Neural Comput & Applic (2013) 22:673–688
DOI 10.1007/s00521-011-0737-9

ORIGINAL ARTICLE

Dynamic selection approaches for multiple classifier systems

Paulo R. Cavalin · Robert Sabourin · Ching Y. Suen

Received: 16 March 2011 / Accepted: 27 August 2011 / Published online: 17 September 2011

Abstract In this paper we propose a new approach for dynamic selection of ensembles of classifiers. Based on the concept named multistage organizations, the main objective of which is to define a multi-layer fusion function adapted to each recognition problem, we propose dynamic multistage organization (DMO), which defines the best multistage structure for each test sample. By extending Dos Santos et al.'s approach, we propose two implementations for DMO, namely DSA^m and DSA^c. While the former considers a set of dynamic selection functions to generalize a DMO structure, the latter considers contextual information, represented by the output profiles computed from the validation dataset, to conduct this task. The experimental evaluation, considering both small and large datasets, demonstrated that DSA^c dominated DSA^m on most problems, showing that the use of contextual information can reach better performance than other existing methods. In addition, the performance of DSA^c can also be enhanced in incremental learning. However, the most important observation, supported by additional experiments, is that dynamic selection is generally preferred over static approaches when the recognition problem presents a high level of uncertainty.

Keywords Multiple classifier systems · Adaptive system · Dynamic selection · Incremental learning · Multistage organizations · Ensembles of classifiers

1 Introduction

Over the past decades, Multiple Classifier Systems have emerged as a viable alternative to make pattern recognition systems achieve lower and lower error rates. This kind of system can be composed of either existing classifiers, aiming at enhancing their individual performances, or classifiers constructed by an automatic method, to which we refer as ensembles of classifiers (EoCs). In both cases, nonetheless, it is well-known that the set of classifiers must contain members that are complementary and diverse [1, 2], so that the combined classifiers outperform the best member of the set.

The task of finding the aforementioned complementary and diverse set of classifiers is not trivial. Actually, the performance of the fusion function, which carries out the combination of the decisions provided by the base classifiers, may heavily depend on such a "good" set of classifiers [3]. For example, it has been shown that the performance of the majority voting function, which is a widely used combination rule, significantly improves for the case of negatively correlated classifiers [4, 5]. However, to construct an EoC with negatively correlated classifiers remains a very unlikely situation in real-world classification problems, and their benefits remain out of reach. If existing classifiers, to which we have no access to

P. R. Cavalin (✉)
Universidade Federal do Tocantins (UFT), Quadra 109 Norte Av. NS15 s/n Bl. II sala 21, 77001-090 Palmas, TO, Brazil
e-mail: cavalin@uft.edu.br

R. Sabourin
École de Technologie Supérieure (ETS), 1100 Notre-dame ouest, Montréal, QC H3C-1K3, Canada
e-mail: robert.sabourin@etsmtl.ca

C. Y. Suen
Centre for Pattern, Recognition and Machine Intelligence (CENPARMI), Concordia University, 1455 de Maisonneuve Blvd West, Montréal, QC H3G-1M8, Canada
e-mail: suen@cse.concordia.ca

change its parameters, are included in the pool, this task may become even less evident.

One way to enhance the use of multiple classifiers, is to define a fusion scheme that takes greater advantage of the diversity presented by the base classifiers, even though such a diversity is not so apparent at first. In other words, we need to define a way to expand the limits of the combination method, to better use the existing diversity of the pool of classifiers. One interesting approach, named multistage organizations (MO), has been proposed in [5, 6] for such an objective.

The main advantage of using MO relies on the ability to construct a multistage structure, which represents the fusion function, that is adapted to each recognition problem. Such an adaptation is achieved by defining the relationships between consecutive layers based on evidences provided by the training data. Nevertheless, only a single structure is created, in an ad-hoc fashion, for all the test samples. Due to its static nature, the method might not be able to handle all the difficulties presented by complex recognition patterns, which supposedly has the same drawback of static approaches to select classifiers.

To deal with those issues, we propose dynamic multistage organizations (DMO), inspired by dynamic selection of classifiers. The main idea consists of defining the multistage structure that best adapts to each test sample. In this case, not only the fusion function adapts to each problem, but also, to each test sample. Such a structure also takes into account an automatic weighting approach, which selects the best weight for each classifier output based on the current test sample.

One approach that is closely related to the idea of DMO is Dos Santos et al.'s (DSA) approach [7]. In this case, one EoC is dynamically selected, from a pool of EoCs, by means of evaluating only the outputs yielded by the members of each ensemble. If we can, for example, select more than one ensemble at a time, we can better generalize the DMO concept, by implementing a two-stage DMO structure. Given these standpoints, we propose two original frameworks based on DSA.

The first framework, named DSA^m, consists of validating the DMO concept, in which we exploit the use of a set of dynamic selection functions to create a DMO structure. In this case, each function performs the selection of an EoC. Note that the main advantage of this method lies on its simplicity. In the second framework, namely DSA^c, we use contextual information to find the best DMO structure based on problem-related knowledge. The evidences produced by the validation set are taken into account in this case, whereas the structure is defined by considering the most similar validation samples using case-based reasoning. The architecture of DSA^c is not only easily adaptable to different problems, but also is incremental-learning ready.

This work aims at accomplishing two main objectives during the experimental evaluation. The first objective is to evaluate both DSA^c and DSA^m against static methods, to observe whether the proposed DMO concept can result in better performance or not. In addition, we aim at evaluating the conditions under which dynamic selection might outperform static selection. Given that in the literature dynamic selection methods are generally compared to static methods for recognizing a given problem, in a single static condition in terms of recognition problem, the goal of these experiments is to provide more insights related to which conditions a dynamic selection approach might be more preferable than a static one. The *NIST-digits* database allows us to simulate these different conditions, as explained later.

The remainder of this paper is organized as follows. In Sect. 2, we describe static and dynamic selections, providing more details about Dos Santos et al.'s approach, to support the content of the subsequent sections. In Sect. 3 we describe the proposed DMO concept with greater detail. Both DSA^c and DSA^m are described in Sect. 4, and in Sect. 5, we present the experimental protocol and the results that were obtained. Finally, in Sect. 6, we present conclusions and point out the future work.

2 Background theory

In this section, we present an overview of dynamic selection methods (DS), in which we also describe Dos Santos et al.'s approach (DSA) in detail.

2.1 Dynamic selection (DS)

Suppose a multiple classifier system is composed of a pool of base classifiers, to which we refer as C. The goal of dynamic selection is to find a subset of classifiers C'_i, where $C'_i \subset C$, which is the best one, by considering all local criteria, to classify the test sample $x_{i,test}$. Note that, in static selection, a single subset C', where $C' \subset C$, is globally selected to recognize all test samples.

In the literature, dynamic selection is divided into dynamic selection of classifiers (DSC), where only a single classifier is selected for each test sample [8–10], and dynamic selection of ensembles of classifiers (DSEoC), where an EoC is selected for each test sample [7, 11, 12].

Usually, the main goal of the systems for both DSC and DSEoC is to find the best subset of classifiers C'_i to classify $x_{i,test}$. This best set is generally associated with the highest level of competence, which is computed by means of, for instance, K nearest neighbors [8], clustering [13], and multiple training datasets [14]. In order to compute the level of competence by using one of these methods, we

must deal with the following issues: a robust feature set must be defined for a desirable reliability, which is not trivial; these approaches are very expensive in terms of computational complexity; and it is not possible to use some types of base classifiers, such as human experts or HMMs, since they do not use feature vectors to conduct the classification task. The KNORA algorithm [12], however, is an example of an approach that tries to overcome some of these issues. The only information this method requires from the base classifiers is whether or not they correctly classify a given validation sample. Nonetheless, KNORA also depends on a very robust feature set to compute similarity between validation samples and the test sample.

A more general approach, though, is Dos Santos et al.'s, which dynamically selects EoCs, whose levels of competence are computed by using only the outputs of their members, based on the extent of consensus. This property makes it a very general approach in terms of base classifier and feature set. However, many sources of knowledge embedded in the structure of DSA have not been exploited yet, for instance, the outputs produced by the base classifiers. Thus, we believe the performance of this method can be improved, resulting in an approach that is both robust and general at the same time. For the sake of completeness, in the remainder of this section we present this method in greater detail.

2.2 Dos Santos et al.'s approach (DSA)

The overall architecture of DSA is depicted in Fig. 1. The main objective of this method is to dynamically find the best EoC, whose members are a subset of $C = \{c_1, c_2, \ldots, c_N\}$, to recognize the test sample $x_{i,test}$. This task is performed by considering only the recognition outputs $O_i = \{o_{i,1}, \ldots, o_{i,N}\}$ computed from C. Each output corresponds to a class from the set $\Omega = \{\omega_1, \ldots, \omega_M\}$.

DSA is divided into two phases: the design phase and the operational phase.

During the design phase, which is performed off-line, it creates the architecture that supports the dynamic selection of EoCs. In other words, the pool of EoCs $C^{*'} = \{C'_1, \ldots, C'_W\}$ where $C'_j \subset C, 1 \leq j \leq W$, is created during this phase. Given that $C^{*'}$ is a subset of all possible EoCs C^*, the main objective is to reduce the complexity for the operational phase since $|C^*|$ is much larger than $|C^{*'}|$ and the time needed to find the best EoCs in considering C^* would be impractical in most applications. The pool $C^{*'}$ is generated by a search algorithm, which is a genetic algorithm in this work. Each individual is represented by a binary vector of N positions, where each bit represents whether or not a classifier is selected as a member of an EoC. The fitness function, which has to be minimized, uses the error rate on the optimization set *Opt*, by applying the majority voting method on the EoCs assigned by each individual. In order to avoid overfitting, each individual is also evaluated on the validation set *Val*, and the best solutions are saved into an archive whose size is W. The archive is then used as $C^{*'}$.

Throughout the operational phase, the dynamic selection of the best EoC C''_i is performed, which consists of a member of the pool of EoCs $C^{*'}$, to recognize the test sample $x_{i,test}$. After the outputs O_i of the set of base classifiers C are computed, we check which member of the pool of EoCs $C^{*'}$ is best to recognize $x_{i,test}$. For each EoC, we apply the dynamic selection function λ to evaluate whether it is the best ensemble or not. The best EoC is then stored in C''_i, the dynamically selected EoC. Finally, the ensemble that was dynamically selected is used to compute the class with the highest number of votes, which is the final decision d_i.

Note that λ can be related to one of the five functions described in Sect. 4.1.1. In this work, λ is computed by taking into account the extent of consensus, as defined in (2) [7].

3 Dynamic multistage organizations (DMO)

The main inspiration for dynamic multistage organizations is multistage organizations (MO). MO consists of structuring classifiers into relevant multistage layers. The outputs of the classifiers are reorganized into subsequent levels, and these outputs are re-evaluated at each level. By structuring classifiers in multi-steps, the main premise is that the influence of individual errors on the final error of the combined systems can be reduced, since the outputs are transformed to another space corresponding to the fusion of some selected classifiers. Hence, given the fact that both selection and fusion are conducted at the same time, the diversity among the classifiers is better exploited, and the limits of majority voting error are widened.

The main advantage of MO is that the whole structure can be defined for a given problem. For example, in [5] a genetic algorithm is used to optimize the MO structure given problem-related training data. Nonetheless, a single structure is defined for all test samples, which, as a consequence, might not cover the different difficulties presented by all test samples in a complex recognition problem. To deal with this issue, we propose DMO, inspired by dynamic selection of classifiers.

DMO basically consists of defining the best multistage structure for each test sample. In this case, the relationships between the outputs are dynamically defined, according to the current test sample $x_{i,test}$. It also takes into account a

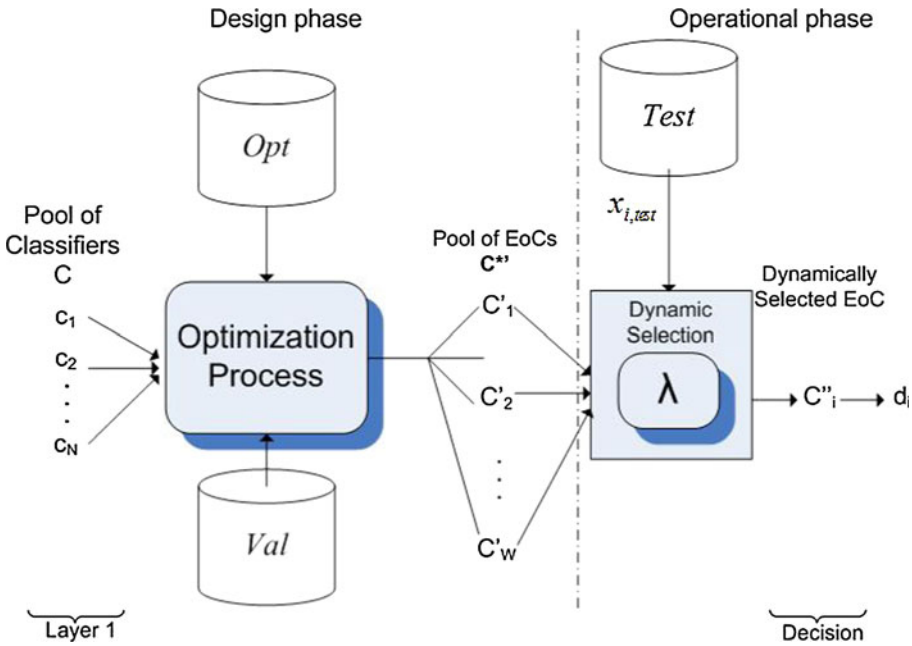

Fig. 1 Dos Santos et al.'s approach (DSA). The pool of classifiers is organized into another pool of EoCs during the design phase. During the operational phase, the EoC, which is dynamically selected by λ, produces the final decision

dynamic weighting approach for further improvements. Note that, instead of using the same structure to recognize all test samples, which might be suboptimal, we define the structure that better models the relationships among the base classifiers, according to the information provided by $x_{i,test}$. By doing so, we may enhance the overall performance of the system not only by using a multi-stage approach, but also by using a dynamic approach that better fits the difficulties presented by each test sample.

In order to illustrate DMO, we use a synthetic recognition example with five binary classifiers. In Fig. 2a, we present a test sample, whose correct label is 1, being recognized by MO. Suppose this MO structure has been considered optimal during the design phase. We can see, though, that this structure does not correctly recognize this test sample. However, as shown in Fig. 2b, by using a DMO approach, we might be able to define a MO structure specifically for this test sample, which can correctly compute the correct class. In this case, given that an EoC that provides the correct answer is selected twice (i.e. it has a heavier weight) to compose the final layer, the correct answer is successfully computed.

One existing method that partially implements the DMO concept is Dos Santos et al.'s approach (see Sect. 2.2), as depicted in Fig. 2c. In this case, one EoC is dynamically selected, from a pool of EoCs, by means of evaluating only the outputs yielded by the members of each ensemble. If we can, for example, select more than one EoC at a time, we can better generalize the DMO concept, by implementing a two-stage DMO structure. For this reason, we extend the architecture of DSA to implement DMO.

4 Extending Dos Santos et al.'s approach to implement DMO

We propose two methods to extend Dos Santos et al.'s approach to implement a dynamic multistage organization. These methods, named DSA^m and DSA^c respectively, are described in the following sections.

4.1 *DSA*^{m}: introducing DMO and high-level decision making

The first framework consists of adding two main extensions to DSA. We refer to this framework as DSA^m, since the use of multiple dynamic selection functions has enabled the implementation of the first extension.

The first extension consists of characterizing the main DSA structure as dynamic multistage organizations. Instead of selecting a single EoC, as in DSA, we now have to select a set of EoCs. The main idea is to compose the second layer of a DMO structure by using this set of EoCs. To achieve this task, we adapt some components of the operational phase. To recognize $x_{i,test}$ we select the set of EoCs $C^{*\prime\prime}_i = \{C''_{i,1}, \ldots, C''_{iU}\}$, as presented in Fig. 3.

Algorithm 1 describes each step of the proposed method. Once the outputs of the base classifiers O_i are computed in step 2, we evaluate each EoC individually. By considering the set of functions $\Lambda = \{\lambda_1, \ldots, \lambda_U\}$, we evaluate each member of $C^{*'}$. The best EoCs, according to Λ, form the set of dynamically selected EoCs $C^{*\prime\prime}_i$. Note that $|C^{*\prime\prime}_i| = U$, since each λ_k selects an EoC, i.e. $C_{i,k}{}''$. It is also worth noting that an EoC may be selected more than once,

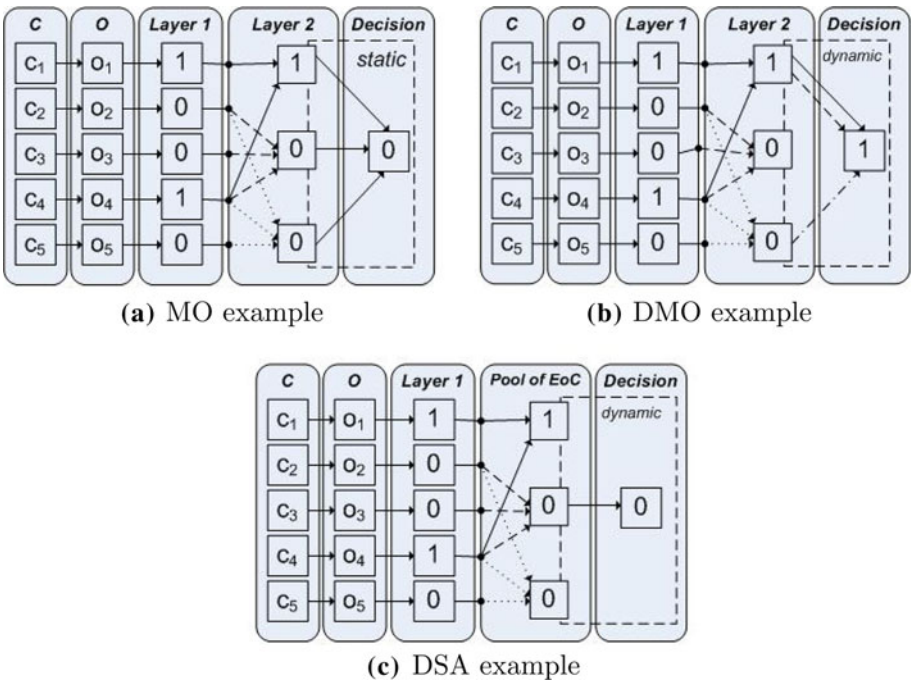

Fig. 2 **a** The sequence of stages processed by multistage organizations (MO), for an example with five classifiers with binary outputs. In this case, each member of layer 2 always provides one vote for the final decision. **b** The same example with dynamic multistage organization (DMO), whereas a member from layer 2 may provide none, one, or more than one vote. **c** The same example using Dos Santos et al.'s approach (DSA), where only a single member of layer 2 gives a vote. Class 1 is the right output in this example

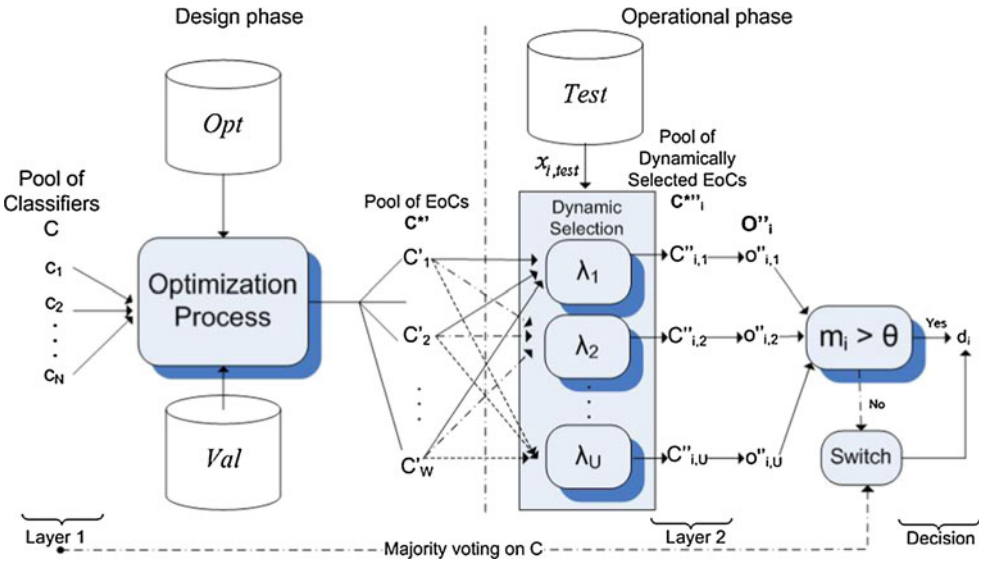

Fig. 3 An overview of the DSA^m approach. This method uses the set of dynamic selection functions Λ to dynamically select a set of EoCs, which results in a two-layer DMO structure

which results in the automatic weighting approach demonstrated in Fig. 2b. In this case all the functions described in Sect. 4.1.1 are used to compose Λ, thus $\Lambda = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5\}$ and $|\Lambda| = 5$. In the example presented in Fig. 2b, in contrast, we consider $|\Lambda| = 3$.

After $C_i^{*\prime\prime}$, the set of dynamically selected EoCs, is defined, the outputs of these EoCs $O''_i = \{o''_{i,1}, \ldots, o''_{i,U}\}$ are computed (step 16 of Algorithm 1). These outputs represent the majority voting class computed from each member in $C_i^{*\prime\prime}$. Then, O''_i is submitted to the switch module. The proposed switch mechanism represents the second extension to DSA. This mechanism, which is represented by steps 18–22 in Algorithm 1, is explained in detail in the following paragraphs.

Despite the expected improvements that a dynamic multistage structure can bring to DSA, we have no guarantee that this complex structure is really better than the pool of base classifiers. In some cases, for example, the dynamically selected EoCs, i.e. $C_i^{*\prime\prime}$, might provide low-confidence results, yielding a tie or the answers below some acceptable confidence level. Note that it is important to detect these cases to avoid random decisions, and select a better source of knowledge, that may be the base classifiers. For this reason, we propose a switch mechanism.

Algorithm 1 DSA^m. $best\_score(k)$ and $score(k)_{j,i}$ represent temporary variables to compute the best EoC, for each of the five functions presented in Sect. 4.1.1

1: **for** each data point $x_{i,test}$ on *Test* **do**
2: Compute $O_i = \{o_1, \ldots, o_N\}$ by considering $C = \{c_1, \ldots, c_N\}$
3: Initialize $best\_score(k)$, $\forall\ \lambda_k$ in Λ
4: **for** each C'_j in $C^{*'}$ **do**
5: **for** each λ_k in Λ **do**
6: Compute $score(k)_{j,i}$ by considering λ_k.
7: **if** $score(k)_{j,i}$ is better than $best\_score(k)$ **then**
8: $C''_{i,k} = C'_j$
9: $best\_score(k) = score(k)_{j,i}$
10: **end if**
11: **end for**
12: **end for**
13: **for** each λ_k in Λ **do**
14: $o''_{i,k}$ = most voted class from $C''_{i,k}$
15: **end for**
16: Compute m_i from $O''_i = \{o''_{i,1}, \ldots, o''_{i,U}\}$ # *see* (1)
17: # *Switch mechanism*
18: **if** $m_i > \theta$ **then**
19: d_i = most voted class from O''_i
20: **else**
21: d_i = most voted class from C
22: **end if**
23: **end for**

Here is the main idea of the switch. First, we employ the concept of margin [15] (see 1, where $v1_i$ and $v2_i$ are, respectively, the most voted and the second most voted classes for $x_{i,test}$) to identify whether or not the answers provided by $C_i^{*''}$ are confident enough, as shown in step 18 of Algorithm 1. When the margin m_i computed by the outputs $C_i^{*''}$ is above the threshold θ, e.g. $m_i > \theta$, we consider that the dynamically selected EoCs are reliable enough and simply use the most voted class in considering O''_i as the final decision d_i (step 19). In contrast, when $m_i \leq \theta$, we switch to the pool of base classifiers and use O_i, i.e. the outputs of C, to compute the most voted class (step 21). This most voted class is used as the final decision d_i. Note that one advantage of the switch mechanism is that instead of relying on random guess, since in the case of a tie we would have to randomly pick one class as the final decision, we use another source of knowledge that is embedded in the architecture of the system to compute such a decision.

$$m_i = v1_i - v2_i \tag{1}$$

In the next section, we describe the dynamic selection functions that are used in step 6 of Algorithm 1 to compute the corresponding score of each λ_k.

4.1.1 Consensus-based dynamic selection functions

The five functions involved in this work, are computed by taking into account the number of votes for each class in Ω, provided by each candidate C'_i. We aimed at using only functions which can compute the level of competence of each EoC based on the votes of the base classifiers. One reason is to avoid the complexity of functions that compute regions of competence based on evaluating distances between $x_{i,test}$ and prototypes in the feature space, as in [8, 11]. Another reason is to enable this approach to deal with any category of base classifier that can output votes.

In this section we use the following notation: $v_{k,j,i}$ is the number of votes for class ω_k provided by C'_j given the test sample $x_{i,test}$, p_j is the global performance of C'_j, and $p_{j,k}$ is the performance of C'_j for class ω_k, both measured on the validation set *Val*; $mv_{j,i}$ represents the majority voting class provided by C'_j given the sample $x_{i,test}$, e.g. $mv_{j,i} = \text{argmax}\ v_{k,j,i}\ \forall\ k$. The cardinality of C'_j is represented by $|C'_j|$.

4.1.1.1 λ_1: ambiguity-guided dynamic selection (ADS) This function is presented in [7]. It selects the solution whose outputs produce the lowest ambiguity, represented by the number of classifiers in disagreement with the majority voting class.

The ambiguity $\gamma_{j,i}$, given C'_j and the test sample $x_{i,test}$, can be computed by the minimization of the following equation:

$$\gamma_{j,i} = \frac{\sum_1^k v_{k,j,i} \forall k \neq mv_{j,i}}{|C'_j|} \tag{2}$$

4.1.1.2 λ_2: margin-based dynamic selection (MDS) This function selects the solution with the highest margin [7]. The margin represents the difference between the majority voting and the second highest number of votes. The main idea is to select the solution that produces the largest difference in number of votes between the highest consensus and the second highest.

The maximization of the following equation, given C'_j and the sample $x_{i,test}$, allows us to dynamically select the most competent candidate by using the margin $\mu_{j,i}$:

$$\mu_{j,i} = \frac{v_{k,j,i} - max_{l \neq k} v_{l,j,i}}{|C'_j|}, \quad \text{where } k = mv_{j,i} \tag{3}$$

4.1.1.3 λ_3: class-strength dynamic selection (CSDS) This function weighs the selection of the best solution [7]. In this case, the margin, as described in (3), is multiplied by $p_{j,k}$. The main idea is to select the candidate that provides the best trade-off between the margin and the performance for recognizing the class with the highest number of votes.

In considering the margin as $\mu_{j,i}$ and the class performance as $p_{j,k}$, the maximization of the following equation

leads us to find the most competent C'_j for $x_{i,test}$ by using CSDS:

$$\Theta_{j,i} = \mu_{j,i} * p_{j,k}, \quad \text{where } k = mv_{j,i} \tag{4}$$

4.1.1.4 λ_4: Pair of votes dynamic selection (PVDS) We propose a new function aiming at selecting EoCs that concentrate their decisions on only two classes. In this case, both values for margin and consensus might be very low, which is counter-intuitive according to other DSFs such as ADS and MDS. However, we suppose that these EoCs are likely to produce less random guesses and wrong decisions, since they concentrate their decisions on reduced boundaries, e.g. only two classes.

In order to implement this idea, we simply sum the number of votes for the top-two classes, and maximize this value. This is represented by $\eta_{j,i}$. Given C'_j and the sample $x_{i,test}$, $\eta_{j,i}$ can be computed by using the following equation:

$$\eta_{j,i} = \frac{v_{k,j,i} + max_{l \neq k} v_{l,j,i}}{|C'_j|}, \quad \text{where } k = mv_{j,i} \tag{5}$$

4.1.1.5 λ_5: global-strength dynamic selection (GSDS) This function is a modification of CSDS. In this case, we consider the global performance p_j of C'_j to weigh the value provided by the margin. The main supposition is that the global performance is more robust than the performance to recognize a specific class to indicate the most competent solution.

Given p_j, C'_j, and $x_{i,test}$, this function can be computed by maximizing the following equation:

$$\iota_{j,i} = \mu_{j,i} * p_j \tag{6}$$

In the next section, we present the second method proposed in this work, whose main goal is to replace these dynamic selection functions by a context-based approach.

4.2 DSA^c: enhancing dynamic selection by using contextual information

Both DSA and DSA^m dynamically select EoCs by considering dynamic selection functions based on the extent of consensus. Despite that the extent of consensus is a well studied concept in the literature [15], only the outputs of the most voted and the second most voted classes are used to select the ensemble. However, the information related to the other classes is wasted, even though such information could help this task. In order to overcome this drawback, we propose DSA^c, which is depicted in Fig. 4.

DSA^c is inspired by both decision templates [16] and the KNORA algorithm [12]. The main objective is to use the validation database, transformed into output profiles, to point out which EoCs are the most competent to recognize the test sample $x_{i,test}$. An output profile is computed by transformation T in (7), where $x_i \in \Re^D, \tilde{x}_i \in \mathbb{Z}^{N+}$, and N is the size of the pool of base classifiers C. Given that we know which EoC correctly recognizes each validation sample, a DMO structure is defined by computing which validation samples are the ones most similar to the test samples in considering the output profiles, and composing the dynamically selected set of EoCs with the EoCs that correctly classify these validation samples.

$$T : x_i \Rightarrow \tilde{x}_i, \tag{7}$$

In greater detail, this approach works as follows. Consider the pool of EoCs $C^{*'}$, generated during the design phase. For each test sample $x_{i,test}$, we compute the best set of EoCs $C^{*''}{}_i$, composed of members from $C^{*'}$. Each EoC from $C^{*'}$ may appear several times in $C^{*''}{}_i$, resulting in an automatic weighting approach. This task is achieved by considering the function ζ, which is depicted in Fig. 5.

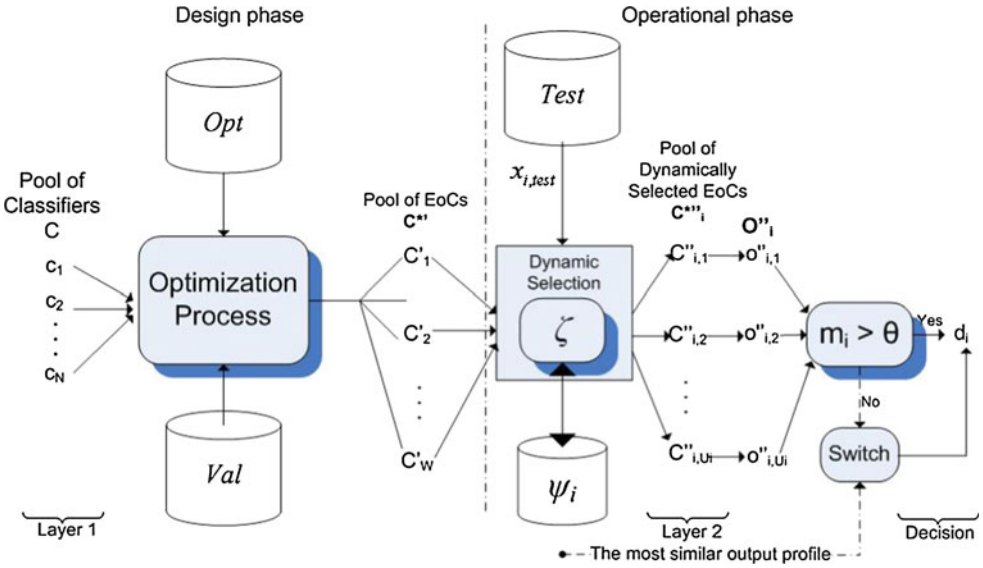

Fig. 4 An overview of the DSA^c approach. This method uses the knowledge provided by *Val* (converted into the set of output profiles *Val'*)

Algorithm 2 describes this method in detail. The first few steps represent the function ζ. First, in step 3 we apply T on $x_{i,test}$, resulting in $\tilde{x}_{i,test}$. Next, as presented in step 4, we compare $\tilde{x}_{i,test}$ to each output profile in Val', which is a database containing the output profiles of all validation samples in Val', $\tilde{x}_{j,val}\,\forall x_{j,val} \in Val$, computed in step 1. We compare these samples in terms of similarity, and save the degree of similarity between $\tilde{x}_{i,test}$ and $\tilde{x}_{j,val}$ in the variable $\delta_{i,j}$. Note that we use the similarity measure presented in (8) to compute $\delta_{i,j}$. The K most similar output profiles $\tilde{x}_{j,val}$, e.g. the validation samples related to the highest values of $\delta_{i,j}$, are stored in Ψ_i. Next, as shown in steps 7 to 11, for each sample in Ψ_i and each member of the pool of EoCs $C^{*\prime}$, we compute if the EoC provides the correct recognition result for this sample. In the case of a positive answer, this EoC is included in $C^{*\prime\prime}{}_i$, worth noting that an EoC appears in $C^{*\prime\prime}{}_i$ as many times as the number of samples that it correctly recognizes. Finally, C_i^* is submitted to the switch mechanism DSA^c.

Steps 15–19 in Algorithm 2 represent the switch module in Fig. 4, which corresponds to the previously mentioned switch mechanism. Similar to DSA^m, it is computeed whether the margin m_i, in considering the dynamically selected EoCs C_i^*, is above the threshold θ or not. If $m_i > \theta$, then we use the most voted class indicated by $C_i^{*\prime\prime}$ (step 16). Otherwise, we use the label of the most similar validation sample from Ψ_i (step 18). The main goal of this scheme is to use contextual information also in the switch mechanism to avoid random decisions.

In order to compute the similarity of output profiles to perform step 4 in Algorithm 2, we use the template matching measure. In considering two output profiles, this measure computes how many classifiers will provide exactly the same output. We can implement this measure by maximizing (8), which depends on (9).

$$\delta_{i,j} = \frac{\sum_{k=1}^{N} \alpha_{i,j,k}}{N} \tag{8}$$

$$\alpha_{i,j,k} = \begin{cases} 1, & \text{if } \tilde{x}_{i,test,k} = \tilde{x}_{j,val,k} \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

Algorithm 2 *DSA^c*

1: Compute Val' using transformation T on all samples in Val
2: **for** each data point $x_{i,test}$ in $Test$ **do**
3: Compute $O_i = \{o_1, \ldots, o_N\}$ by considering $C = \{c_1, \ldots, c_N\}$, and use transformation T to compute $\tilde{x}_{i,test}$
4: Find the K $\tilde{x}_{j,val}$ most similar to $\tilde{x}_{i,test}$ and put into Ψ_i
5: $C_i^{*\prime\prime} = \emptyset$
6: **for** each $\tilde{x}_{j,val}$ in Ψ_i **do**
7: **for** each C'_k in $C^{*\prime}$ **do**
8: **if** C'_k correctly recognizes $x_{j,val}$ **then**
9: Insert C'_k into C_i^* (re-insert another instance if C'_k is already in the pool)
10: **end if**
11: **end for**
12: **end for**
13: Compute m_i from O''_i
14: *# Switch mechanism*
15: **if** $m_i > \theta$ **then**
16: d_i = most voted class from O''_i
17: **else**
18: d_i = the label of the most similar $\tilde{x}_{j,val}$ from Ψ_i
19: **end if**
20: **end for**

4.3 DSA^c for incremental learning

One by-product of this approach is the ability to adapt to knowledge acquired over time. Such a task is realized by simply adding more data to Val, and computing the corresponding output profiles for Val'. In this case, we can conduct incremental learning without the need to change the parameters the base classifiers. As a consequence, this system can be used with virtually any type of base classifier.

The computation time of the operational phase of DSA^c, however, depends heavily on the size of Val. Also, the application of this approach in an incremental scenario

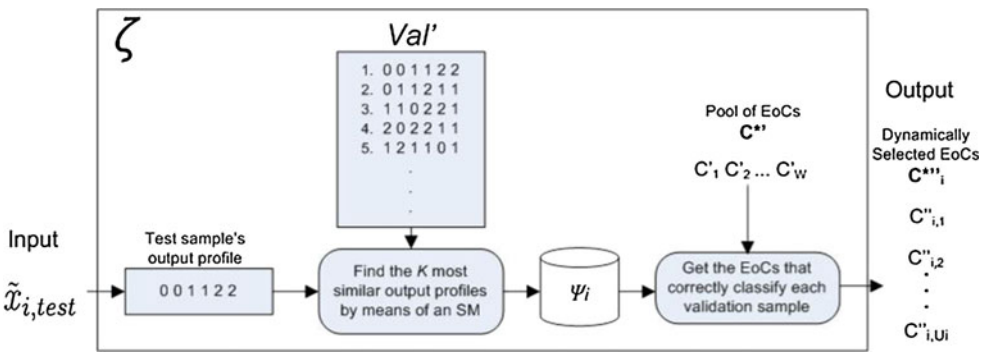

Fig. 5 DSF ζ. For each test sample, we find K validation samples with the most similar output profiles, to form the set ψ_i. The EoCs that correctly classify the validation samples in Ψ_i are used to compose the set $C^{*\prime\prime}$, which is then used to compute the final decision of DSA^c

can slow down very significantly the operational phase since the larger the size of *Val*, the slower is the recognition module. Nevertheless, if we control the inclusion of new samples in *Val* by only injecting those that provide really useful information, we might reduce very significantly the increase of complexity resulting from incremental learning. For this reason, we present a control mechanism to avoid continuously appending new samples to *Val* during incremental learning. This mechanism works as follows.

The control mechanism selects samples, to compose *Val*, only when they are below a threshold ϑ, in considering the margin of the base classifiers, e.g. $m_i < \vartheta$. Note that m_i is defined in Eq. 1. In this case, we suppose that only the samples that possess uncommon output profiles are appended to *Val*, since the contrary is likely to result in the addition of redundant samples. As a consequence, *Val* will only acquire new samples if uncommon samples are observed.

5 Experiments

In this section we present a series of experiments with the following objectives. First, the main goal is to compare the performance of the proposed approaches, i.e. DSA^m and DSA^c, against existing methods. By comparing them against DSA, which provides the baseline architecture for the proposed methods, we aim at observing the impact of the proposed enhancements. By conducting the same comparisons against state-of-the-art static methods, on the other hand, we can observe the advantages of dynamic methods over static ones.

The aforementioned static methods are the followings:

- *All features*: the original classifier with full representation space (all original features);
- *Best from C*: the best base classifier from *C*;
- *MV all C*: fusion of all base classifiers in *C* by majority voting (MV);
- *DT all C*: fusion of all base classifiers in *C* using decision templates (DT), by considering template matching. The decision templates are computed by using *Val*′.
- *Best from* $C^{*\prime}$: the best EoC from $C^{*\prime}$;

All methods are evaluated using seven datasets, divided into two large and five small ones. The small datasets represent problems with a different number of features, generally with a small amount of samples. The datasets considered as small are: the DNA and Satimage datasets provided by Project Stalog on http://www.niaad.liacc.up.pt/old/stalog; Feltwell dataset, which is a multisensor remote-sensing dataset [17]; Ship, which is composed of forward-looking infra-red (FLIR) ship images [18]; and Texture, available in the UCI Machine Learning Repository. These databases, due to their sizes, are divided into ten folds, each time seven folds are used for training, one for optimization, one for validation, and the other one for testing. This process is repeated ten times for each replication, whereas each time a different set of samples was used.

The large datasets represent two handwriting recognition problems, e.g. the recognition of isolated digits and uppercase letters, extracted from the NIST-SD19 database. Two different test sets are used to evaluate digit recognition: *NIST-digits-test1* and *NIST-digits-test2*. For both digits and letters, the original feature set is composed of 132 features, extracted from concavities and contours [19]. Table 1 presents a detailed description of each database.

Given the large amount of training samples available in the *NIST-digits* database (in addition to the training samples described in Table 1, there are 185,000 additional training samples), and the use of a well studied feature set, we can reduce the size of the training set to increase the level of uncertainty of the recognition problem, and

Table 1 Experimental setup

Problem	NC	Train	Opt	Val	Test	NF	NE	VM
DNA	3	2,232	318	318	318	180	45	KF
Feltwell	5	7,662	1,094	1,094	1,094	15	8	KF
Satimage	6	4,506	643	643	643	36	18	KF
Ship	8	1,780	255	255	255	11	6	KF
Texture	11	3,850	550	550	550	40	20	KF
Digits	10	5,000	10,000	10,000	*t1* 60,089 *t2* 58,646	132	32	HO
Letters	26	43,160	3,980	7,960	12,092	132	32	HO

NC number of classes; *Train*, *Opt*, *Val*, and *Test*: number of samples in these respective sets; *NF* number of features, *NE* number of features in the ensemble, after applying the RSS method, *VM* validation method, *KF* k-fold validation, *HO* hold-out validation. Each dataset of the methods using KF had ten different re-samplings, with no overlapping among the sets

simulate different conditions of uncertainty (or confusion, which is a term used interchangeably with uncertainty hereafter). Consequently, this database does not only allow for simulating an incremental learning scenario, but also for evaluating how an approach can behave at different degrees of confusion. For this reason, in this section, we also aim at answering the following questions:

1. How can DSA^c behave in an incremental learning scenario, by just appending new samples to *Val*?
2. How dynamic selection, represented by DSA^c, performs against static selection when the size of *Val* ranges from small (high level of uncertainty) to high (low level of uncertainty)?

For all experiments, the following parameters were considered. For each dataset, 100 base classifiers, with a pre-defined number of features, are generated from the baseline feature set, based on the random subspaces (RSS) ensemble generation method [20]. The base classifiers can be considered weak classifiers in two aspects. First, the two different types of classifiers, e.g. k-nearest neighbors classifiers with $k = 1$ (1NN), and C4.5 decision tree (DTree) classifiers, can be considered very weak for many problems. Second, the reduced number of features used by the RSS method (see Table 1 for the number of features used for each problem) greatly contributes to weaken the performance of the classifiers.

To generate the pool of EoCs, a genetic algorithm (GA) is used, in an off-line fashion, to find an archive with the 25 best solutions on *Val*, representing $C^{*\prime}$, guided by the optimization set *Opt*. The following parameters were used in this work: population size: 128; number of generations: 1,000; probability of crossover: 0.8; probability of mutation: 0.01; one-point crossover and bit-flip mutation [7]. The experiments are replicated 30 times, where in each replication the archive provided by GA is generally different. The results represent the mean error rates over the 30 replications. For each of the sets using k-fold validation, each replication represents the mean over the ten re-samplings of each dataset.

For the large datasets, we also evaluated the method known as Bagging to generate the base classifiers. We used the same scheme employed in [7], where 100 DTree classifiers were generated by dividing the training set into 100 subsets of equal size, where the samples for each set were randomly chosen, with no overlapping among the sets. DTree is used as the base classifier given that Bagging works better with unstable classifiers.

The results are statistically validated by the Kruskal-Wallis nonparametric statistical test. We test the equality among the mean values, using a confidence level of 95%. Dunn-Sidak correction is applied to critical values.

5.1 Results and discussion

The results from the evaluation of small datasets are presented in Tables 2 and 3, for 1NN and DTrees, respectively. Results from the evaluation of large datasets are presented in Table 4. In all tables, we present only the error rates of DSA^c with $K = 30$. The impact of K will described later.

For both DSA^m and DSA^c, $\theta = 0$, since this was the best value after preliminary evaluations as shown in Fig. 6. As demonstrated, the switch works very well as a tie-breaking mechanism for both approaches. It is worth noting that when we increase the value of θ, the final error rates

Table 2 Error rates on small datasets using 1NN classifiers

Method	Dna	Felt	Sat	Ship	Text
Static selection					
Oracle C	0.03 (−)	0.67 (−)	0.36 (−)	0.28 (−)	0.04 (−)
All features	26.30 (−)	12.35 (−)	9.84 (−)	11.24 (−)	1.13 (−)
Best from C	23.10 (−)	9.46 (−)	8.95 (−)	10.26 (−)	0.62 (−)
MV all C	6.87 (−)	10.44 (−)	8.59 (−)	9.94 (−)	1.11 (−)
Best from $C^{*\prime}$	9.14 (1.60)	9.37 (2.09)	8.19 (2.89)	9.41 (1.66)	0.71 (1.74)
DT C	8.53 (−)	14.76 (−)	8.97 (−)	10.03 (−)	4.56 (−)
Dynamic selection					
Oracle $C^{*\prime}$	1.12 (0.86)	6.06 (2.46)	3.98 (0.83)	3.92 (1.40)	0.40 (0.24)
DSA	10.47 (3.17)	10.76 (4.90)	9.17 (0.99)	11.21 (3.48)	1.03 (0.34)
*DSA^m	5.57 (1.33)	9.35 (4.62)	7.61 (0.87)	8.80 (2.30)	0.93 (0.37)
*DSA^c	**5.46** (0.26)	**8.93** (0.30)	**7.42** (0.31)	**8.10** (0.38)	**0.56** (0.10)

Results in bold present the best approach among static MO, DSA, DT, and the proposed DSA^m and DSA^c, with K set to 30. Underlined results represent the statistically-significant best method. Highlighted by * are the proposed approaches. Between parentheses is the standard deviation of each approach ($\times 10^{-2}$)

Table 3 Error rates on small datasets using DTree classifiers

Method	Dna	Felt	Sat	Ship	Text
Static selection					
Oracle C	0.03 (−)	0.60 (−)	0.22 (−)	0.24 (−)	0.02 (−)
All features	6.85 (−)	16.81 (−)	4.17 (−)	10.92 (−)	7.56 (−)
Best from C	11.33 (−)	11.86 (−)	11.83 (−)	10.45 (−)	6.07 (−)
MV all C	5.05 (−)	11.86 (−)	8.64 (−)	6.80 (−)	2.56 (−)
Best from $C^{*'}$	5.71 (1.30)	**10.22** (2.11)	8.35 (1.01)	7.02 (1.59)	2.04 (2.60)
DT C	4.53 (−)	13.93 (−)	8.96 (−)	7.74 (−)	1.34 (−)
Dynamic selection					
Oracle $C^{*'}$	1.07 (0.92)	5.82 (3.94)	3.78 (0.78)	3.18 (1.76)	0.81 (0.24)
DSA	7.55 (2.47)	12.52 (5.28)	10.29 (2.16)	10.16 (4.38)	2.42 (0.82)
*DSA^m	4.07 (1.07)	10.77 (4.82)	7.42 (0.76)	5.89 (1.82)	2.13 (0.78)
*DSA^c	**3.05** (0.34)	10.32 (0.41)	**7.11** (0.30)	**5.52** (0.45)	**1.11** (0.17)

Results in bold present the best approach among static MO, DSA, DT, and the proposed DSA^m and DSA^c, with K set to 30. Underlined results represent the statistically-significant best method. Highlighted by * are the proposed approaches. Between parentheses is the standard deviation of each approach ($\times 10^{-2}$)

Table 4 The same evaluations in error rates as in Table 2, but considering both 1NN and DTrees with large datasets

Classifier method	1NN—RSS			DTree—RSS			DTree—Bagging		
	Digits		Letters	Digits		Letters	Digits		Letters
	Test1	Test2		Test1	Test2		Test1	Test2	
Static selection									
Oracle C	0.05 (−)	0.17 (−)	0.18 (−)	0.01 (−)	0.04 (−)	0.04 (−)	0.24 (−)	0.63 (−)	0.29 (−)
All features	6.66 (−)	9.76 (−)	7.82 (−)	11.07 (−)	18.20 (−)	13.50 (−)	6.66 (−)	9.76 (−)	7.82 (−)
Best from C	7.52 (−)	13.99 (−)	14.47 (−)	10.30 (−)	19.18 (−)	17.13 (−)	9.70 (−)	16.62 (−)	14.31 (−)
MV all C	3.72 (−)	8.10 (−)	6.60 (−)	2.92 (−)	6.67 (−)	6.06 (−)	5.65 (−)	10.99 (−)	7.63 (−)
Best from $C^{*'}$	3.60 (1.95)	7.77 (2.78)	6.56 (2.59)	2.98 (2.23)	6.77 (1.05)	6.21 (2.79)	5.31 (0.06)	10.28 (0.03)	7.62 (0.02)
DT C	2.55 (−)	5.74 (−)	4.95 (−)	2.00 (−)	5.00 (−)	4.64 (−)	3.65 (−)	7.65 (−)	6.49 (−)
Dynamic selection									
Oracle $C^{*'}$	1.97 (0.14)	4.59 (0.37)	3.87 (2.10)	1.87 (1.01)	4.39 (2.08)	4.53 (1.57)	3.72 (0.04)	7.42 (0.02)	4.68 (0.01)
DSA	3.61 (0.28)	7.87 (0.41)	6.43 (0.69)	2.87 (0.24)	6.61 (0.54)	6.06 (0.64)	5.33 (0.05)	10.45 (0.06)	7.11 (0.03)
*DSA^m	3.45 (0.22)	7.53 (0.40)	6.12 (0.66)	2.72 (0.42)	6.26 (0.76)	5.83 (0.61)	5.10 (0.08)	9.96 (0.05)	7.23 (0.04)
*DSA^c	**2.37** (0.14)	**5.34**(0.21)	**4.62** (0.41)	**1.76** (0.14)	**4.36** (0.20)	**4.20** (0.22)	**2.98** (0.04)	**6.17** (0.03)	**5.58** (0.05)

The standard deviation in this case was multiplied by 10^{-3}. In addition, we present the evaluation of DTree classifiers created by bagging

also increase. This fact suggests that by relying more on the decisions provided by the main structure of either DSA^m or DSA^c (note the higher the value of θ, the more often the switch is used), and only using the base classifiers when a tie occurs, the final approach is more reliable.

The error rates resulting from the evaluation of small databases show that both DSA^m and DSA^c are very promising for problems presenting a high level of confusion. The only database for which neither of the proposed methods resulted in the lowest error rates was the Feltwell database, using DTree as base classifiers. On all the other databases, DSA^c achieved the lowest recognition rates.

For the large databases, DSA^c yielded the lowest error rates on all databases. DSA^m, in contrast, has performed poorly compared to other static methods. Note that DSA^c uses the validation dataset to compute the DMO structure, and given the larger amount of training samples compared to the small datasets, we believe that this approach has been able to take better advantage of the lower level of uncertainty of large problems, so that it reaches the best performance in this evaluation. Given the better performance of DSA^c over DSA^m, hereafter, we pursue the experimental evaluation by considering only the former for the sake of simplicity.

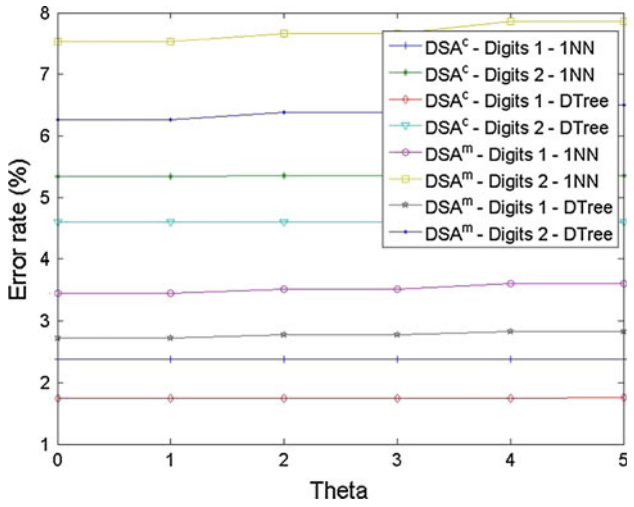

Fig. 6 Evaluation of the parameter θ for the switch mechanism

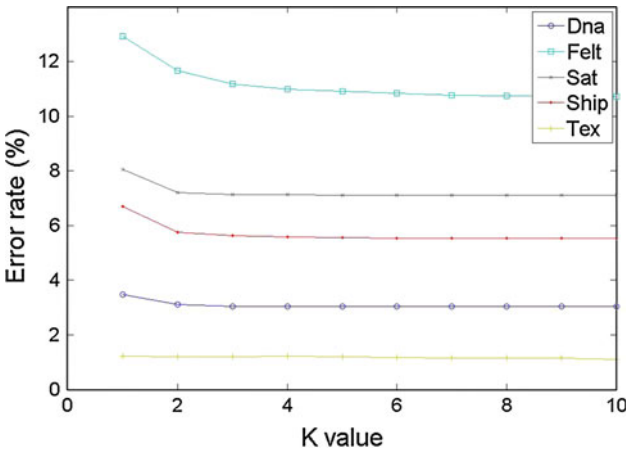

Fig. 8 Evaluation of DSA^c on small datasets with DTree classifiers, K varying from 1 to 10

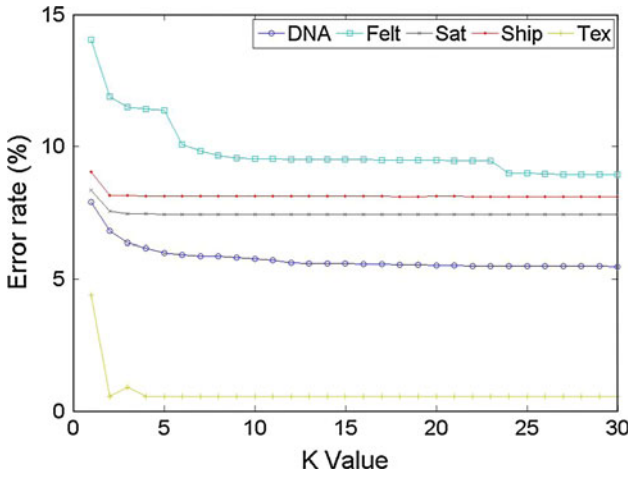

Fig. 7 Evaluation of DSA^c on small datasets with 1NN classifiers, K varying from 1 to 30

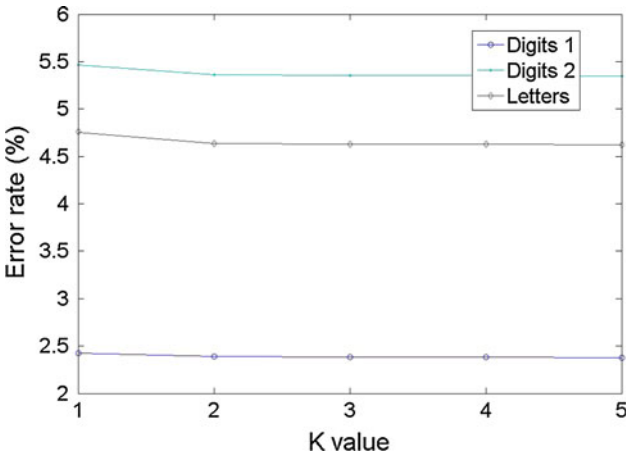

Fig. 9 Evaluation of DSA^c on large datasets with 1NN classifiers, K varying from 1 to 5

In order to provide a broader overview of the performance of DSA^c, we show the impact of the value of K, in a range between 1 and 30. Such an evaluation is presented in Figs. 7 and 8 for small problems, with 1NN and DTrees, respectively. In Figs. 9 and 10, we present the same evaluation in large problems, with 1NN and DTrees, respectively. We observe that the best value for this parameter is problem-dependent. Databases that generate higher error rates, such as Feltwell, require high K values, and databases with very low error rates, such as Texture, require very low K values. Consequently, even though by setting $K = 30$ DSA^c is able to perform well, this value could be adapted to either improve performance or reduce complexity.

Even though the main goal of this paper was to improve the performance of fusion functions, in Table 5 we present a summary of the results presented by DSA^c against the best results reported in the literature for the same databases evaluated in this work. This table can provide us an idea to what level of performance a multiple classifier system, using weak classifiers, can attain by using a very robust combination approach.

In considering small databases, DSA^c has been able to outperform the best results thus far published in the literature, on all databases. It is worth noting that none of the methods presented in Table 5 uses exactly the same experimental protocol, so this comparison is not as accurate as for large databases. However, the use of data from the same database provides a good idea on the difference in performance among the different methods.

For large databases, the error rates presented by DSA^c are slightly higher than the lowest error rates reported in the literature. However, the best results so far have been achieved by using strong classifiers, such as Support Vector Machines (SVM) [22] and Multilayer Perceptron

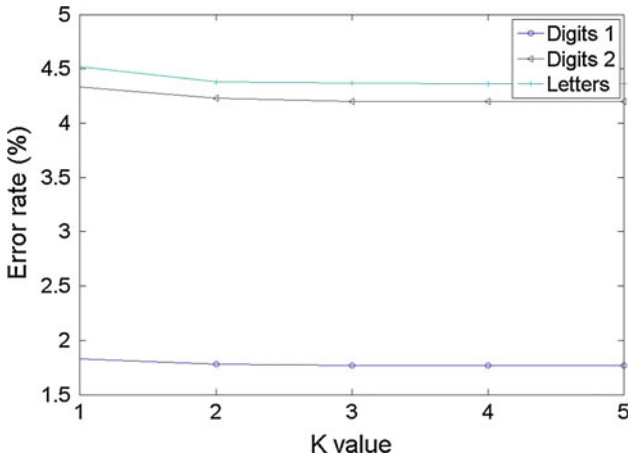

Fig. 10 Evaluation of DSA^c on large datasets with DTree classifiers, K varying from 1 to 5

(MLP) Neural Networks [23], which generally deal very well with large training sets. In this paper, we limited the scope of the work to consider only weak classifiers and small training datasets in order to better observe the behavior of combination approach in conditions that might generate a high level of confusion for the base classifiers. The results from the literature, in contrast, might have dealt with lower levels of confusion due to the much larger amount of samples used for training.

As a consequence, the remainder of this section aims at comparing the performance of DSA^c against MLP and SVM, which are state-of-the-art static approaches, at the same conditions. First, we evaluate what level of performance DSA^c can reach if we incrementally learn the information provided by the remaining training samples in the *NIST-digits* database. Next, we retrain MLPs and SVMs at different levels of uncertainty, which are achieved by downsizing the *NIST-digits* database, and compare their results against the ones produced by DSA^c.

5.1.1 *Evaluation of DSA^c in an incremental learning scenario*

In this section, we evaluate the impact of increasing the size of *Val* to improve the overall performance of DSA^c, by simulating an incremental scenario. Such a simulation consists of gradually adding new samples to *Val*, as previously discussed in Sect. 4.2. We take advantage of the large set of digits available in the NIST SD19 database, by increasing the size of *Val* from 10,000 to 180,000 samples. Those are the remaining samples in the hsf_{1-3} series of the database.

The results of these experiments are shown in Fig. 11, considering both 1NN and DTrees with RSS, and both *NIST-digits-test1* and *NIST-digits-test2*. Note that these evaluations do not only aim at evaluating the behavior of the approach in the incremental scenario, but also aim at comparing the final results against the literature, since the best results thus far consider methods that used all samples from this database. As a consequence, in the following paragraphs we discuss the first topic, while the second topic is discussed afterwards.

Generally, the impact of the size of *Val* is more significant when the size of *Val* is relatively small, and it tends to gradually converge with larger validation sets. Nevertheless, with any increase in *Val* we can observe some improvement. This fact shows that the approach can incrementally acquire knowledge by only increasing the size of this set, so that it can be a generic approach for incremental learning. This allows us to use a heterogeneous pool of classifiers in the incremental learning process.

Figure 12 plots the results of the evaluation of different values for ϑ using the control mechanism described in Sect. 4.2. Compared to the performance of the system using all 180,000, we see that the control mechanism is able not only to maintain the performance of the system, but also to reduce the final error rates. With $\vartheta = 40$, the final error rates are reduced to about 1.1%. In addition, we

Table 5 Error analysis, in which we compare the results of the proposed method DSA^c with the best results published in the literature

Database	Proposed method		Literature	
	Average (variance)	Best result	Method	Result
DNA	3.05 (0.12)	2.88	EoC+DS [7]	4.59
Feltwell	8.85 (0.12)	8.72	EoC+DS [7]	11.50
Satimage	6.89 (0.11)	6.78	EoC+DS [7]	8.64
Ship	5.51 (0.27)	5.32	EoC [21]	5.68
Texture	0.56 (0.01)	0.52	EoC+DS [8]	0.66
NIST-digits-test1	1.76 (0.02)	1.08	Single classifier [22]	0.63
NIST-digits-test2	3.31 (0.04)	3.28	EoC+SS [23]	2.33
NIST-letters	3.89 (0.06)	3.87	SC [22]	3.18

The second column represents the average over 30 replications

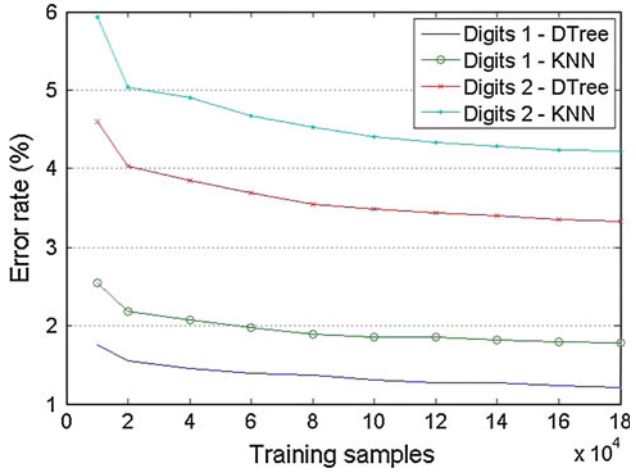

Fig. 11 Incremental evaluation of DSA^c, with $K = 30$, using validation set sizes from 10,000 to 180,000, on both *NIST-digits-test1* and *NIST-digits-test2*

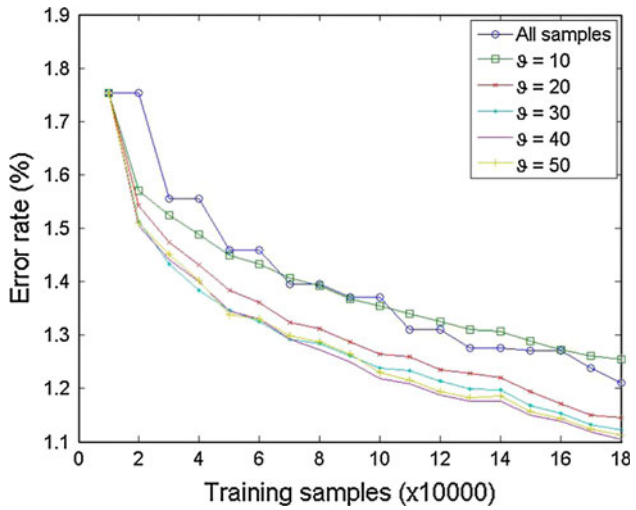

Fig. 12 Incremental evaluation of DSA^c ($K = 30$) with DTree classifiers on *NIST-digits-test1* using a control mechanism

demonstrate in Fig. 13 this mechanism on the size of *Val*. The best approach, represented by $\vartheta = 40$, used only 25,948 samples in *Val*. Comparing with the use of all 180,000 samples, we can reach better results by using only around 15% of this set and drastically reduce the search space of DSA^c for recognition.

The final results can be summarized as follows. With 1NN, the error rates have been reduced from about 2.55% to about 1.78% on *NIST-digits-test1*, and from about 5.9% to about 4.2% on *NIST-digits-test2*. With DTrees, the error rates decreased from about 1.75% to about 1.1% on *NIST-digits-test1*, and from about 4.6% to about 3.31% on *NIST-digits-test2*. Note that on *NIST-digits-test1*, the best results reported in the literature are around 0.63% [22], using 132

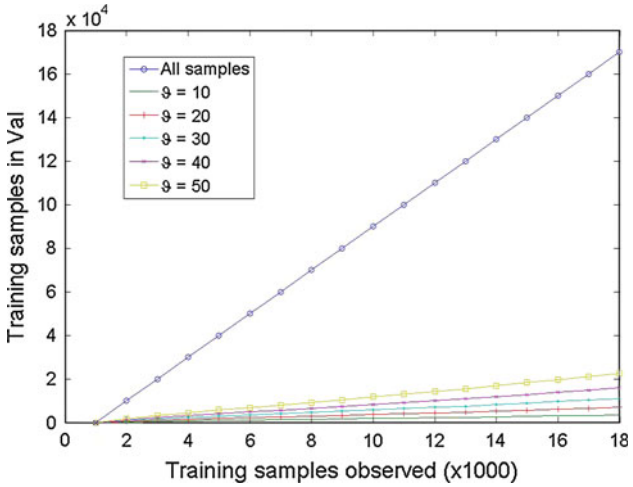

Fig. 13 Size of the validation set for the evaluation presented in Fig. 12

features, 195,000 samples for training, and MLP as classifier. In this work we could get very close (only 0.47% below) to these results by using weak classifiers, trained with only 10,000 samples, of which the range of individual error rates is, for example with 1NN, between 15.92 and 7.53%. Even though in the end we have used the same number of samples to get these results, we have shown that our approach is able to improve weak classifiers to a level which is comparable to the best classification methods in the literature, without changing their parameters.

5.1.2 *Evaluation of DSA^c against MLP and SVM at varied conditions*

As demonstrated in the previous section, by using all the training samples provided by *NIST-digits*, DSA^c can attain a level of performance that is close to state-of-the-art classifiers such as MLP and SVM, consisting of static approaches. However, the higher complexity of DSA^c, in both the design and operational phases, might be a barrier for its application in the real world. For this reason, the main goal of this section is to compare the proposed method, which is a dynamic approach, against MLP and SVM, which are static approaches, under various conditions created by downsizing the *NIST-digits* database. The idea is to observe under which condition dynamic selection might be worth the higher complexity. As we previously mentioned, such a downsizing allows for increasing the level of uncertainty of the problem by simply reducing its training set, since the empirical lower-bound of the *NIST-digits* database is known.

By using a setup similar to that described in the previous section, the training database was reduced to these sizes: 5,000, 10,000, 15,000, 20,000, and 25,000. However, for

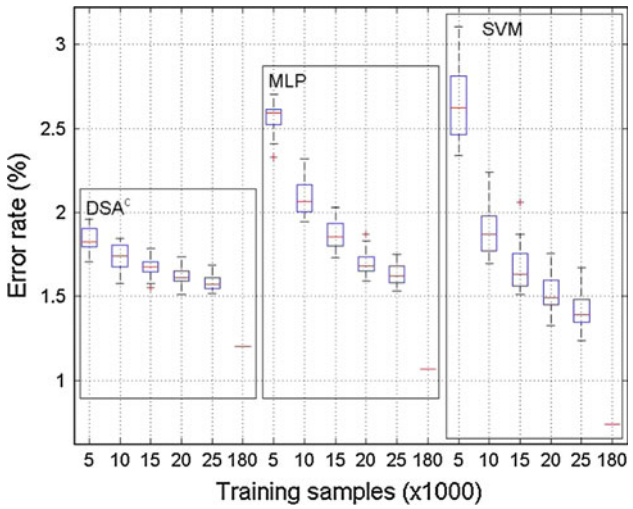

Fig. 14 Evaluation of different sizes of the training set for NIST-digits, using *NIST-digits-test1*. These experiments were replicated 15 times by resampling the training set each time (a single replication for 180,000 samples, which corresponds to the entire dataset). Note that the experiments are grouped by approach, e.g. DSA^c, MLP, and SVM, respectively, and for each approach, we evaluated training sets with 5,000, 10,000, 15,000, 20,000, 25,000, and 180,000 samples, respectively

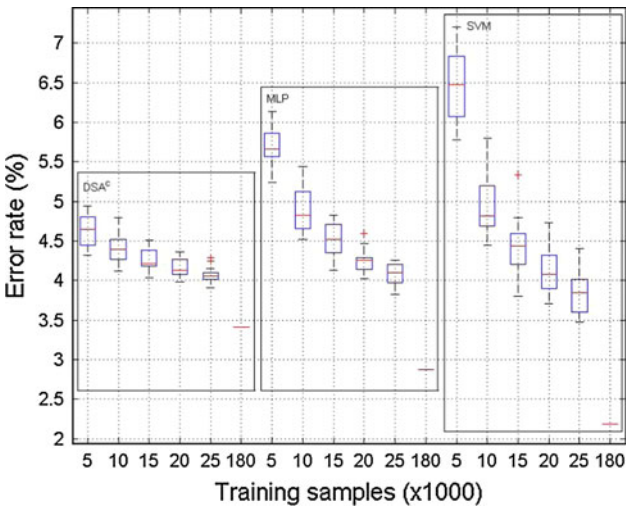

Fig. 15 The same evaluations as in Fig. 14, but using *NIST-digits-test2*

each training set, we did 15 different resamplings so that we could conduct 15 replications for each size of the training set. The parameters for both SVM and MLP were set to the same as reported in [24], which were found as the best parameters for this database. Note that for DSA^c we conduct the incremental learning of *Val*. For MLP and SVM, in contrast, batch learning is considered, since for each training set size, we retrain the classifiers. In addition, it is worth noting that *Val* and *Opt* are merged together to define a single set of samples, which is used as hold-out validation set by MLP and SVM.

The main results are presented in Fig. 14, for *NIST-digits-test1*, and Fig. 15 *NIST-digits-test2*. The most remarkable observation lies in the experiments using only 5,000 samples for training. In this case, DSA^c was significantly superior to both MLP and SVM, showing that the proposed approach can deal better with a high level of uncertainty under these conditions. However, this gap becomes narrower and narrower as we increase the size of the training set, e.g. when we decrease the level of confusion. As a result, the main observation from these experiments is that dynamic selection, despite generally presenting higher complexity than static selection, may be the most recommended approach to attain high performance when the level of confusion of the recognition problem is high. When the level of confusion is low, on the other hand, a static approach may work very well without all the complexity brought by dynamic selection.

6 Conclusion and future work

In this paper we first proposed dynamic multistage organizations to enhance classifier fusion. Based on Dos Santos et al.'s approach (DSA), we first implemented DSA^m to validate these concepts by using multiple dynamic selection functions. Next, we extended DSA^m to use the knowledge provided by the output profiles of validation samples to create DMO, resulting in DSA^c.

Experiments conducted on both small and large databases have confirmed that the proposed DMO concept looks really promising in improving the use of multiple classifiers, since the proposed enhancements have been effective in improving DSA. We also observed a significant improvement in performance of DSA^c over DSA^m, due to the use of contextual information. The use of simulated incremental learning scenario showed that we can improve the performance of DSA^c by only increasing the size of the validation set, without changing the parameters of the base classifiers. Although other classification approaches such as SVMs and MLPs can present better performances than DSA^c when large training sets are available, we demonstrated that the proposed approach results in better performance when one can use only small training databases, e.g. when the level of confusion for recognition is high.

As future work, many directions can be followed. The most important, in our opinion, is to better investigate the observation that DSA^c is better suited to problems presenting a high level of uncertainty. We can evaluate, for example, the current system on other recognition problems. We can, as well, implement the system with other base classifiers and different methods to generate the pool of base classifiers, to evaluate whether the system maintains the same behavior with a different baseline architecture or

not. In addition, reducing the complexity of DSA^c is a key point to better justify its deployment in real-life systems. In this work we simply performed a flat search on *Val*, but other more time-efficient methods can be investigated, for instance some ideas proposed to reduce the complexity of 1NN classifiers [25] to conduct the search for the most similar samples.

Acknowledgments The authors would like to acknowledge the CAPES-Brazil and NSERC-Canada for the financial support.

References

1. Brown G, Wyatt J, Harris R, Yao X (2005) Diversity creation methods: a survey and categorization. Inf Fusion 6(1):5–20
2. Dos Santos EM, Sabourin R, Maupin P (2006) Single and multi-objective genetic algorithms for the selection of ensemble of classifiers. In: Proceedings of international joint conference on neural networks, 2006, Vancouver, Canada, pp 3070–3077
3. Shipp CA, Kuncheva LI (2002) Relationships between combination methods and measures of diversity in combining classifiers. Inf Fusion 3(2):1351–48
4. Kuncheva LI, Whitaker CJ, Shipp C, Duin R (2003) Limits on the majority vote accuracy in classifier fusion. Pattern Anal Appl 6(1):22–31
5. Ruta D, Gabrys B (2002) A theoretical analysis of the limits of majority voting errors for multiple classifier systems. Pattern Anal Appl 5:333–350
6. Ruta D, Gabrys B (2005) Classifier selection for majority voting. Inf Fusion 1:63–81
7. Dos Santos EM, Sabourin R, Maupin P (2008) A dynamic overproduce-and-choose strategy for the selection of classifier ensembles. Pattern Recognit 41:2993–3009
8. Woods K, Kegelmeyer JWP, Bowyer K (1997) Combination of multiple classifiers using local accuracy estimates. IEEE Trans Pattern Anal Mach Intell 19(4):405–410
9. Giacinto G, Roli F (2001) Dynamic classifier selection based on multiple classifier behaviour. Pattern Recognit 34:1879–1881
10. Zhu X, Wu X, Yang Y (2004) Dynamic classifier selection for effective mining from noisy data streams. In: Proceedings of the 4th IEEE international conference on data mining, IEEE Computer Society, Washington, DC, USA, 2004, pp 305–312
11. Soares RGF, Santana A, Canuto AMP, de Souto MCP (2006) Using accuracy and diversity to select classifiers to build ensembles. In: Proceedings of the 2006 international joint conference on neural networks, Vancouver, Canada, 2006, pp 1310–1316
12. Ko AH, Sabourin R, Britto J (2008) From dynamic classifier selection to dynamic ensemble selection. Pattern Recognit 41(5):1718–1731
13. Kuncheva LI (2000) Cluster-and-selection model for classifier combination. In: Proceedings of international conference on knowledge based intelligent engineering systems and allied technologies, Brighton, UK, 2000, pp 185–188
14. Singh S, Singh M (2005) A dynamic classifier selection and combination approach to image region labelling. Signal Process Image Commun 20(3):219–231
15. Hansen LK, Liisberg C, Salamon P (1997) The error-reject tradeoff. Open Syst Inf Dyn 4(2):159–184
16. Kuncheva LI, Bezder JC, Duin RPW (2001) Decision templates for multiple classifier fusion: an experimental comparison. Pattern Recognit 34:299–314
17. Serpico SB, Bruzzone L, Roli F (1996) An experimental comparison of neural and statistical non-parametric algorithms for supervised classification of remote-sensing images. Pattern Recognit Lett 17(3):1331–1341
18. Park Y, Sklansky J (1990) Automated design of linear tree classifiers. Pattern Recognit 23(12):1393–1412
19. Oliveira LES, Sabourin R, Bortolozzi F, Suen CY (2002) Automatic recognition of handwritten numeral strings: a recognition and verification strategy. IEEE Trans Pattern Anal Mach Intell 24(11):1438–1454
20. Ho T (1998) The random subspace method for construction decision forests. IEEE Trans Pattern Anal Mach Intell 20:832–844
21. Rheaume F, Jousselme A-L, Grenier D, Bosse E, Valin P (2002) New initial basic probability assignments for multiple classifiers. In: Kadar I (eds) Society of photo-optical instrumentation engineers (SPIE) conference series, vol 4729, pp 319–328
22. Milgram J, Cheriet M, Sabourin R (2006) One against one" or "One against all": which one is better for handwriting recognition with SVMs? In: Proceedings of 10th international workshop on frontiers in handwriting recognition, La Baule, France, 2006
23. Radtke P (2006) Classification systems optimization with multi-objective evolutionary algorithms, Ph.D. thesis, École de Technologie Supérieure (ETS), Montreal, Canada
24. Milgram J (2007) Contribution à l'intégration des machines à vecteurs de support au sein de systèmes de reconnaissance de formes: application à la lecture automatique de l'écriture manuscrite (in french), Ph.D. thesis, École de Technologie Supérieure
25. Cui B, Ooi BC, Su J, Tan K-L (2003) Contorting high dimensional data for efficient main memory knn processing. In: Proceedings of the 2003 ACM SIGMOD international conference on management of data, San Diego, USA, 2003, pp 479–490