On the Characterization of the Oracle for Dynamic Classifier Selection

Mariana A. Souza and George D. C. Cavalcanti
Centro de Informática
Universidade Federal de Pernambuco
Email: mas2@cin.ufpe.br, gdcc@cin.ufpe.br

Rafael M. O. Cruz and Robert Sabourin
École de Technologie Supérieure
Université du Québec
Email: cruz@livia.etsmtl.ca, robert.sabourin@etsmtl.ca

Abstract*—The Oracle model has been used not only for comparison between techniques but also in the design of different methods in Multiple Classifier Systems (MCS). Even though the model represents the ideal classifier selection scheme, Dynamic Classifier Selection (DCS) techniques present a large performance gap from the Oracle. This means that, for a significant number of instances, the DCS techniques are not able to select a competent classifier, despite the Oracles assurance of its presence in the pool. Given that issue, this work aims to investigate the reasons why the Oracle model may not be well suited for guiding the search for a promising pool of classifiers for DCS techniques. For this purpose, a pool generation method that guarantees an Oracle accuracy rate of 100% in the training set is proposed. This method is further used to analyse the behavior of DCS techniques when the presence of at least one competent classifier in the pool for each training sample is assured. Experiments show that integrating Oracle information in the generation phase of an MCS has little impact on the gap between the accuracy rates of DCS techniques and the Oracle. Moreover, it is also shown that, for a theoretical limit of 100%, the DCS techniques were only able to select a competent classifier for at most 85% of the instances, on average. Results suggest that the Oracle is not the best guide for generating a pool of classifiers for DCS, for the model is performed globally whilst DCS techniques work with local data only.

I. Introduction

The Oracle, defined in [1], is an abstract model in which, for each test instance, the classifier that correctly labels it is selected, if the pool contains such a classifier. 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 [1], [2].

Moreover, the concept of the Oracle is used in different areas of MCS. For instance, it is quite related to the construction of diverse pools. 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. Intuitively, a highly diverse pool of classifiers would have a high Oracle accuracy rate [3], [4].

Another area in which the Oracle's properties are explored is Dynamic Selection (DS) techniques [5], which select specific ensembles 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 [6], 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 [7] 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.

For DCS techniques, in which only one classifier is selected to label a query instance, the Oracle 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 [2], [6]. For instance, in [8], 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 [9]. 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 are investigated in this work. To that end, a supervised method for producing a pool of classifiers that guarantees an Oracle accuracy rate of 100% in the training set is proposed. That is, the proposed method 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, is then analyzed, and based on that analysis the relationship between the Oracle model and DCS techniques is discussed.

Apart from guaranteeing an Oracle of 100% in the training set, the proposed technique for generating a pool of classifiers presents other advantages in comparison with classical ensemble methods such as Bagging, Random Subspace [10], and Boosting [11]. 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, it uses a heuristic to train the classifiers, which makes the training much faster than in these methods.

This article is organized as follows. In Section II, the proposed ensemble generation method used for the subsequent analysis of the relationship between the Oracle and DCS techniques is presented. Experiments are conducted in Section III, and the results are evaluated and interpreted. Finally, in the last section, experimental results are summarized, and future researches are suggested.

II. The Proposed Method

The proposed method is presented in Algorithm 1. It 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 [12], and hence a better comparison between them. Therefore, the algorithm itself discovers the amount of Perceptrons needed in the pool, according to the training dataset. Furthermore, to speed up the training process, we also proposed a heuristic to find the Perceptrons' weights without explicitly training them.

Algorithm 1 Pseudocode of the proposed method.

```
1: $\Gamma \leftarrow \{z_1, z_2, ..., z_N\}$ {Training dataset}
2: $C \leftarrow \{c_1, c_2, ..., c_{|C|}\}$ {Problem classes}
3: $Pool \leftarrow \{\}$
4: while $\Gamma \neq \{\}$ do
5:     for $j \leftarrow 1, |C|$ do
6:         $R(j) \leftarrow centroid(c_j)$ {Centroid of class $j$}
7:     end for
8:     $d \leftarrow max(pairwiseDistance(R))$ {Maximum distance between centroids}
9:     $a, b \leftarrow findIndex(d)$
10:    $midPoint \leftarrow (R(a) + R(b))/2$
11:    $normal \leftarrow (R(a) - R(b))/d$
12:    $w_p \leftarrow \{normal\}$ {Perceptron p weights}
13:    $\theta_p \leftarrow -midPoint \cdot normal$ {Perceptron p bias}
14:    $p \leftarrow perceptron(w_p, \theta_p)$
15:    for $i \leftarrow 1, N$ do
16:        if $test(p, z_i) = label(z_i)$ then {Perceptron p classifies instance $i$ correctly}
17:            $\Gamma \leftarrow \Gamma - \{z_i\}$ {Excludes instance $i$ from dataset}
18:        end if
19:    end for
20:    $Pool \leftarrow Pool \cup \{p\}$ {Add Perceptron p to the Pool}
21: end while
22: return $Pool$
```

Figure 1a shows a training dataset with N = 350 instances and $|C|$ = 5 classes. The step-by-step execution of the algorithm for this example happens as follows. Steps 1 and 2 of Algorithm 1 consist of assigning to Γ the entire training dataset and to C the five classes of the problem ({1,2,3,4,5}). Step 4 assigns the Pool to an empty set.

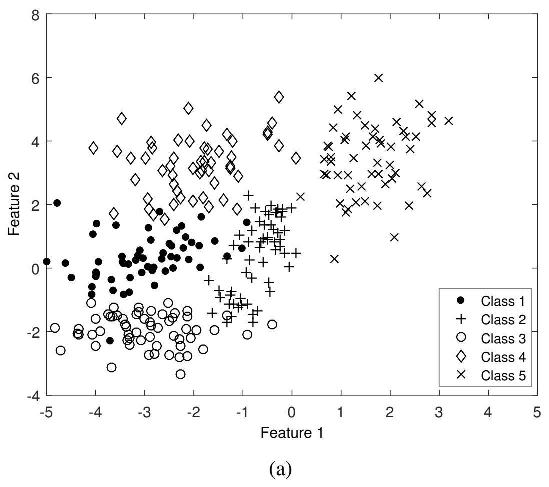
In the first iteration of the algorithm's outer loop, the centroids of each of the five classes are calculated and stored in the matrix R (steps 6 to 8). All five centroids are represented by asterisks (*) in Figure 2a. Then, in steps 9 and 10, the two most distant points in R are chosen, in this case *R(3)* and *R(5)*, and their classes *a,b = 3,5* identified. In Figure 2a, Class 3 and Class 5 centroids are the large asterisks. From step 11 to step 15, the weights w_p and bias θ_p of Perceptron 1 are calculated, using Equations 1 and 2 with *a,b = 3,5*. The weights w_p of the Perceptron are the coordinates of the normalized distance vector between the centroids, while the Perceptron bias θ_p is obtained by applying the scalar product between the midpoint and the normalized distance vector between the centroids. These two parameters are calculated so that Perceptron 1 separates *R(3)* and *R(5)* halfway between them, as can be observed in Figure 2a. In steps 16 to 20, each instance in Γ is tested with Perceptron 1, and the instances correctly classified are then excluded from Γ. Since Perceptron 1 correctly classifies all instances of Class 3 and Class 5, as can be seen in Figure 2a, by the end of that iteration Γ no longer contains instances of both classes, though it still contains all instances of the other classes. In step 21 the Perceptron 1 is then added to the Pool.

$$\mathbf{w_p} = \frac{\mathbf{R}(a) - \mathbf{R}(b)}{\|\mathbf{R}(a) - \mathbf{R}(b)\|} \quad (1)$$

$$\theta_p = -\mathbf{w_p} \cdot \frac{\mathbf{R}(a) + \mathbf{R}(b)}{2} \quad (2)$$

In the second iteration of the outer loop, Γ contains all instances of Class 1, Class 2, and Class 4, so the centroids of these classes are calculated from steps 6 to 8 and stored in R. These centroids are represented by the three asterisks in Figure 2b. The centroids chosen in steps 9 and 10 are *R(2)* and *R(4)*, since they are the most distant to each other, as can be noticed in Figure 2b, in which they are the large asterisks. Then, the weights w_p and bias θ_p of Perceptron 2 are calculated from step 11 to step 15, dividing the space between *R(2)* and *R(4)* right in the middle, as Figure 2b shows. Perceptron 2 is then used to test each instance in Γ from steps 16 to 20, and the instances that remain in Γ are the ones Perceptron 2 classifies incorrectly. Since Class 2 and Class 4 are not linearly separable, Perceptron 2 is not able to eliminate all instances of those classes. Perceptron 2 is added to the Pool in step 21.

In the third iteration, Γ still has instances of Class 1, Class 2, and Class 4, so their centroids *R(1)*, *R(2)*, and *R(4)* are calculated from steps 6 to 8. 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 centroid of Class 1, as neither Perceptron 1 nor Perceptron



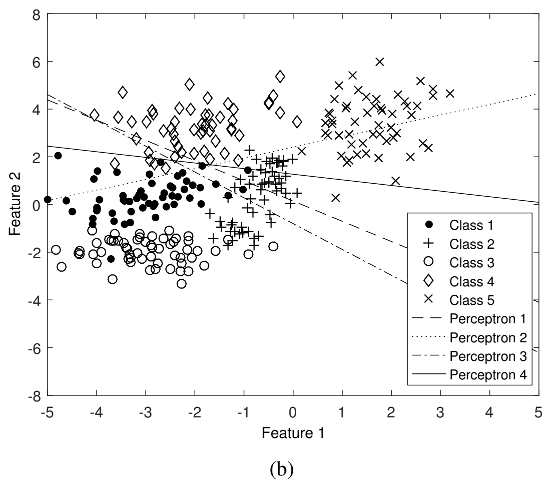


Fig. 1: Toy problem. (a) Training dataset. (b) Perceptrons generated by the proposed method.

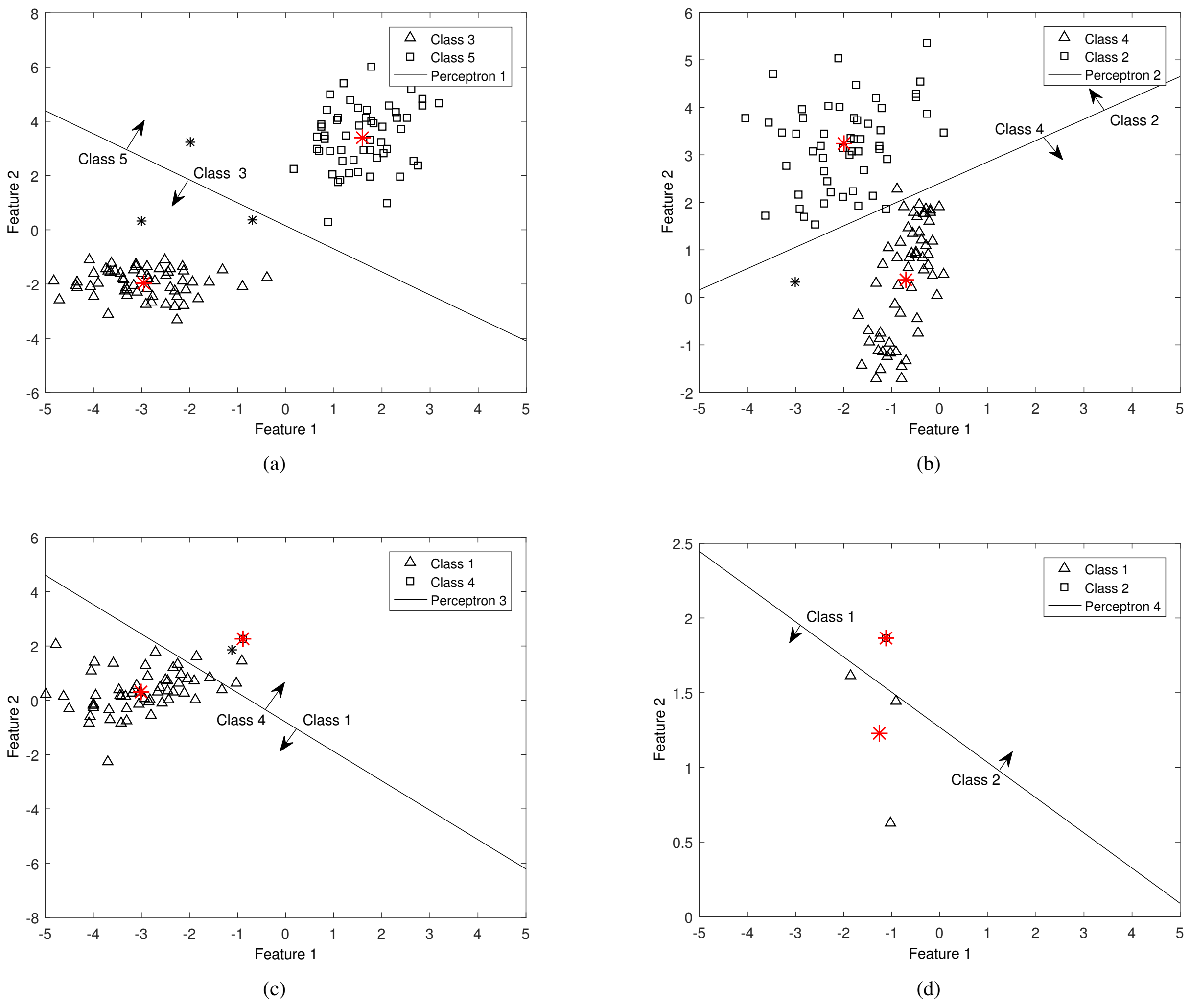


Fig. 2: Generation of Perceptrons using toy problem. (a) First iteration. (b) Second iteration. (c) Third iteration. (d) Last iteration.

2 were able to classify Class 1 instances. The large asterisks in Figure 2c show that centroids *R(1)* and *R(4)* are the most distant ones in this iteration, with centroid *R(2)* in black. Perceptron 3 is then created from step 11 to step 15 so that it splits the plane in a half. From steps 16 to 20, each instance remaining in Γ is then tested with Perceptron 3, and the instances it correctly classifies are further eliminated from Γ. It can be observed that the remaining Class 4 instance is correctly classified by Perceptron 3, so Γ only possesses Class 1 and Class 2 instances after the third iteration. In step 21 the Perceptron 3 is then added to the Pool.

In the fourth and last iteration, Γ contains only 4 instances, 3 of Class 1 and 1 of Class 2, as showed in Figure 2d. Centroids *R(1)* and *R(2)* are calculated from steps 6 to 8 and chosen to calculate the weights w_p and bias θ_p of Perceptron 4 from steps 11 to 15. Each instance in Γ is tested with Perceptron 4 in steps 16 to 20, and since it correctly classifies all 4 remaining instances, they are eliminated and Γ turns into an empty set. Perceptron 4 is then added to the Pool in step 21, and the algorithm leaves the outer loop, returning the Pool in step 23.

Figure 1b shows the entire training dataset with all four Perceptrons generated by the proposed method. The spatial disposition of the only four Perceptrons necessary to "cover" the entire dataset can be observed. It is 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. Furthermore, the algorithm guarantees an Oracle accuracy rate of 100% for the training dataset.

III. Experiments

In order to assess the presence of a significant difference between the Oracle accuracy rate and the accuracy rate yielded by different DCS techniques, as well as its relation to Oracle-guided ensemble methods, experiments were conducted over a total of 20 datasets. All of them are public datasets. Eleven from the UCI machine learning repository [13], three from the Ludmila Kuncheva Collection [14] of real medical data, three from the STATLOG project [15], two from the Knowledge Extraction based on Evolutionary Learning (KEEL) repository [16] and one from the Enhanced Learning for Evolutive Neural Architectures (ELENA) project [17]. The main characteristics of each dataset are shown in Table I. For these experiments, 20 replications of each dataset were used. Each replication was generated by randomly splitting the dataset in two parts: 75% for training and 25% for test.

The accuracy rates of different DCS techniques were obtained using four state-of-the-art methods: Overall Local Accuracy (OLA) [18], Local Class Accuracy (LCA) [18], Modified Local Accuracy (MLA) [19], and Multiple Classifier Behaviour (MCB) [20]. These techniques were chosen due to their superior performance in comparison with other DCS techniques in a recent survey regarding dynamic selection of classifiers [5]. The neighborhood size K for each of the DCS techniques is set to 7, for they also performed the best with this configuration in the survey.

TABLE I: 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

The gap between the Oracle and the DCS techniques is initially evaluated using Bagging, for it is a classical ensemble method used in several DS works [8], [21]. Furthermore, the gap is evaluated using the proposed generation method presented in Section II since it guarantees an Oracle accuracy rate of 100% for the training data, which allows further analysis to be performed.

Table II shows the mean accuracy rates for the pools generated using Bagging for the test data. The generated pool for all datasets, in this case, was always 100 Perceptrons. The use of Perceptrons as base classifiers was done so that it could match the type of base classifier used in the proposed method and in previous works regarding DS techniques. The chosen pool size is fixed for simplicity and contains the same amount of classifiers as in previous works [8], [21], [22].

It can be observed in Table II that, even though the mean Oracle accuracy rate for the test data was over 90%, all DCS techniques tested in this experiment yielded an average accuracy rate below 80%. For almost half of the datasets, the difference between the Oracle and each DCS technique mean generalisation accuracy rates were over 20 percentile points. Thus, as presented in the column Oracle-DCS accuracy gap in this table, there is a significant difference between the theoretical limit and the actual recognition rate of DCS schemes for pools generated by ensemble methods that do not take into account the Oracle model.

To evaluate if the same behavior occurs with Oracle-guided generation methods, the same datasets were tested with pools generated by the proposed method and the accuracy rates of both Oracle and DCS techniques for the test data can be observed in Table III. Since the number of classifiers in the pool is determined by the method itself, the column N_p shows the mean pool size for each dataset. It can be observed that the same behavior present in the previous experiment with Bagging occurs in this case. The mean Oracle accuracy rate for the test data was also over 90%, and although it is quite close to 100%, the mean DCS accuracy rates still did not reach

TABLE II: Mean and standard deviation of the generalisation accuracy rate of the Oracle model and the four chosen DCS techniques for the test dataset, for a pool of 100 Perceptrons generated using Bagging. The Oracle-DCS accuracy gap column denotes the difference between the mean accuracy rates of the Oracle and each DCS technique. Best results are in bold.

Datasets	Oracle	DCS accuracy rate				Oracle-DCS accuracy gap			
		OLA	LCA	MCB	MLA	OLA	LCA	MCB	MLA
Adult	95.59 (0.39)	82.08 (2.42)	**83.58 (2.32)**	78.61 (3.32)	80.34 (1.32)	13.51	12.01	16.98	15.25
Blood	94.20 (2.08)	75.00 (2.36)	75.00 (2.87)	73.40 (4.19)	**76.06 (2.68)**	19.20	19.20	20.80	18.14
CTG	93.08 (1.46)	**86.65 (2.35)**	**86.65 (2.35)**	85.71 (2.21)	86.27 (1.78)	6.43	6.43	7.37	6.81
Faults	88.72 (1.89)	66.52 (1.65)	66.00 (1.69)	**68.17 (1.59)**	67.76 (1.54)	22.20	22.72	20.55	20.96
German	99.12 (0.70)	71.20 (2.52)	73.33 (2.85)	**73.60 (3.30)**	71.20 (2.52)	27.92	25.79	25.52	27.92
Glass	90.65 (0.00)	57.60 (3.65)	59.45 (2.65)	**67.92 (3.24)**	57.60 (3.65)	33.05	31.20	22.73	33.05
Haberman	97.36 (3.34)	69.73 (4.17)	70.16 (3.56)	67.10 (7.65)	**73.68 (3.61)**	27.63	27.20	30.26	23.68
Heart	95.90 (1.02)	85.29 (3.69)	85.29 (3.69)	83.82 (4.05)	**86.76 (5.50)**	10.61	10.61	12.08	9.14
Ionosphere	96.20 (1.72)	**88.63 (1.98)**	88.00 (1.98)	87.50 (2.15)	81.81 (2.52)	7.57	8.20	8.70	14.39
Laryngeal1	98.86 (0.98)	**77.35 (4.45)**	**77.35 (4.45)**	**77.35 (4.45)**	75.47 (5.55)	21.51	21.51	21.51	23.39
Laryngeal3	100.0 (0.00)	71.91 (1.01)	**72.90 (2.30)**	71.91 (1.01)	61.79 (7.80)	28.09	27.10	28.09	38.21
Liver	93.07 (2.41)	**58.13 (3.27)**	58.13 (4.01)	58.00 (4.25)	58.00 (4.25)	34.94	34.94	35.07	35.07
Mammographic	99.59 (0.15)	**82.21 (2.27)**	**82.21 (2.27)**	81.25 (2.07)	75.55 (5.50)	17.38	17.38	18.34	24.04
Monk2	98.98 (1.19)	74.07 (6.60)	74.07 (6.60)	74.07 (6.60)	**75.92 (5.65)**	24.91	24.91	24.91	23.06
Phoneme	99.34 (0.24)	**78.84 (2.53)**	**78.84 (2.53)**	73.37 (5.55)	64.94 (7.75)	20.50	20.50	25.97	34.40
Pima	95.10 (1.19)	73.95 (2.56)	73.95 (2.98)	76.56 (3.71)	**77.08 (4.56)**	21.15	21.15	18.54	18.02
Sonar	94.46 (1.63)	74.52 (1.54)	76.51 (2.06)	76.56 (2.58)	**76.91 (3.20)**	19.94	17.95	17.90	17.55
Vehicle	96.80 (0.94)	81.50 (3.24)	80.33 (1.84)	**84.90 (2.01)**	74.05 (6.65)	15.30	16.47	11.90	22.75
Vertebral	97.40 (0.54)	**85.89 (3.74)**	85.00 (3.25)	84.61 (3.95)	77.94 (5.80)	11.51	12.40	12.79	19.46
Weaning	92.10 (0.92)	77.63 (2.35)	77.63 (2.35)	**81.57 (2.86)**	80.26 (1.52)	14.47	14.47	10.53	11.84
Average	95.82	75.93	76.21	**76.29**	73.96	19.89	19.60	19.52	21.85

TABLE III: Mean and standard deviation of the generalisation accuracy rate of the Oracle model and the four chosen DCS techniques for the test dataset, for a pool generated using the proposed method. N_p is the average number of Perceptrons in the pool. The Oracle-DCS accuracy gap column denotes the difference between the mean accuracy rates of the Oracle and each DCS technique. Best results are in bold.

Datasets	N_p	Oracle	DCS accuracy rate				Oracle-DCS accuracy gap			
			OLA	LCA	MCB	MLA	OLA	LCA	MCB	MLA
Adult	3.10 (0.31)	99.83 (0.38)	**88.15 (2.93)**	87.40 (2.82)	88.15 (2.93)	85.66 (1.71)	11.68	12.43	11.68	14.17
Blood	3.00 (0.00)	100.0 (0.00)	75.53 (1.14)	**75.74 (1.04)**	75.53 (1.14)	71.01 (3.26)	24.47	24.26	24.47	28.99
CTG	4.30 (0.47)	100.0 (0.00)	90.24 (0.77)	**90.30 (0.84)**	90.24 (0.77)	24.91 (1.21)	9.76	9.70	9.76	75.09
Faults	9.10 (0.31)	99.67 (0.27)	71.91 (1.60)	**71.99 (1.53)**	71.91 (1.60)	24.50 (0.57)	27.76	27.68	27.76	75.17
German	3.10 (0.31)	99.64 (0.22)	70.04 (2.35)	**70.84 (1.87)**	70.52 (2.08)	63.96 (2.27)	29.60	28.80	29.12	35.68
Glass	4.80 (0.77)	97.92 (1.83)	66.79 (4.17)	**69.43 (3.33)**	66.79 (4.17)	49.62 (2.13)	31.13	28.49	31.13	48.30
Haberman	3.80 (0.41)	100.0 (0.00)	71.58 (5.24)	71.05 (1.91)	**71.71 (4.91)**	70.13 (4.05)	28.42	28.95	28.29	29.87
Heart	3.20 (0.41)	99.56 (0.97)	**86.62 (2.18)**	86.47 (2.85)	86.18 (2.36)	82.79 (5.61)	12.94	13.09	13.38	16.77
Ionosphere	3.70 (0.47)	99.20 (1.05)	87.16 (2.76)	**87.27 (3.21)**	87.16 (2.71)	81.93 (2.78)	12.04	11.93	12.04	17.27
Laryngeal1	2.40 (0.68)	98.49 (1.16)	80.38 (4.26)	**80.94 (4.70)**	80.57 (4.59)	79.06 (5.01)	18.11	17.55	17.92	19.43
Laryngeal3	4.80 (1.11)	99.89 (0.35)	72.25 (1.71)	**72.58 (2.14)**	71.80 (1.58)	43.71 (3.70)	27.64	27.31	28.09	56.18
Liver	3.20 (0.41)	99.77 (0.72)	58.37 (3.53)	**58.37 (2.81)**	58.37 (3.49)	56.40 (2.98)	41.40	41.40	41.40	43.37
Mammographic	2.90 (0.31)	99.66 (0.73)	**82.60 (2.47)**	81.63 (3.06)	**82.60 (2.47)**	81.78 (2.35)	17.06	18.03	17.06	17.88
Monk2	2.50 (0.51)	100.0 (0.00)	86.20 (3.74)	**90.28 (2.18)**	87.96 (3.80)	72.41 (3.80)	13.80	9.72	12.04	27.59
Phoneme	4.40 (0.50)	100.0 (0.00)	86.74 (0.73)	**87.01 (0.77)**	86.73 (0.73)	72.74 (1.06)	13.26	12.99	13.27	27.26
Pima	3.50 (0.51)	99.69 (0.35)	72.29 (2.39)	**73.23 (3.39)**	72.71 (2.67)	69.95 (1.97)	27.40	26.46	26.98	29.74
Sonar	3.30 (0.66)	99.23 (0.97)	**80.00 (3.33)**	78.08 (5.01)	79.81 (3.09)	60.96 (5.65)	19.23	21.15	19.42	38.27
Vehicle	5.60 (0.50)	99.81 (0.32)	70.09 (2.57)	**70.75 (2.22)**	70.14 (2.52)	37.41 (2.18)	29.72	29.06	29.67	62.40
Vertebral	2.50 (0.69)	98.97 (0.79)	81.41 (2.06)	82.31 (1.93)	**82.69 (2.22)**	72.95 (5.78)	17.56	16.66	16.28	26.02
Weaning	3.00 (0.00)	98.55 (1.12)	78.68 (3.71)	78.82 (3.05)	**79.21 (3.30)**	67.11 (2.41)	19.87	19.73	19.34	31.44
Average	3.80	99.49	77.85	**78.22**	78.03	63.45	21.64	21.26	21.45	36.04

80%. As in the previous case, in almost half of the datasets the gap between the generalisation accuracy rates were over 20 percentile points for all DCS techniques, which shows that, even though the proposed method uses the Oracle as a guide to generate the pool, it did not make the recognition rates much closer to the theoretical limit.

It can also be observed in Table III that the Oracle generalisation accuracy rate significantly increased in comparison with the previous case, reaching more than 99% on average. However, the average accuracy rates of all DCS techniques but MLA increased by only a slight difference, which shows that a considerable improvement in the theoretical limit did not lead to a significant increase in classification accuracy by these techniques.

Moreover, the performance of MLA using the pool generated by the proposed method was much inferior to the other three DCS techniques. The cases in which the difference was the greatest were all multi-class datasets. This happened because the local competence estimation in MLA 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. 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 proposed method 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.

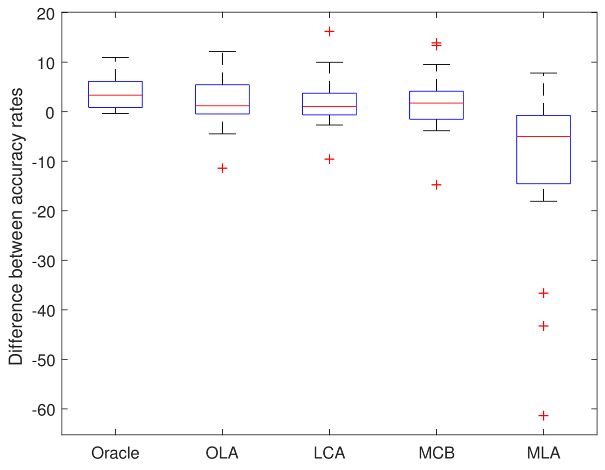


Fig. 3: Difference between the mean generalisation accuracy rates of the proposed method (Table III) and Bagging (Table II), for the Oracle and the four DCS techniques.

To validate the comparison between the behavior of the Oracle in the two previous experiments, Figure 3 shows the boxplot of the difference between the generalisation accuracy rates of the proposed method (Table III) and Bagging (Table II), for the Oracle and all four DCS techniques. It can be observed that the performance of both methods is to some degree similar since the medians of the difference between the accuracy rates are close to zero for all DCS techniques but MLA. Therefore, the results suggest 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. It is also worth noting that the proposed method was able to obtain similar accuracy rates to a pool of size 100 generated by Bagging with considerably fewer classifiers (3.8 in average), as the column N_p from Table III shows. Also, the number of classifiers generated by the proposed method was automatically obtained according to the training data, as opposed to Bagging, which requires the pool size to be set beforehand.

To further investigate the performance of the DCS techniques and its relationship to the Oracle, the training set was used as test set using the pools generated by the proposed method. Table IV shows the hit rate of OLA, LCA, MCB and MLA, 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 16 of Algorithm 1). It is important to remember that the proposed ensemble generation approach guarantees an Oracle of 100%, in other words, each instance in the training set has at least one classifier that correctly classifies it. So, it can be observed that, even though the Oracle for the training dataset is always 100%, the hit rate of the DCS techniques were above 90% in only two of the 20 datasets, 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, which suggests 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 may not be much relevant for dynamic selection schemes.

Furthermore, the results from Table IV shows that the accuracy rate of the DCS techniques in the training data was always similar or greater than their hit rate, which suggests that, when the correct classifier is not chosen, it usually selects another suitable one to label each instance. It also shows the same behavior seen previously regarding the use of the Oracle as a guide to generate the pool, since for an Oracle accuracy rate of 100%, it correctly classified less than 90% of the training instances, on average. This further shows that the Oracle is not a very good information to use when generating a pool of classifiers. The hit rate, on the other hand, captures in a better way the relationship between the

TABLE IV: Mean and standard deviation of the hit rate and the accuracy rate of the DCS techniques using the training dataset for testing, for the same pool generated by the proposed method in Table III. The hit rate is the rate at which the correct classifier for a given training instance is selected by the DCS technique. The theoretical limit is always 100%, since the evaluation is done over the training set. Best results are in bold.

Dataset	**Hit rate**				**Accuracy rate**			
	OLA	**LCA**	**MCB**	**MLA**	**OLA**	**LCA**	**MCB**	**MLA**
Adult	86.91 (0.87)	86.77 (0.92)	**87.14 (0.73)**	80.95 (0.28)	88.43 (0.66)	**89.23 (0.41)**	88.70 (0.50)	84.43 (0.62)
Blood	79.59 (0.51)	**80.20 (0.35)**	79.61 (0.51)	66.29 (1.14)	82.46 (0.34)	**82.55 (0.38)**	82.50 (0.37)	69.41 (1.16)
CTG	92.50 (0.59)	**92.63 (0.44)**	92.49 (0.63)	23.93 (0.60)	93.86 (0.40)	**93.92 (0.43)**	93.86 (0.40)	24.13 (0.68)
Faults	**76.88 (1.26)**	76.84 (1.01)	76.87 (1.26)	24.49 (0.25)	80.96 (0.69)	**81.84 (0.65)**	80.96 (0.69)	24.83 (0.33)
German	71.05 (1.44)	**75.75 (1.35)**	71.23 (1.47)	45.29 (3.53)	82.75 (0.91)	**83.19 (0.56)**	83.05 (0.91)	65.64 (1.97)
Glass	76.21 (1.98)	**77.95 (1.92)**	76.27 (1.99)	46.83 (2.28)	77.64 (1.66)	**79.75 (1.71)**	77.58 (1.60)	47.64 (2.52)
Haberman	76.26 (1.10)	**76.61 (1.46)**	76.35 (1.10)	65.83 (4.34)	80.52 (1.76)	**80.78 (1.89)**	80.65 (1.57)	70.83 (3.33)
Heart	**84.06 (1.92)**	83.86 (2.40)	83.96 (1.72)	72.48 (5.83)	86.19 (0.98)	**87.23 (1.06)**	86.09 (0.66)	80.69 (3.32)
Ionosphere	86.46 (1.48)	**87.34 (1.53)**	86.43 (1.43)	75.74 (1.71)	**90.08 (0.96)**	89.35 (0.97)	86.25 (3.03)	79.58 (2.07)
Laryngeal1	84.75 (2.07)	**84.81 (2.38)**	84.75 (1.93)	78.38 (3.53)	85.44 (2.13)	**86.00 (1.77)**	85.44 (1.97)	79.38 (2.95)
Laryngeal3	74.81 (2.95)	73.98 (1.99)	**74.85 (2.90)**	38.75 (0.83)	80.15 (1.64)	79.92 (1.28)	**80.19 (1.55)**	43.11 (3.24)
Liver	67.22 (1.40)	**70.62 (2.91)**	67.34 (1.28)	44.71 (2.86)	79.19 (3.12)	**81.47 (3.05)**	79.31 (3.06)	58.80 (1.16)
Mammographic	82.72 (0.64)	**82.83 (1.54)**	82.68 (0.73)	76.53 (1.34)	84.20 (1.12)	**84.34 (1.20)**	84.16 (1.22)	79.23 (1.29)
Monk2	85.77 (3.60)	**91.82 (3.61)**	86.67 (4.48)	49.23 (10.62	94.81 (1.02)	**96.54 (1.00)**	95.71 (0.59)	70.80 (3.98)
Phoneme	87.40 (0.46)	**89.48 (0.44)**	87.40 (0.47)	70.16 (0.62)	92.20 (0.35)	92.29 (0.38)	**98.59 (0.12)**	71.68 (0.60)
Pima	75.64 (1.55)	**76.02 (1.67)**	75.82 (1.83)	59.76 (2.17)	81.65 (0.76)	**82.66 (0.75)**	81.81 (0.67)	69.53 (0.95)
Sonar	80.00 (3.62)	**83.46 (3.45)**	80.19 (3.63)	48.33 (5.71)	88.91 (1.50)	88.46 (1.72)	**89.04 (1.48)**	64.55 (2.81)
Vehicle	76.14 (1.49)	**77.98 (1.57)**	76.20 (1.51)	36.92 (1.01)	80.98 (0.98)	**81.69 (0.72)**	80.99 (0.97)	39.50 (0.96)
Vertebral	82.39 (2.14)	**84.33 (2.32)**	82.39 (2.19)	69.46 (3.69)	86.31 (1.35)	**86.76 (1.16)**	86.31 (1.27)	73.60 (2.51)
Weaning	83.45 (1.33)	**84.38 (1.72)**	83.36 (1.20)	56.11 (1.42)	88.27 (1.26)	**88.81 (1.15)**	88.23 (1.10)	65.66 (3.06)
Average	80.51	**81.88**	80.60	56.50	85.25	**85.84**	85.47	63.15

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.

IV. Conclusion

In this work, a general view of the Oracle model and its usage in the literature was introduced. More specifically, the relationship between the Oracle model and DS schemes was discussed, and it was claimed that, although the model indicates the upper limit for DCS techniques, these techniques usually struggle to select the classifier deemed by the Oracle the best one in the pool. This difficulty is characterized by the notable difference between the Oracle and the DCS techniques accuracy rates.

With that in mind, it was proposed an ensemble generation method that guarantees an Oracle accuracy rate of 100% in the training set. The proposed method is an incremental ensemble generation method which generates binary classifiers by placing hyperplanes in the feature space until at least one classifier correctly classifies each training instance in the pool. This method is faster than classical ensemble methods, since the classifiers are not trained, and it can find the pool size automatically according to the training data. The purpose of using such a method was 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.

Experiments have shown that there is a significant gap between the theoretical limit and the DCS techniques for pools generated without Oracle information and that difference remained roughly the same for pools with an Oracle accuracy rate of 100% in the training set. In fact, for a theoretical limit of 100%, the average accuracy rate was approximately 85% in the best case, which shows the DCS techniques have difficulty in selecting the most competent classifier, despite there being one for sure in the pool. The reason for this is that, although the Oracle model is performed globally, DCS techniques use only local data to select the best classifier, which suggests that, despite its use in the literature for such a task, the Oracle model is not the best guide in the search for a promising pool for DCS techniques. It was also suggested that the hit rate, that is, the rate at which a DCS technique selects the right classifier for a given instance, was a better guide in this search, since it conveys both local and global information regarding the performance of a pool of classifiers using DS schemes.

A further investigation of the use of the hit rate in ensemble generation methods for dynamic selection techniques may be conducted in the future. Moreover, since the proposed method usually generates only one competent classifier per training instance, the effect of multiple correct choices during the selection phase was not examined. Thus, an analysis regarding the importance of redundancy, that is, the presence of more than one competent classifier per sample in the pool, for DS techniques may also be performed henceforth. Furthermore, studies regarding the diversity in the generation process by the proposed method may be conducted in the future.

Acknowledgment

This work was supported by Brazilian agencies: CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) and FACEPE (Fundação de Amparo à Ciência e Tecnologia de Pernambuco).

References

[1] L. I. Kuncheva, "A theoretical study on six classifier fusion strategies," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 2, pp. 281–286, 2002.

[2] L. Didaci, G. Giacinto, F. Roli, and G. L. Marcialis, "A study on the performances of dynamic classifier selection based on local accuracy estimation," *Pattern Recognition*, vol. 38, no. 11, pp. 2188–2191, 2005.

[3] L. I. Kuncheva and C. J. Whitaker, "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy," *Machine Learning*, vol. 51, no. 2, pp. 181–207, 2003.

[4] G. Brown, J. Wyatt, R. Harris, and X. Yao, "Diversity creation methods: a survey and categorisation," *Information Fusion*, vol. 6, no. 1, pp. 5–20, 2005.

[5] A. Britto, R. Sabourin, and L. Oliveira, "Dynamic selection of classifiers - A comprehensive review," *Pattern Recognition*, vol. 47, no. 11, pp. 3665–3680, 2014.

[6] A. H.-R. Ko, R. Sabourin, and 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.

[7] L. I. Kuncheva and J. J. Rodriguez, "Classifier ensembles with a random linear oracle," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 4, pp. 500–508, 2007.

[8] R. M. O. Cruz, R. Sabourin, G. D. C. Cavalcanti, and T. I. Ren, "META-DES: A dynamic ensemble selection framework using meta-learning," *Pattern Recognition*, vol. 48, no. 5, pp. 1925–1935, 2015.

[9] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[10] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832–844, 1998.

[11] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee, "Boosting the margin: a new explanation for the effectiveness of voting methods," in *14th International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., 1997, pp. 322–330.

[12] A. H. R. Ko, R. Sabourin, and A. S. B. Jr., "From dynamic classifier selection to dynamic ensemble selection," *Pattern Recognition*, vol. 41, no. 5, pp. 1718–1731, 2008.

[13] K. Bache and M. Lichman, "UCI machine learning repository," Available: http://archive.ics.uci.edu/ml, 2013, [Online].

[14] L. Kuncheva, "Ludmila Kuncheva Collection," Available: http://pages.bangor.ac.uk/~mas00a/activities/real_data.htm, 2004, [Online].

[15] R. D. King, C. Feng, and A. Sutherland, "Statlog: comparison of classification algorithms on large real-world problems," *Applied Artificial Intelligence*, vol. 9, no. 3, pp. 289–333, 1995.

[16] J. Alcalá, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera, "KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework," *Journal of Multiple-Valued Logic and Soft Computing*, vol. 17, no. 2-3, pp. 255–287, 2011.

[17] C. Jutten, "The enhanced learning for evolutive neural architectures project," Available: https://www.elen.ucl.ac.be/neural-nets/Research/Projects/ELENA/elena.htm, 2002, [Online].

[18] K. Woods, W. P. Kegelmeyer Jr, and K. Bowyer, "Combination of multiple classifiers using local accuracy estimates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 4, pp. 405–410, 1997.

[19] P. C. Smits, "Multiple classifier systems for supervised remote sensing image classification based on dynamic classifier selection," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 40, no. 4, pp. 801–813, 2002.

[20] G. Giacinto, F. Roli, and G. Fumera, "Selection of classifiers based on multiple classifier behaviour," in *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition and Structural and Syntactic Pattern Recognition*. Springer-Verlag, 2000, pp. 87–93.

[21] R. M. Cruz, R. Sabourin, and 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.

[22] A. Roy, R. M. Cruz, R. Sabourin, and G. D. Cavalcanti, "Meta-learning recommendation of default size of classifier pool for META-DES," *Neurocomputing*, vol. 216, pp. 351–362, 2016.