87.87 (3.97)	87.92 (3.13)	86.20 (3.74)	**94.91 (0.97)**	94.07 (0.76)	94.07 (0.76)
Phoneme	80.31 (0.68)	76.02 (1.17)	86.74 (0.73)	**88.58 (0.63)**	86.60 (0.67)	86.60 (0.67)
Pima	72.40 (2.73)	68.78 (3.03)	72.29 (2.39)	72.03 (1.68)	**76.77 (2.26)**	**76.77 (2.26)**
Sonar	80.96 (4.04)	79.90 (4.02)	80.00 (3.33)	**83.27 (5.79)**	75.00 (4.14)	75.00 (4.14)
Vehicle	73.61 (2.56)	**74.43 (1.95)**	70.09 (2.57)	74.39 (2.09)	69.74 (1.66)	74.39 (2.09)
Vertebral	85.38 (4.04)	84.49 (4.70)	81.41 (2.06)	85.19 (2.14)	**86.47 (2.65)**	**86.47 (2.65)**
Weaning	77.50 (3.36)	77.57 (3.40)	78.68 (3.71)	**86.05 (1.73)**	85.66 (2.37)	85.66 (2.37)
Average	77.31	75.59	77.85	**78.69**	78.64	80.02
Wilcoxon (Bagging)	n/a	-	∼	∼	∼	+
Wilcoxon (FIRE)	+	n/a	∼	+	+	+
Wilcoxon (GP)	∼	∼	n/a	∼	∼	+
(b)						
Adult	86.88 (3.17)	85.72 (3.59)	**87.40 (2.82)**	84.71 (3.73)	87.11 (2.40)	87.11 (2.40)
Blood	76.14 (2.24)	71.06 (3.44)	75.74 (1.04)	**77.95 (2.51)**	76.89 (1.67)	76.89 (1.67)
CTG	88.38 (1.37)	88.18 (1.36)	90.30 (0.84)	**92.22 (1.10)**	90.58 (0.39)	**92.22 (1.10)**
Faults	66.00 (1.69)	65.67 (2.23)	71.99 (1.53)	**73.20 (1.22)**	66.28 (1.15)	**73.20 (1.22)**
German	70.66 (2.06)	70.40 (1.22)	70.84 (1.87)	72.88 (2.37)	**74.08 (1.84)**	**74.08 (1.84)**
Glass	56.13 (5.47)	56.04 (5.41)	**69.43 (3.33)**	67.45 (2.73)	62.55 (4.83)	67.45 (2.73)
Haberman	73.03 (3.58)	69.87 (4.87)	71.05 (1.91)	70.79 (3.71)	**72.11 (2.12)**	**72.11 (2.12)**
Heart	82.35 (4.84)	82.21 (4.77)	**86.47 (2.85)**	82.50 (5.54)	83.09 (3.32)	83.09 (3.32)
Ionosphere	86.14 (4.90)	86.19 (4.70)	87.27 (3.21)	91.53 (1.45)	**92.44 (2.56)**	**92.44 (2.56)**
Laryngeal1	81.23 (2.70)	80.09 (3.80)	**80.94 (4.70)**	79.25 (5.05)	80.57 (5.87)	80.57 (5.87)
Laryngeal3	71.57 (5.25)	68.88 (6.19)	72.58 (2.14)	**73.48 (2.48)**	67.42 (1.86)	**73.48 (2.48)**
Liver	64.59 (4.87)	66.74 (2.70)	58.37 (2.81)	62.21 (5.26)	**66.98 (1.79)**	**66.98 (1.79)**
Mammographic	82.00 (3.18)	78.89 (4.09)	81.63 (3.06)	80.05 (1.84)	**82.57 (1.77)**	82.57 (1.77)
Monk2	86.06 (3.06)	85.88 (3.16)	90.28 (2.18)	**94.91 (0.97)**	94.07 (0.76)	94.07 (0.76)
Phoneme	80.78 (0.65)	77.26 (0.89)	87.01 (0.77)	**89.18 (0.50)**	86.62 (0.69)	86.62 (0.69)
Pima	74.66 (2.39)	72.19 (3.63)	73.23 (3.39)	73.46 (1.01)	**76.74 (2.24)**	**76.74 (2.24)**
Sonar	76.35 (5.41)	75.87 (4.93)	78.08 (5.01)	**82.98 (5.27)**	76.35 (3.64)	76.35 (3.64)
Vehicle	72.03 (1.63)	72.41 (1.86)	70.75 (2.22)	**73.51 (1.64)**	71.34 (1.28)	**73.51 (1.64)**
Vertebral	84.55 (3.42)	85.51 (3.30)	82.31 (1.93)	85.32 (2.68)	**86.47 (2.65)**	**86.47 (2.65)**
Weaning	73.88 (2.78)	73.75 (3.51)	78.82 (3.05)	**86.51 (1.90)**	85.66 (2.37)	85.66 (2.37)
Average	76.67	75.64	78.22	**79.70**	78.99	80.08
Wilcoxon (Bagging)	n/a	-	∼	+	+	+
Wilcoxon (FIRE)	+	n/a	+	+	+	+
Wilcoxon (GP)	∼	-	n/a	+	∼	+
(c)						
Adult	85.00 (2.53)	83.70 (3.25)	**88.15 (2.93)**	84.45 (3.98)	87.75 (2.18)	87.75 (2.18)
Blood	75.16 (2.07)	68.88 (3.24)	75.53 (1.14)	**76.99 (2.15)**	76.57 (1.90)	76.57 (1.90)
CTG	88.87 (1.24)	88.61 (1.54)	90.24 (0.77)	**92.10 (1.20)**	90.23 (0.36)	**92.10 (1.20)**
Faults	66.58 (1.37)	65.77 (2.32)	71.91 (1.60)	**72.80 (1.29)**	66.24 (0.86)	**72.80 (1.29)**
German	70.16 (2.41)	68.94 (2.72)	70.52 (2.08)	72.84 (2.36)	**74.08 (1.84)**	**74.08 (1.84)**
Glass	60.00 (5.83)	60.19 (6.45)	**66.79 (4.17)**	66.42 (4.27)	61.04 (2.89)	66.42 (4.27)
Haberman	72.17 (5.87)	68.36 (4.95)	71.71 (4.91)	69.80 (2.85)	**72.37 (2.67)**	72.37 (2.67)
Heart	81.10 (4.11)	81.32 (4.82)	**86.18 (2.36)**	82.06 (4.86)	83.09 (3.32)	83.09 (3.32)
Ionosphere	88.24 (2.56)	86.36 (2.55)	87.16 (2.71)	91.48 (1.40)	**92.16 (2.39)**	**92.16 (2.39)**
Laryngeal1	**83.02 (4.29)**	81.51 (6.44)	80.57 (4.59)	78.30 (5.03)	80.47 (5.75)	80.47 (5.75)
Laryngeal3	70.96 (5.62)	67.19 (4.71)	71.80 (1.58)	**72.19 (2.65)**	66.18 (1.41)	**72.19 (2.65)**
Liver	61.98 (4.86)	63.49 (5.01)	58.37 (3.49)	61.34 (4.71)	**67.03 (1.32)**	67.03 (1.32)
Mammographic	82.31 (2.32)	79.01 (3.40)	**82.60 (2.47)**	78.87 (2.55)	82.52 (1.65)	82.52 (1.65)
Monk2	88.06 (4.18)	87.69 (4.29)	87.96 (3.80)	**94.91 (0.97)**	94.07 (0.76)	94.07 (0.76)
Phoneme	80.53 (0.79)	76.12 (1.05)	86.73 (0.73)	**88.98 (0.56)**	86.68 (0.74)	86.68 (0.74)
Pima	72.73 (2.60)	68.36 (2.96)	72.71 (2.67)	72.92 (1.56)	**76.74 (2.28)**	**76.74 (2.28)**
Sonar	80.67 (4.11)	80.29 (4.13)	79.81 (3.09)	**83.08 (5.42)**	76.15 (3.33)	76.15 (3.33)
Vehicle	73.30 (2.54)	74.58 (2.45)	70.14 (2.52)	**74.88 (1.57)**	70.75 (1.65)	**74.88 (1.57)**
Vertebral	84.55 (4.75)	85.38 (4.86)	82.69 (2.22)	85.58 (2.38)	**86.41 (2.71)**	**86.41 (2.71)**
Weaning	76.38 (2.43)	76.05 (3.10)	79.21 (3.30)	**86.38 (1.72)**	85.59 (2.36)	85.59 (2.36)
Average	77.08	75.59	78.03	**79.31**	78.80	80.00
Wilcoxon (Bagging)	n/a	-	∼	+	∼	+
Wilcoxon (FIRE)	+	n/a	∼	+	+	+
Wilcoxon (GP)	∼	∼	n/a	∼	∼	+

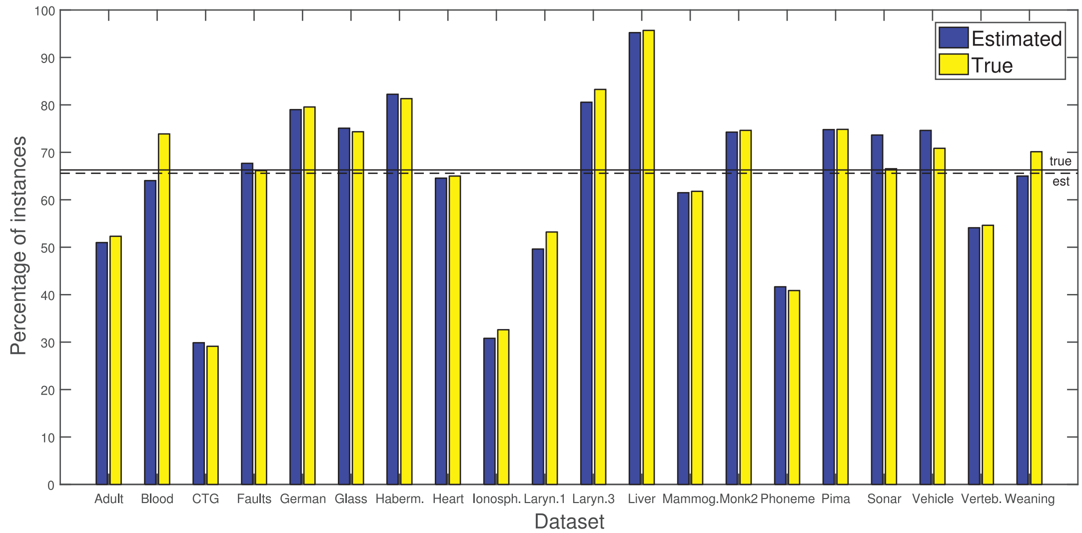


Fig. 7. Mean percentage of instances in difficult regions for all datasets from Table 1. The *Estimated* bar indicates the times the local pool was used to classify an instance, while the *True* bar indicates the percentage of instances with true kDN value above zero. The lines *true* and *est* indicate the averaged values of all datasets for the estimated and true percentage of hard instances, respectively.

which belongs to Class 1, though the classifier from Fig. 8b seems better adjusted than the one from Fig. 8a.

On the other hand, Fig. 8c 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 Fig. 8c 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 $\mathbf{x_q}$.

Fig. 8d depicts the same scenario from Fig. 8c, 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*, which evaluates over the original neighborhood and is likely to give less weight to less relevant classes in the border region. Hence, the LP^{mc} column in Table 4 shows the result of the combined LP^{e} and *LP* configurations, in which the k-NNE is used for 2-class problems and the k-NN for the multi-class problems. It can be observed from the Wilcoxon rows that this scheme is significantly better than Bagging, FIRE-DES and the *GP* configurations for all DCS techniques.

4.2. Comparison with state-of-the-art models

A comparative study on the performances of the proposed method and nine state-of-the-art models is presented in this section. The purpose of this study is to assess whether the proposed method achieves similar recognition rates to the most well-performing models in the literature, considering single models and other MCS.

Five static state-of-the-art classifiers feature in the comparative study: the Multi-layer Perceptron (MLP) model with the Levenberg-Marquadt algorithm, the Support Vector Machine (SVM) model with a Gaussian Kernel, the Random Forest (RF) [34] classifier, the AdaBoost [35] classifier, and the Oblique Decision Tree (DT) ensemble [36]. These models belong to the best performing families of classifiers, according to [37], and also were among the best performing models in [36]. Since static models do not need a DSEL dataset, it was used as a validation set for the MLP classifier and added to the training set for the remaining static models.

Furthermore, four state-of-the-art DS techniques were also included in this analysis: the Randomized Reference Classifier (RRC) [38], the META-DES [5], the META-DES.Oracle (META-DES.O) [39] and the FIRE-KNORA-U (F-KNU) [32]. The latter consists of the FIRE-DES framework coupled with the K-Nearest Oracles Union (KNORA-U) [40] selection technique. The same Bagging-generated pool of 100 Perceptrons used in the previous section was used for these techniques. The region of competence size was also set to 7, as in the previous experiments.

All classifiers were evaluated using the MATLAB PRTOOLS toolbox [41], and the parameters of the static models were set to the default. Moreover, the proposed method's configuration used for comparison in this analysis was the LP^{mc} with LCA, since it yielded the highest mean accuracy rate in the previous experimental study.

Table 5 shows the mean accuracy rate of the static classification models and the proposed method, for all datasets from Table 1. It can be observed that the proposed configuration yielded a higher overall accuracy rate than all static models but the Oblique DS ensemble. It is important to remember, though, that no fine tuning of parameters was performed, and that stands for all static

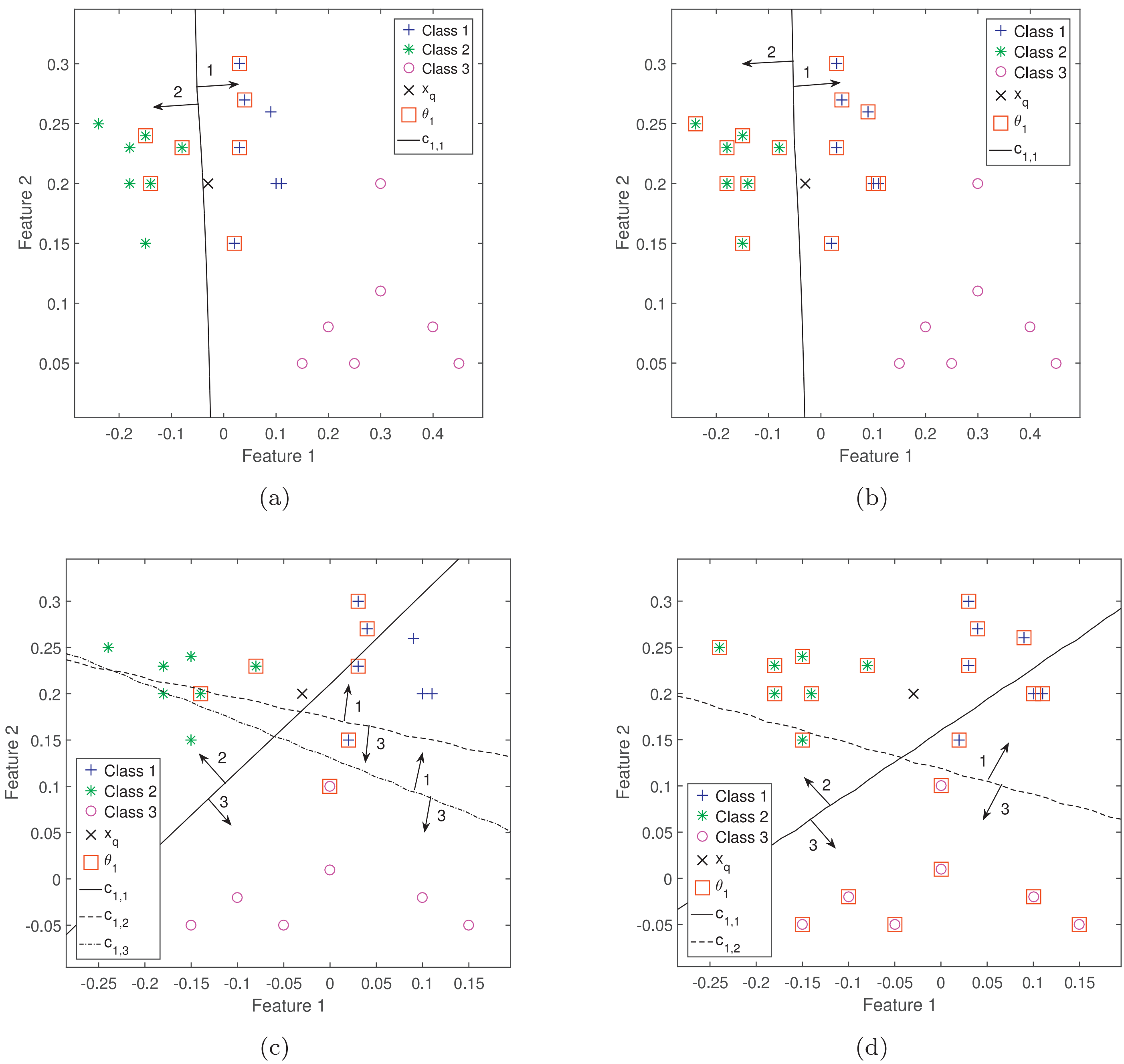


Fig. 8. Example of pool generation for multi-class problems. In all scenarios, x_q belongs to Class 1. In (a) and (c), the query instance's ($\mathbf{x_q}$) neighborhood θ_1 was obtained using k-NN with $k_1 = 7$. In (b) and (d), θ_1 was obtained using a version of k-NNE with $k_1 = 7$ as well. These neighborhoods were used as input to the SGH method, which yielded the corresponding subpool of classifiers depicted in the images.

models as well as the proposed technique. A Wilcoxon signed-rank test was also performed over the results (row Wilcoxon), and it can be observed that the performance of the proposed configuration was significantly superior to the MLP and SVM models.

Table 6 shows the mean accuracy rate of the four state-of-the-art DES techniques and the proposed configuration. It can be observed that the proposed technique obtained a greater mean accuracy rate in comparison with three of the four DES techniques. Moreover, according to a Wilcoxon signed-rank test with significance level of $\alpha = 0.05$ (Wilcoxon row), the proposed configuration obtained a significantly superior performance to the F-KNU technique. In comparison to the remaining DES techniques, the proposed method yielded a statistically similar performance.

4.3. Computational complexity

An analysis of the complexity of the proposed method versus the complexity of different DS techniques is performed next using the Big-O ($\mathcal{O}$) and Big-Omega (Ω) notations [42] to represent the worst and best running time scenarios, respectively. This analysis is made taking into account the dataset size n, the classifiers pool size m, the dimensionality of the dataset d and the neighborhood size k. For simplicity, we consider that all base classifiers are from the same model (Perceptron), and the cost of training the base classifier is denoted by l.

During the offline phase of the proposed method, the hardness value of all instances are estimated, so the final cost of the

Table 5
Mean and standard deviation of the accuracy rate of MLP, SVM, RF, AdaBoost, Oblique DT ensemble and the LP^{mc} configuration. The row *Wilcoxon* shows the result of a Wilcoxon signed rank test over the mean accuracy rates of the proposed configuration and each of the remaining methods. 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. Best results are in bold.

Dataset	MLP	SVM	RF [34]	AdaBoost [35]	Oblique DT ens. [36]	LP^{mc}
Adult	82.83 (3.61)	73.99 (2.88)	67.83 (8.34)	88.44 (2.05)	**88.76 (1.43)**	87.11 (2.40)
Blood	78.11 (1.63)	**78.14 (1.08)**	72.07 (3.00)	77.29 (1.74)	77.23 (0.95)	76.89 (1.67)
CTG	89.52 (1.46)	84.51 (0.53)	91.20 (0.94)	92.42 (1.69)	**93.96 (0.68)**	92.22 (1.10)
Faults	68.79 (4.45)	49.59 (0.30)	70.86 (2.35)	54.66 (2.43)	**77.04 (2.06)**	73.20 (1.22)
German	70.24 (3.08)	70.12 (0.19)	38.28 (6.78)	74.82 (1.94)	**75.18 (1.88)**	74.08 (1.84)
Glass	61.32 (7.20)	62.83 (3.47)	71.04 (4.59)	48.30 (5.98)	**78.58 (3.14)**	67.45 (2.73)
Haberman	70.72 (2.33)	**74.34 (2.02)**	69.41 (4.24)	68.62 (3.48)	72.04 (3.69)	72.11 (2.12)
Heart	75.15 (5.02)	58.82 (1.79)	62.94 (6.45)	84.26 (4.80)	**86.32 (3.20)**	83.09 (3.32)
Ionosphere	87.84 (4.30)	68.07 (2.24)	93.81 (2.16)	**96.36 (1.50)**	95.17 (1.32)	92.44 (2.56)
Laryngeal1	79.53 (6.16)	76.23 (3.26)	81.79 (4.55)	81.60 (3.29)	**85.94 (2.90)**	80.57 (5.87)
Laryngeal3	67.87 (5.36)	68.88 (2.78)	72.53 (2.93)	70.11 (2.88)	**73.54 (3.25)**	73.48 (2.48)
Liver	68.31 (4.80)	64.53 (3.82)	69.30 (4.52)	69.71 (3.49)	**71.05 (4.11)**	66.98 (1.79)
Mammographic	84.45 (2.94)	83.46 (2.87)	59.28 (8.08)	80.70 (2.25)	**84.57 (1.45)**	82.57 (1.77)
Monk2	99.49 (1.06)	95.28 (1.37)	89.54 (2.93)	**100.0 (0.00)**	96.90 (1.28)	94.07 (0.76)
Phoneme	83.51 (1.07)	87.43 (0.43)	90.34 (0.49)	**90.66 (0.55)**	89.65 (0.52)	86.62 (0.69)
Pima	74.35 (3.63)	71.46 (2.20)	76.25 (2.67)	75.42 (1.94)	**77.21 (1.37)**	76.74 (2.24)
Sonar	78.08 (5.49)	81.15 (4.02)	83.75 (5.38)	**85.38 (5.16)**	83.56 (5.56)	76.35 (3.64)
Vehicle	**76.91 (2.38)**	63.63 (2.85)	73.77 (2.21)	68.94 (3.30)	74.27 (2.43)	73.51 (1.64)
Vertebral	81.03 (4.15)	85.00 (2.76)	85.45 (3.63)	84.04 (2.48)	85.96 (3.80)	**86.47 (2.65)**
Weaning	78.42 (5.20)	69.74 (6.80)	86.97 (2.73)	**87.76 (1.87)**	86.38 (2.06)	85.66 (2.37)
Average	77.82	73.36	75.32	78.97	**82.66**	80.08
Wilcoxon	-	-	$\sim$	$\sim$	+	n/a

Table 6
Mean and standard deviation of the accuracy rate of the Randomized Reference Classifier (RRC), the META-DES, the META-DES.Oracle (META-DES.O), the FIRE-KNORA-U (F-KNU) and the LP^{mc} configuration. The row *Wilcoxon* shows the result of a Wilcoxon signed rank test over the mean accuracy rates of the proposed configuration and each of the remaining methods. 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. Best results are in bold.

Dataset	RRC [38]	META-DES [5]	META-DES.O [39]	F-KNU [32]	LP^{mc}
Adult	**88.87 (2.27)**	84.45 (6.41)	80.78 (7.33)	84.86 (2.85)	87.11 (2.40)
Blood	76.30 (1.41)	**77.98 (1.81)**	**77.98 (1.20)**	64.87 (2.56)	76.89 (1.67)
CTG	89.41 (0.71)	91.49 (0.66)	92.10 (1.12)	88.34 (0.94)	**92.22 (1.10)**
Faults	70.35 (1.06)	**73.58 (1.57)**	73.39 (1.61)	67.44 (1.74)	73.20 (1.22)
German	**76.42 (2.14)**	75.70 (1.69)	74.76 (1.82)	70.30 (1.01)	74.08 (1.84)
Glass	65.19 (4.39)	**70.19 (3.44)**	69.53 (5.17)	65.47 (3.73)	67.45 (2.73)
Haberman	74.08 (1.71)	73.82 (5.79)	**75.26 (2.36)**	57.24 (4.91)	72.11 (2.12)
Heart	86.62 (1.42)	**86.47 (3.53)**	81.10 (4.35)	85.51 (2.35)	83.09 (3.32)
Ionosphere	88.75 (2.24)	88.47 (2.19)	85.17 (5.10)	88.35 (1.91)	**92.44 (2.56)**
Laryngeal1	**85.19 (3.08)**	80.28 (4.95)	78.21 (6.14)	80.94 (5.30)	80.57 (5.87)
Laryngeal3	**74.27 (3.40)**	73.54 (3.31)	73.48 (3.63)	66.24 (4.13)	73.48 (2.48)
Liver	65.81 (4.34)	**68.95 (3.25)**	67.21 (3.57)	60.58 (3.99)	66.98 (1.79)
Mammographic	**85.77 (2.08)**	73.13 (15.9)	72.36 (18.2)	78.53 (2.58)	82.57 (1.77)
Monk2	85.23 (2.71)	**96.76 (1.22)**	**96.76 (1.22)**	85.32 (2.57)	94.07 (0.76)
Phoneme	74.08 (1.57)	87.49 (0.82)	**89.34 (0.69)**	73.89 (1.61)	86.62 (0.69)
Pima	76.95 (2.33)	**77.40 (1.94)**	76.98 (2.49)	67.29 (2.73)	76.74 (2.24)
Sonar	80.96 (2.92)	82.98 (3.38)	**83.75 (2.96)**	81.35 (2.93)	76.35 (3.64)
Vehicle	75.40 (1.97)	75.94 (2.21)	75.12 (2.11)	**76.56 (1.84)**	73.51 (1.64)
Vertebral	85.19 (3.01)	86.22 (3.45)	85.77 (2.79)	**87.12 (3.98)**	86.47 (2.65)
Weaning	81.84 (3.27)	84.28 (3.37)	82.89 (3.28)	81.12 (3.29)	**85.66 (2.37)**
Average	79.33	**80.45**	79.59	75.50	80.08
Wilcoxon	$\sim$	$\sim$	$\sim$	-	n/a

proposed method in memorization is equal to applying the k-NN rule over the training set, $\mathcal{O}(n^2d)$. For the DS techniques, however, the training step includes generating a pool of size m, which yields a cost of $\mathcal{O}(ml)$, and pre-processing the base classifiers outputs for each sample in the training set, thus adding a cost of $\mathcal{O}(mn)$.

As for the computational cost in generalization, the analysis is conducted by dividing each algorithm into the DS techniques's three steps [9]: region of competence definition, competence estimation and classification. Then, the computational complexity of each part is obtained individually.

For the region of competence definition, the proposed method and all DS methods studied in this work are based on the k-NN algorithm. Thus, it is of order $\mathcal{O}(nd)$. The two versions of the META-DES framework also require a local competence estimation on the decision space, resulting in a computational complexity of $\mathcal{O}(nd + nm)$ for these techniques.

In the competence level estimation, the cost involved to calculate the competence level of m base classifiers is equal to getting their performance on the query sample's neighbors, $\mathcal{O}(mk)$. The Randomized Reference Classifier (RRC) algorithm, however, uses the whole dataset to calculate the competence level of the base classifiers, making this step of order $\mathcal{O}(nm)$.

In the proposed method, the SGH method is applied in each iteration over a different neighborhood, resulting in m'' classifiers.

These classifiers are then evaluated over a neighborhood of size k, so, for each iteration, $\mathcal{O}(km''d)$. The best classifier produced in each iteration is added to the local pool, so, for a local pool size of m' hyperplanes, the cost involved in this part is $\mathcal{O}(m''km'd)$.

The classification step using the proposed method is performed by applying the majority voting rule over the m' local classifiers, so the computational cost is of order $\mathcal{O}(m'd)$. In contrast, the DCS techniques use only one classifier for classification, which yields a computational complexity of $\mathcal{O}(d)$. The DES techniques, however, can select an arbitrary number of base classifiers, so in the worst case scenario, when all classifiers in the pool are selected, it has a complexity of order $\mathcal{O}(md)$.

This analysis explains why the proposed method was much faster in generalization than the DS techniques using a Bagging-generated pool. In the best case scenario, when the sample is located in an easy region, no local pool is generated, yielding a cost equal to the k-NN classifier's, $\Omega(nd)$. In the worst case, the computational cost is the sum of the three steps: $\mathcal{O}(nd + m''dkm' + m'd)$. However, DCS and DES techniques always require the computation of the three steps regardless of whether or not the query sample is located in an easy region. For this reason, the average running time of the proposed method was three times faster than the DCS techniques using Bagging-generated pool, eight times faster than the KNORA-U and also around 14 times faster than the RRC and META-DES techniques. However, the DCS techniques using the GP were 10 times faster than the proposed method, probably due to its reduced pool size (3.80 classifiers, on average).

5. Conclusion

In [8], it was shown that the DCS techniques had difficulty in selecting a competent classifier even though the presence of such a classifier in the pool was assured. The generation method used in that work guaranteed an Oracle accuracy of 100%. It was concluded that the Oracle model, 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.

In this work, an instance hardness analysis was performed in order to draw a correlation between hardness measures and the error rates of DCS techniques. Based on that relationship, an online pool generation scheme was proposed with the purpose of increasing the accuracy rates of the instances the DCS techniques had difficulty in labelling. The proposed technique involved generating subpools for each identified difficult region in the feature space, so that another, more locally accurate pool could be used for hard instances, in hopes that, by fully covering these regions with a locally specialist pool, it would be easier for the DS techniques to select the best classifiers for these samples. The instances deemed “easy”, however, would be classified using a simple k-NN rule.

Experiments were performed over 20 public datasets, and two configurations of the proposed scheme were analyzed. It was shown that the use of local pools increased the hit rate of the global pool (GP) for most datasets, suggesting the use of such pools indeed helps the DS in selecting the most competent ones for a given query instance. The overall performances of both configurations were compared with a pool of 100 Perceptrons generated using Bagging, with an online pruned pool of originally 100 Bagging-generated Perceptrons and with the GP. It was observed that a combination of both proposed configurations yields a significantly increase in accuracy rate compared to the other tested methods for the three evaluated DCS techniques, suggesting that, not only do the DCS techniques select the best classifier more frequently, but also the recognition rates of the DCS techniques fairly increase when using a local perspective during generation. The choice of which proposed configuration to use is based on the characteristics of each problem, and this selection is necessary due to a limitation in the SGH method. Furthermore, the proposed technique was compared to nine state-of-the-art classification models, including five static and four DES techniques, and it yielded a significant superior performance to three of the models and a statistically similar performance to five of them.

Improvements to the proposed technique may involve developing an automatic scheme for defining the input parameters and an adaptation for better dealing with multi-class problems. Moreover, the impacts of data preprocessing on the performance of DS techniques over imbalanced problems have been analyzed in [43]. Thus, a study on the robustness of the proposed method to class imbalance and the suitability of using data preprocessing techniques for imbalance learning may also be performed in future works.

Acknowledgements

The authors would like to thank Brazilian agencies: CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior), CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) and FACEPE (Fundação de Amparo à Ciência e Tecnologia de Pernambuco).

References

[1] M. Woźniak, M. Graña, E. Corchado, A survey of multiple classifier systems as hybrid systems, Inf. Fusion 16 (2014) 3–17.
[2] J. Kittler, M. Hatef, R.P.W. Duin, J. Matas, On combining classifiers, IEEE Trans. Pattern Anal. Mach. Intell. 20 (1998) 226–239.
[3] L. Kuncheva, Combining pattern classifiers: Methods and algorithms, J. Wiley, 2014.
[4] A. Britto, R. Sabourin, L. Oliveira, Dynamic selection of classifiers - A comprehensive review, Pattern Recognit. 47 (11) (2014) 3665–3680.
[5] R.M.O. Cruz, R. Sabourin, G.D.C. Cavalcanti, T.I. Ren, META-DES: A dynamic ensemble selection framework using meta-learning, Pattern Recognit. 48 (5) (2015) 1925–1935.
[6] A.H.R. Ko, R. Sabourin, A.S.B. Jr., From dynamic classifier selection to dynamic ensemble selection, Pattern Recognit. 41 (5) (2008) 1718–1731.
[7] L.I. Kuncheva, A theoretical study on six classifier fusion strategies, IEEE Trans. Pattern Anal. Mach. Intell. 24 (2) (2002) 281–286.
[8] M.A. Souza, G.D. Cavalcanti, R.M. Cruz, R. Sabourin, On the characterization of the oracle for dynamic classifier selection, in: International Joint Conference on Neural Networks, IEEE, 2017, pp. 332–339.
[9] R.M. Cruz, R. Sabourin, G.D. Cavalcanti, Dynamic classifier selection: recent advances and perspectives, Inf. Fusion 41 (2018) 195–216.
[10] L. Bottou, V. Vapnik, Local learning algorithms, Neural Comput. 4 (6) (1992) 888–900.
[11] G. Armano, N. Hatami, Mixture of random prototype-based local experts, in: International Conference on Hybrid Artificial Intelligence Systems, Springer, 2010, pp. 548–556.
[12] G. Armano, E. Tamponi, Building forests of local trees, Pattern Recognit. 76 (2018) 380–390.
[13] M.R. Smith, T. Martinez, C. Giraud-Carrier, An instance level analysis of data complexity, Mach. Learn. 95 (2) (2014) 225–256.
[14] M. Dong, R. Kothari, Feature subset selection using a new definition of classifiability, Pattern Recognit. Lett. 24 (9) (2003) 1215–1225.
[15] M.R. Smith, T. Martinez, A comparative evaluation of curriculum learning with filtering and boosting in supervised classification problems, Comput. Intell. 32 (2) (2016) 167–195.
[16] S. Singh, Multiresolution estimates of classification complexity, IEEE Trans. Pattern Anal. Mach. Intell. 25 (12) (2003) 1534–1539.
[17] T.K. Ho, M. Basu, Complexity measures of supervised classification problems, IEEE Trans. Pattern Anal. Mach. Intell. 24 (3) (2002) 289–300.
[18] L.P. Garcia, A.C. de Carvalho, A.C. Lorena, Effect of label noise in the complexity of classification problems, Neurocomputing 160 (2015) 108–119.
[19] M.R. Smith, T. Martinez, Improving classification accuracy by identifying and removing instances that should be misclassified, in: International Joint Conference on Neural Networks, IEEE, 2011, pp. 2690–2697.
[20] C.E.C. de Melo, R.B.C. Prudêncio, Cost-sensitive measures of algorithm similarity for meta-learning, in: Brazilian Conference on Intelligent Systems, IEEE, 2014, pp. 7–12.
[21] R.M. Cruz, H.H. Zakane, R. Sabourin, G.D. Cavalcanti, Dynamic ensemble selection vs k-nn: why and when dynamic selection obtains higher classification performance? in: 2017 Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA), IEEE, 2017.
[22] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140.
[23] K. Bache, M. Lichman, UCI machine learning repository, 2013, (Available: http://archive.ics.uci.edu/ml). [Online].

[24] L. Kuncheva, Ludmila Kuncheva Collection, 2004, (Available: http://pages.bangor.ac.uk/~mas00a/activities/real_data.htm). [Online].
[25] R.D. King, C. Feng, A. Sutherland, Statlog: comparison of classification algorithms on large real-world problems, Appl. Artif. Intell. 9 (3) (1995) 289–333.
[26] J. Alcalá, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, KEEL Data-mining software tool: data set repository, integration of algorithms and experimental analysis framework, J. Multiple-Valued Logic Soft Comput. 17 (2–3) (2011) 255–287.
[27] C. Jutten, The enhanced learning for evolutive neural architectures project, 2002, (Available: https://www.elen.ucl.ac.be/neural-nets/Research/Projects/ELENA/elena.htm). [Online].
[28] K. Woods, W.P. Kegelmeyer Jr, K. Bowyer, Combination of multiple classifiers using local accuracy estimates, IEEE Trans. Pattern Anal. Mach. Intell. 19 (4) (1997) 405–410.
[29] G. Valentini, An experimental bias-variance analysis of svm ensembles based on resampling techniques, IEEE Trans. Syst. Man. Cybern. Part B 35 (2005) 1252–1271.
[30] R.M. Cruz, R. Sabourin, G.D. Cavalcanti, META-DES.H: a dynamic ensemble selection technique using meta-learning and a dynamic weighting approach, in: 2015 International Joint Conference on Neural Networks, IEEE, 2015, pp. 1–8.
[31] B. Sierra, E. Lazkano, I. Irigoien, E. Jauregi, I. Mendialdua, K nearest neighbor equality: giving equal chance to all existing classes, Inf. Sci. (Ny) 181 (23) (2011) 5158–5168.
[32] D.V. Oliveira, G.D. Cavalcanti, R. Sabourin, Online pruning of base classifiers for dynamic ensemble selection, Pattern Recognit. 72 (2017) 44–58.
[33] A. Benavoli, G. Corani, F. Mangili, Should we really use post-hoc tests based on mean-ranks? J. Mach. Learn. Res. 17 (5) (2016) 1–10.
[34] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[35] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, J. Comput. Syst. Sci. 55 (1997) 119–139.
[36] L. Zhang, P.N. Suganthan, Benchmarking ensemble classifiers with novel co-trained kernal ridge regression and random vector functional link ensembles [research frontier], IEEE Comput. Intell. Mag. 12 (4) (2017) 61–72.
[37] M. Fernández-Delgado, E. Cernadas, S. Barro, D. Amorim, Do we need hundreds of classifiers to solve real world classification problems, J. Mach. Learn. Res 15 (1) (2014) 3133–3181.
[38] T. Woloszynski, M. Kurzynski, A probabilistic model of classifier competence for dynamic ensemble selection, Pattern Recognit. 44 (10) (2011) 2656–2668.
[39] R.M. Cruz, R. Sabourin, G.D. Cavalcanti, Meta-des. oracle: meta-learning and feature selection for dynamic ensemble selection, Inf. Fusion 38 (2017) 84–103.
[40] A.H.-R. Ko, R. Sabourin, A. de Souza Britto Jr, A new dynamic ensemble selection method for numeral recognition, in: 7th International Conference on Multiple Classifier Systems, Springer-Verlag, 2007, pp. 431–439.
[41] R.P.W. Duin, P. Juszczak, D. de Ridder, P. Paclik, E. Pekalska, D.M. Tax, Prtools, a matlab toolbox for pattern recognition, 2004, (http://www.prtools.org).
[42] D.E. Knuth, Big omicron and big omega and big theta, ACM Sigact. News 8 (2) (1976) 18–24.
[43] A. Roy, R.M. Cruz, R. Sabourin, G.D. Cavalcanti, A study on combining dynamic selection and data preprocessing for imbalance learning, Neurocomputing 286 (2018) 179–192.

Mariana A. Souza is a Ph.D. student at the Center for Informatics, Federal University of Pernambuco, Brazil. Her main research interests are ensemble of classifiers, dynamic ensemble selection and prototype selection.

George D. C. Cavalcanti is an Associate Professor with the Center for Informatics, Federal University of Pernambuco, Brazil. Dr. Cavalcanti received the D.Sc. degree in Computer Science from the Center for Informatics, Federal University of Pernambuco, Brazil in 2005. He is the author/co-author of more than 150 scientific publications including journals and conference proceeding. His research interests include machine learning, pattern recognition, and image analysis.

Rafael M. O. Cruz obtained a Ph.D. in Engineering from the École de Technologie Supérieure (ÉTS), Université du Québec in 2016. Currently he is a post-doctoral researcher at LIVIA (Laboratoire d'imagerie, de vision et d'intelligence artificielle). His main research interests are ensemble of classifiers, dynamic ensemble selection, meta-learning, prototype selection and handwritten recognition.

Robert Sabourin joined the physics department of the Montreal University in 1977 where his main contribution was the design and the implementation of a microprocessor-based fine tracking system combined with a low-light level CCD detector. In 1983, he joined the staff of the École de Technologie Supérieure, Université du Québec, in Montreal, where he co-founded the Dept. of Automated Manufacturing Engineering where he is currently Full Professor and teaches Pattern Recognition, Evolutionary Algorithms, Neural Networks and Fuzzy Systems. In 1992, he joined also the Computer Science Department of the Pontifćia Universidade Católica do Paraná (Curitiba, Brazil). Since 1996, he is a senior member of the Centre for Pattern Recognition and Machine Intelligence (CENPARMI, Concordia University). Since 2012, he is the Research Chair holder specializing in Adaptive Surveillance Systems in Dynamic Environments. Dr. Sabourin is the author (and co-author) of more than 400 scientific publications including journals and conference proceeding. He was co-chair of the program committee of CIFED98, in Quebec, Canada and IWFHR04 in Tokyo, Japan. He was nominated as Conference co-chair of ICDAR07 that has been held in Curitiba, Brazil in 2007. His research interests are in the areas of adaptive biometric systems, adaptive surveillance systems in dynamic environments, intelligent watermarking systems, evolutionary computation and bio-cryptography.