Information Fusion 10 (2009) 150–162

Contents lists available at ScienceDirect

Information Fusion

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

Overfitting cautious selection of classifier ensembles with genetic algorithms

Eulanda M. Dos Santos *, Robert Sabourin, Patrick Maupin

Ecole de Technologie Superieure – ETS, Genie de la production automatisee, 1100, Rue Notre-Dame Ouest, Montreal, Quebec, Canada H3C1K3

A R T I C L E I N F O

Article history:
Received 19 April 2006
Received in revised form 3 November 2008
Accepted 6 November 2008
Available online 27 November 2008

Keywords:
Classifier ensemble selection
Overfitting
Genetic algorithms
Diversity

A B S T R A C T

Information fusion research has recently focused on the characteristics of the decision profiles of ensemble members in order to optimize performance. These characteristics are particularly important in the selection of ensemble members. However, even though the control of overfitting is a challenge in machine learning problems, much less work has been devoted to the control of overfitting in selection tasks. The objectives of this paper are: (1) to show that overfitting can be detected at the selection stage; and (2) to present strategies to control overfitting. Decision trees and k nearest neighbors classifiers are used to create homogeneous ensembles, while single- and multi-objective genetic algorithms are employed as search algorithms at the selection stage. In this study, we use bagging and random subspace methods for ensemble generation. The classification error rate and a set of diversity measures are applied as search criteria. We show experimentally that the selection of classifier ensembles conducted by genetic algorithms is prone to overfitting, especially in the multi-objective case. In this study, the partial validation, backwarding and global validation strategies are tailored for classifier ensemble selection problem and compared. This comparison allows us to show that a global validation strategy should be applied to control overfitting in pattern recognition systems involving an ensemble member selection task. Furthermore, this study has helped us to establish that the global validation strategy can be used to measure the relationship between diversity and classification performance when diversity measures are employed as single-objective functions.

1. Introduction

The application of an ensemble creation method such as bagging [4], boosting [6] or random subspace [9] generates an initial set of classifiers $\mathcal{C}$, where $\mathcal{C} = \{c_1, c_2, \ldots, c_n\}$. Given such a pool of classifiers, there are two main options for designing ensembles: (1) fusion; and (2) selection. The most common and most general operation is the fusion of all n classifier decisions. Majority voting, sum, product, maximum and minimum [11] are examples of functions used to combine the decisions of ensemble members. Classifier decision fusion is based on a hope that each classifier makes independent errors. However, it is difficult to impose independence among an ensemble's component members, especially since the component classifiers are redundant [31], i.e. they provide responses to the same problem. As a consequence, there is no guarantee that a particular ensemble combination method will achieve error independence.

By contrast, classifier selection has focused on finding the most efficient subset of classifiers, rather than combining all available n classifiers. This approach is mentioned under different names in the literature, such as "overproduce and choose strategy" [19] and "test and select methodology" [31]. In any case, the principle is that, given the initial pool $\mathcal{C}$, the best performing subset of classifiers in $\mathcal{P}(\mathcal{C})$ must be found, and this is the powerset of $\mathcal{C}$ defining the population of all possible candidate ensembles C_j. Although the search for the best subset of classifiers can be exhaustive [31], search algorithms might be used when a large $\mathcal{C}$ is available due to the exponential computing complexity of an exhaustive search, since the size of $\mathcal{P}(\mathcal{C})$ is 2^n.

Several search algorithms have been applied in the literature for classifier selection, ranging from ranking the n best classifiers [19] to genetic algorithms (GAs) [28,33]. Ensemble combination performance [27], diversity measures [33,1,30] and ensemble size [23] are search criteria which are often employed. GAs are attractive since they allow the fairly easy implementation of classifier selection tasks as optimization processes [32]. However, it has been shown that such stochastic search algorithms, when used in conjunction with machine learning techniques, are prone to overfitting in different application problems like distribution estimation algorithms [36], the design of an evolutionary multi-objective learning system [14], multi-objective pattern classification [2] and wrapper-based feature subset selection [15,7].

Overfitting is a key problem in supervised machine learning tasks. It is the phenomenon detected when a learning algorithm fits the training set so well that noise and the peculiarities of the training data are memorized. As a result of this, the learning algorithm's performance drops when it is tested in an unknown dataset. The

* Corresponding author. Tel.: +1 4504627600.
E-mail address: eulanda@livia.etsmtl.ca (E.M. Dos Santos).

1566-2535/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.inffus.2008.11.003

amount of data used for the learning process is fundamental in this context. Small datasets are more prone to overfitting than large datasets [12], although, due to the complexity of some learning problems, even large datasets can be affected by overfitting. In an attempt to tackle this issue, several machine learning studies have proposed solutions, such as: regularization methods, adding noise to the training set, cross-validation and early stopping [25]. Early stopping is the most common solution for overfitting control.

The problem of selecting classifier ensembles, using an optimization dataset $\mathcal{O}$ can be formulated as a learning task, since the search algorithm operates by minimizing/maximizing the objective function. We may define overfitting in the context of ensemble selection inspired by the definition provided by Mitchell [16] in the following way. Let C_j^* and $C_j^{*'}$ be the best performing candidate ensembles found through calculating the error rate ϵ for each element of $\mathcal{P}(\mathcal{C})$ over samples contained in $\mathcal{O}$ and in a validation set $\mathcal{V}$ respectively. Consider the classification error ϵ of these two candidate ensembles measured using samples from $\mathcal{V}$. We will denote this classification error by $\epsilon(\mathcal{V}, C_j^*)$ and $\epsilon(\mathcal{V}, C_j^{*'})$. In this setting, C_j^* is said to overfit on $\mathcal{O}$ if an alternative candidate ensemble $C_j^{*'} \in \mathcal{P}(\mathcal{C})$ can be found such that $\epsilon(\mathcal{V}, C_j^*) > \epsilon(\mathcal{V}, C_j^{*'})$. In this way, overfitting is measured as

$$\text{overfitting} = \epsilon(\mathcal{V}, C_j^*) - \epsilon(\mathcal{V}, C_j^{*'}). \tag{1}$$

Even though different aspects have been addressed in studies investigating overfitting in the context of classifier ensembles, for instance regularization terms [24] and methods for tuning classifier members [20], very few authors have focused on proposing methods to tackle overfitting at the selection stage. Tsymbal et al. [34] suggested that using individual member accuracy (instead of ensemble accuracy) together with diversity in a genetic search can overcome overfitting. Radtke et al. [23] proposed a global validation method for multi-objective evolutionary optimization including classifier ensemble selection.

Nonetheless, overfitting in pattern recognition has attracted considerable attention in other optimization applications. Llorà et al. [14] suggested the use of optimization constraints to remove the overfitted solutions over the Pareto front in an evolutionary multi-objective learning system. Wiegand et al. [35] [23] proposed global validation strategies for the evolutionary optimization of neural network parameters. Loughrey and Cunningham [15] presented an early-stopping criterion to control overfitting in wrapper-based feature subset selection using stochastic search algorithms such as GA and Simulated Annealing. Finally, Robilliard and Fonlupt [26] proposed "backwarding", a method for preventing overfitting in Genetic Programming (GP).

Overfitting control methods applied to pattern recognition problems may be divided into three categories. The first contains *problem-dependent* methods, which, as the name suggests, cannot be widely reused [14,34]. The second contains the *early stopping-based* methods [15], which can be directly used in single-objective optimization problems. The method proposed in [15], for example, relies on determining a stopping generation number by averaging the best solution from each generation over a set of 10-fold cross-validation trials. However, defining an early-stopping criterion is more difficult when a Pareto front (i.e. a set of nondominated solutions) is involved due to the fact that comparing sets of equal importance is a very complex task. Finally, the third contains the *archive-based* methods [23,35,26], which have been shown to be efficient tools for tackling overfitting [35,26,29] and may be widely reused in all optimization problems [23].

We show in this paper that overfitting can be detected during the process of classifier ensemble selection, this process being formulated as an optimization problem using population-based evolutionary algorithms. We attempt to prove experimentally that an overfitting control strategy must be conducted *during* the optimization process. Since both single-objective GA and multi-objective GA (MOGA) are examined in this paper, we investigate the use of an auxiliary archive $\mathcal{A}$ to store the best performing candidate ensembles (or Pareto fronts in the MOGA case) obtained in a validation process using the validation partition $\mathcal{V}$ to control overfitting. Three different strategies for update $\mathcal{A}$ have been compared and adapted in this paper to the context of the single- and multi-objective selection of classifier ensembles: (1) *partial validation* where $\mathcal{A}$ is updated only in the last generation of the optimization process; (2) *backwarding* [26] which relies on monitoring the optimization process by updating $\mathcal{A}$ with the best solution from each generation; and (3) *global validation* [23] updating $\mathcal{A}$ by storing in it the Pareto front (or the best solution in the GA case) identified on $\mathcal{V}$ at each generation step.

The initial pools of classifiers are created using bagging and the random subspace method while decision trees (DT) and k nearest neighbors (kNN) are the base classifiers. Four diversity measures (described in Section 4.1.3) and the classification error rate ϵ are used to guide the optimization process. Diversity measures are applied by MOGA in combination with ϵ in pairs of objective functions. Moreover, the diversity measures, as well as ϵ, are employed as single-objective functions by GA. The global validation strategy is also used to show the relationship between diversity and performance, specifically when diversity measures are used to guide GA. The assumption is that, if a strong relationship exists between diversity and performance, the solution obtained by performing global validation solely guided by diversity should be close, or equal, to the solution with highest performance among all solutions evaluated. This offers a new possibility for analyzing the relationship between diversity and performance, which has received a great deal of attention in the literature [13,5,27].

Our objective in this paper is to answer the following questions:

(1) Which is the best strategy employing an archive $\mathcal{A}$ for reducing overfitting in classifier ensemble selection when this selection is formulated as an optimization problem?
(2) Are classifier ensembles generated by bagging and the random subspace method equally affected by overfitting in classifier ensemble selection?

The following section demonstrates the circumstances under which the process of classifier ensemble selection results in overfitting. In Section 3, three strategies used to control overfitting are introduced. The parameters employed in the experiments are described in Section 4. Finally, the experimental results are presented in Section 5. Conclusions and suggestions for future work are discussed in Section 6.

2. Overfitting in selecting classifier ensembles

The selection process for classifier ensembles is illustrated in Fig. 1. An ensemble creation method is employed using a training dataset ($\mathcal{T}$) to generate the initial pool of classifiers $\mathcal{C}$. Thus, the search algorithm calculates fitness on $\mathcal{O}$ by testing different candidate ensembles. The best candidate ensemble ($C_j^{*'}$) is identified in $\mathcal{V}$ to prevent overfitting. Finally, the generalization performance of $C_j^{*'}$ is measured using a test dataset ($\mathcal{G}$), ω_k representing a class label. Hence, the selection stage requires at least these four datasets. However, when it is necessary to adjust base classifier parameters, such as weights in multilayer perceptron (MLP), the number of neighbors considered (k value) in kNN classifiers, etc, it may be better to use a fifth dataset, as was done in [23], than to use the same $\mathcal{V}$ for both validation tasks, control of overfitting in optimization process and classifier parameter adjustment. The following

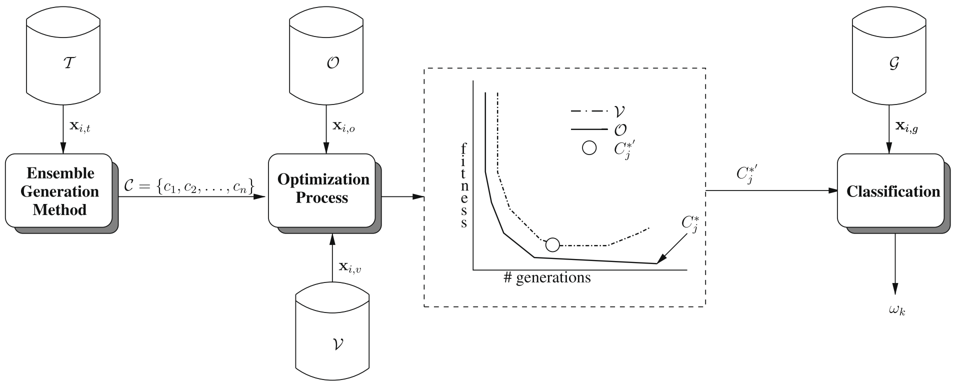


Fig. 1. Overview of the process of selection of classifier ensembles and the points of entry of the four datasets used.

sections describe how overfitting can be detected in single- and multi-objective optimization problems performed by GAs.

2.1. Overfitting in single-objective GA

Traditionally, in single-objective problems, the search algorithm is guided by an objective function during a fixed maximum number of generations ($max(g)$). In Fig. 2, ϵ is employed as the objective function to guide GA using the NIST-digits database (Section 4.1.1). The random subspace (RSS) method was used to generate $\mathscr{C}$ as a pool of 100 kNN classifiers (see Section 4.1.2 for further details). Although GA was only guided by ϵ, we show plots of ϵ versus the ensemble's size to better illustrate the process.

We denote $\mathbf{C}(g)$ as the population of candidate classifier ensembles found at each generation g. The best candidate ensemble found by evaluating all individuals from $\mathbf{C}(g)$ on $\mathscr{O}$ is represented by $C_j^*(g)$, while each point on the plot corresponds to a candidate ensemble C_j taken from $\mathscr{P}(\mathscr{C})$ and evaluated during the optimization process. Indeed, these points represent the complete search space explored for $max(g) = 1000$. The number of individuals at any $\mathbf{C}(g)$ is 128.

Actually, the overfitting phenomenon measured when selecting classifier ensembles presents a behavior very similar to what is seen in a typical Machine Learning process, i.e. the chance that the search algorithm will overfit on samples taken from $\mathscr{O}$ increases with the number of generations. This phenomenon can be observed in Fig. 2(e), which shows the evolution of $\epsilon(\mathscr{O}, C_j^{*'})$ in the search space. On the one hand, it is clear that the optimization process could be stopped before overfitting starts to occur in this problem using GA. On the other hand, it is difficult to define a stopping criterion to control overfitting in a multi-objective optimization process, as explained in Section 2.2.

2.2. Overfitting in MOGA

Pareto-based evolutionary algorithms often constitute solutions to optimization processes guided by multi-objective functions, such as the combination of ϵ and diversity measures. These algorithms use Pareto dominance to reproduce the individuals. A Pareto front is a set of nondominated solutions representing different tradeoffs between the multi-objective functions. In our classifier ensemble selection application, a candidate ensemble solution C_i is said to dominate solution C_j, denoted $C_i \preceq C_j$, if C_i is no worse than C_j on all the objective functions and C_i is better than C_j in at least one objective function. Based on this nondominance criteria, solutions over the Pareto front are considered to be equally important.

Among several Pareto-based evolutionary algorithms proposed in the literature, NSGA-II (elitist non-dominated sorting genetic algorithm) [10] appears to be interesting because it has two important characteristics: a full elite-preservation strategy and a diversity-preserving mechanism using the crowding distance as the distance measure. The crowding distance does not need any parameter to be set [10]. Elitism is used to provide the means to keep good solutions among generations, and the diversity-preserving mechanism is used to allow a better spread among the solutions over the Pareto front.

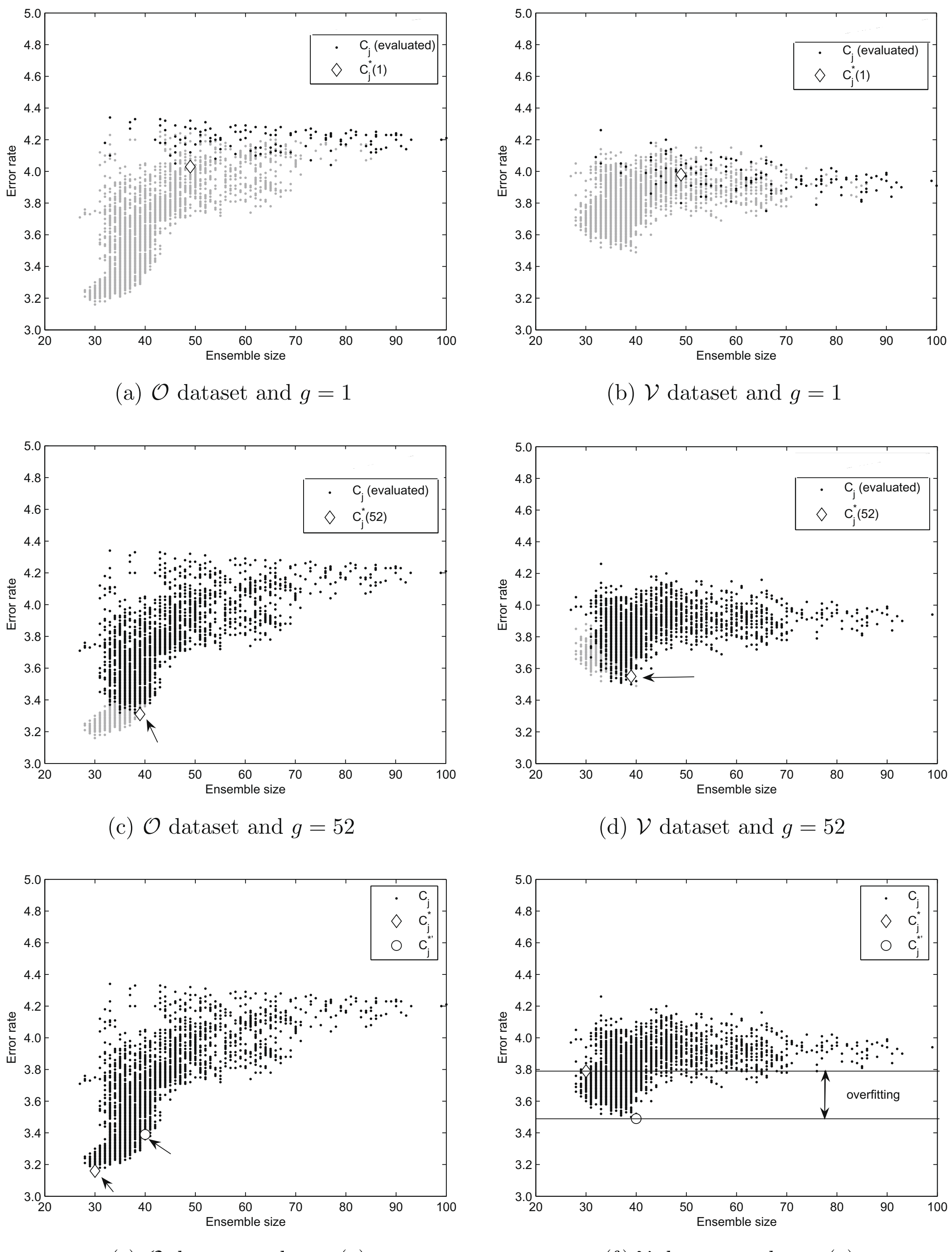
NSGA-II [10] works as follows. At each generation step g, a parent population $\mathbf{C}(g)$ of size w evolves and an offspring population $\mathbf{C}^q(g)$, also of size w, is created. These two populations are combined to create a third population $\mathbf{C}^r(g)$ of size $2w$. The population $\mathbf{C}^r(g)$ is sorted according to the nondominance criteria, and different non-dominated fronts are obtained. Then, the new population $\mathbf{C}(g+1)$ is filled by the fronts according to the Pareto ranking. In this way, the worst fronts are discarded, since the size of $\mathbf{C}(g+1)$ is w. When the last front allowed to be included in $\mathbf{C}(g+1)$ has more solutions than the $\mathbf{C}(g+1)$ available free space, the crowding distance is measured in order to select the most isolated solutions in the objective space to increase diversity. Algorithm 1 summarizes NSGA-II.

Algorithm 1. NSGA-II

```
1: Creates initial population $\mathbf{C}(1)$ of $w$ chromosomes
2: while $g < max(g)$
3:    create $\mathbf{C}^q(g)$
4:    set $\mathbf{C}^r(g) = \mathbf{C}(g) \cup \mathbf{C}^q(g)$
5:    perform a non-dominated sorting to $\mathbf{C}^r(g)$ and identify different fronts $\mathbf{C}_k$, $k = 1, 2, \ldots, etc.$
6:    while $|\mathbf{C}(g+1)| + |\mathbf{C}_k| \leqslant w$
7:       set $\mathbf{C}(g+1) := \mathbf{C}(g+1) \cup \mathbf{C}_k$
8:       set $k := k+1$
9:    endwhile
10:   perform crowding distance sort to $\mathbf{C}_k$
11:   set $\mathbf{C}(g+1) := \mathbf{C}(g+1) \cup \mathbf{C}_k\lfloor 1 : (w - |\mathbf{C}(g+1)|)\rfloor$
12:   create $\mathbf{C}^q(g+1)$ from $\mathbf{C}(g+1)$
13:   set $g := g+1$
14: endwhile
```

Fig. 3(left) illustrates the optimization process performed for $max(g) = 1000$ using NSGA-II guided by the following pair of objective functions: jointly minimize ϵ and difficulty measure (Section 4.1.3) in the same problem investigated with GA in last section. The Pareto front from population $\mathbf{C}(g)$ found on $\mathscr{O}$ is denoted by $\mathbf{C}_k(g)$. Thus, $\mathbf{C}_k^*$ (diamonds) and $\mathbf{C}_k^{*'}$ (circles) represent the final Pareto fronts found on $\mathscr{O}$ and $\mathscr{V}$, respectively.

Especially noteworthy in Fig. 3(f) is that, besides the fact that the solutions over $\mathbf{C}_k^{*'}$ are different from solutions over $\mathbf{C}_k^*$, which was expected considering that $\mathscr{V}$ and $\mathscr{O}$ are different datasets, the non-dominated solutions over $\mathbf{C}_k^{*'}$ are discarded during the optimization process (Fig. 3(e)). Hence, the definition of a stopping criterion for multi-objective optimization problems is difficult for the



(a) $\mathcal{O}$ dataset and $g = 1$ (b) $\mathcal{V}$ dataset and $g = 1$

(c) $\mathcal{O}$ dataset and $g = 52$ (d) $\mathcal{V}$ dataset and $g = 52$

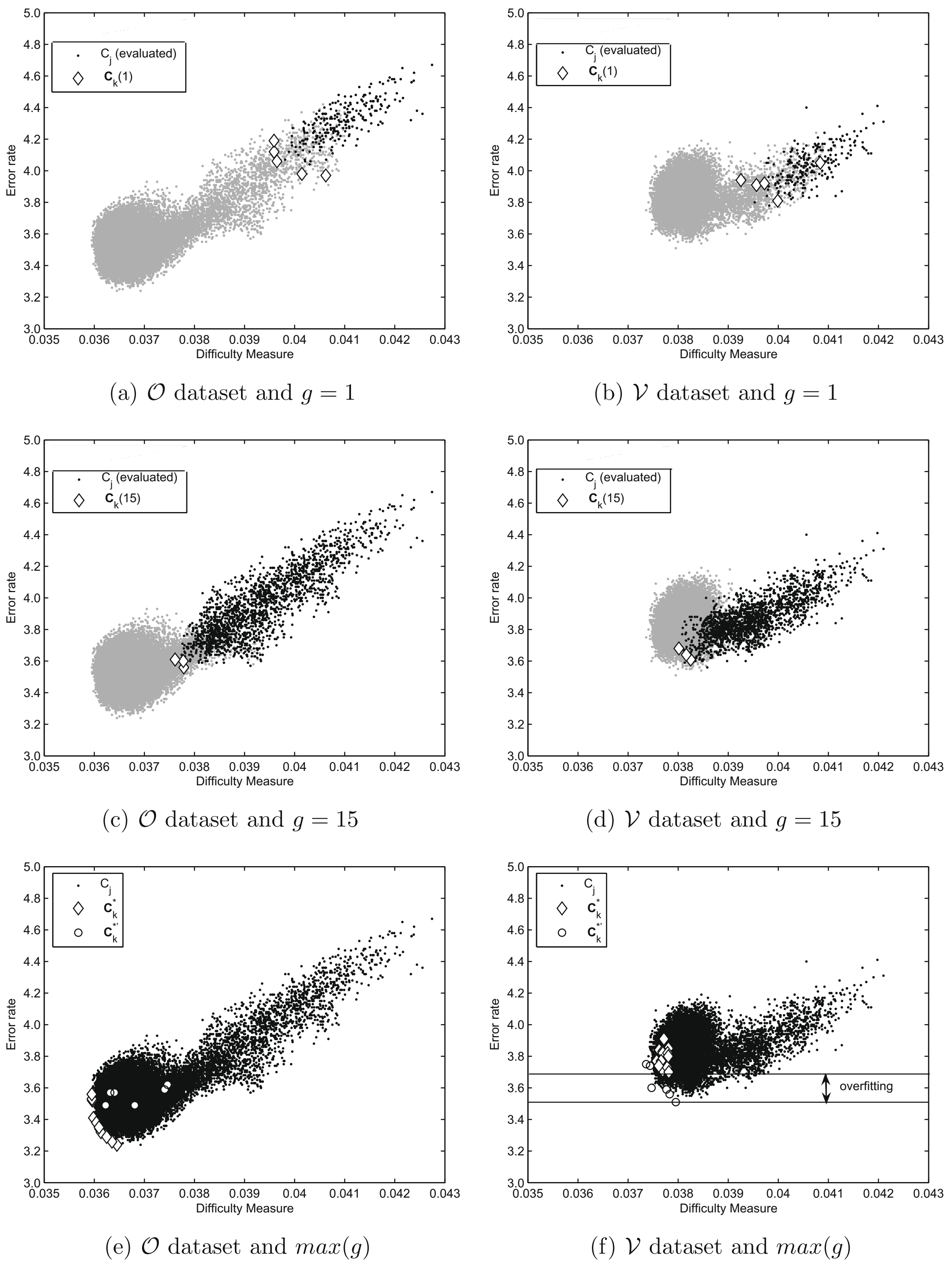
(e) $\mathcal{O}$ dataset and $max(g)$ (f) $\mathcal{V}$ dataset and $max(g)$

Fig. 2. Optimization using GA guided by ϵ. Here, we follow the evolution of $C_j^*(g)$ (diamonds) from $g = 1$ to $max(g)$ (a,c,e) on the optimization dataset $\mathcal{O}$, as well as on the validation dataset $\mathcal{V}$ (b,d,f). The overfitting is measured as the difference in error between $C_j^{*\prime}$ (circles) and C_j^* (f). There is a 0.30% overfit in this example, where the minimal error is reached slightly after $g = 52$ on $\mathcal{V}$, and overfitting is measured by comparing it to the minimal error reached on $\mathcal{O}$. Solutions not yet evaluated are in grey and the best performing solutions are highlighted by arrows.

following reasons: (1) solutions over $\mathbf{C}_k(g)$ found during the optimization process may not be non-dominated solutions over $\mathcal{V}$ and (2) since the evolution of $\mathbf{C}_k(g)$ must be monitored on $\mathcal{V}$, comparing sets of equally important solutions is a complex task. We show in the next section that the use of an auxiliary archive $\mathcal{A}$ makes it possible to control overfitting taking into account these aspects.

3. Overfitting control methods

The focus of this paper is to evaluate strategies that rely on using $\mathcal{V}$ to create an archive $\mathcal{A}$ to control overfitting, since these memory-based strategies may be applied in any optimization problem. The idea is to use the error rate measured on $\mathcal{V}$ as an



(a) $\mathcal{O}$ dataset and $g = 1$ (b) $\mathcal{V}$ dataset and $g = 1$

(c) $\mathcal{O}$ dataset and $g = 15$ (d) $\mathcal{V}$ dataset and $g = 15$

(e) $\mathcal{O}$ dataset and $max(g)$ (f) $\mathcal{V}$ dataset and $max(g)$

Fig. 3. Optimization using NSGA-II and the pair of objective functions: difficulty measure and ϵ. We follow the evolution of $\mathbf{C}_k(g)$ (diamonds) from $g = 1$ to $max(g)$ (a,c,e) on the optimization dataset $\mathcal{O}$, as well as on the validation dataset $\mathcal{V}$ (b,d,f). The overfitting is measured as the difference in error between the most accurate solution in $\mathbf{C}_k^{*'}$ (circles) and in $\mathbf{C}_k^{*}$ (f). There is a 0.20% overfit in this example, where the minimal error is reached slightly after $g = 15$ on $\mathcal{V}$, and overfitting is measured by comparing it to the minimal error reached on $\mathcal{O}$. Solutions not yet evaluated are in grey.

estimation of the generalization error. In order to accomplish the objectives of this paper, we have adapted two overfitting control strategies to the context of classifier ensemble selection for both single- and multi-objective optimization problems: (1) *backwarding* originally proposed in [26] to prevent overfitting in GP; and (2) *global validation* proposed in [23] and [26] to control overfitting in multi-objective evolutionary optimization problems. *Partial validation*, a control strategy traditionally used to avoid overfitting in classifier ensemble selection problems [33] is also investigated. It is interesting to note that these strategies can be ordered with respect to the cardinality of the solution set used for overfitting control: *partial validation* uses only the last population of solutions (or $\mathbf{C}_k^*$ in the MOGA case), *backwarding* validates the best solution (or $\mathbf{C}_k^*(g)$) at each g, while *global validation* uses all solutions at each g.

It is important to mention that the *global validation* strategy is employed in [23] in a complex two-level system which includes feature extraction, feature selection and classifier ensemble selection, all performed by MOGA. In this paper, we adapt their strategy to single-objective GA. As mentioned earlier, we use ensembles generated by bagging (BAG) and RSS.

3.1. Partial validation (PV)

The first and simplest procedure relies on selecting $C_j^{*'}$ at the end of the optimization process by validating the last population $\mathbf{C}(max(g))$ (for GA) or $\mathbf{C}_k^*$ (respectively for MOGA). Consequently, in PV, it is assumed that, $C_j^{*'} \in \mathbf{C}(max(g))$ and $C_j^{*'} \in \mathbf{C}_k^*$, $C_j^{*'}$ denotes the ensemble candidate with the best performance. Hence, there is only one update on $\mathscr{A}$, more precisely for $max(g)$. PV is summarized in Algorithm 2.

Algorithm 2. PV

```
1: Create initial population $\mathbf{C}(1)$ of $w$ chromosomes
2: $\mathscr{A} = \emptyset$
3: for each generation $g \in \{1, \dots, max(g)\}$
4:    perform all genetic operators and generate new population $\mathbf{C}(g+1)$.
5: endfor
6: validate all solutions from $\mathbf{C}(max(g))$ (for GA) or $\mathbf{C}_k^*$ (for MOGA)
7: choose as $C_j^{*'}$ the solution with highest recognition rate
8: update $\mathscr{A}$ by storing $C_j^{*'}$
9: return $C_j^{*'}$ stored in $\mathscr{A}$
```

3.2. Backwarding (BV)

Overfitting remains uncontrolled even when PV is used, because $\mathbf{C}(max(g))$ (for GA) or $\mathbf{C}_k^*$ for MOGA are composed of overfitted solutions, as explained in Section 2. An alternative to PV is to validate not only $\mathbf{C}(max(g))$ (or $\mathbf{C}_k^*$), but the $C_j^*(g)$ (or $\mathbf{C}_k(g)$) found at each generation. The BV method [26] proposed for GP problems is based on this idea. The authors advocate that GP is prone to overfitting especially on later generations as in machine learning tasks. Motivated by this observation, they proposed BV to control overfitting by monitoring the optimization process on $\mathscr{V}$ to determine the point where GP starts to overfit $\mathscr{O}$. This approach uses $\mathscr{A}$ to store the best solution found before overfitting starts to occur. Even though ϵ is not the only objective function used to guide GA in this paper, we use ϵ to represent the objective function in Algorithm 3, which shows how BV is employed with GA. However, other objective functions can be used without loss of generality.

Algorithm 3. BV for GA

```
1: Create initial population $\mathbf{C}(1)$ of $w$ chromosomes
2: $\mathscr{A} = \emptyset$
3: Find $C_j^*(1)$ and set $C_j^{*'} := C_j^*(1)$
4: Store $C_j^{*'}$ in $\mathscr{A}$
5: for each generation $g \in \{1, \dots, max(g)\}$
6:    perform all genetic operators
7:    generate new population $\mathbf{C}(g+1)$ and find $C_j^*(g+1)$ as usual
8:    if $\epsilon(\mathscr{V}, C_j^*)(g+1) < \epsilon(\mathscr{V}, C_j^{*'})$
9:       set $C_j^{*'} := C_j^*(g+1)$
10:      update $\mathscr{A}$ by storing in it the new $C_j^{*'}$
11:   endif
12: endfor
13: return $C_j^{*'}$ stored on $\mathscr{A}$
```

Where multi-objective optimization is concerned, Pareto fronts must be stored in $\mathscr{A}$ instead of individual solutions. Taking into account that BV relies on comparing each new solution found on $\mathscr{O}$ to the solution stored on $\mathscr{A}$, this fact leads to the following question: How can Pareto fronts be compared so as to identify whether or not one Pareto front is better than the others? Because of the various tradeoffs over the Pareto front, the definition of a quality measure is much more complex in multi-objective than in single-objective optimization problems. Some quality measures such as the Pareto front spread, the objective functions spread [37], the epsilon indicator and the coverage function [39] have been proposed. In order to determine whether or not the $\mathbf{C}_k(g)$ found at each generation is better than the $\mathbf{C}_k^{*'}$ stored in $\mathscr{A}$, we propose to use the Pareto quality measure called the *coverage* function, introduced by Zitzler et al. [39].

The coverage function, measured on $\mathscr{V}$, is based on the weak dominance criteria and indicates the number of candidate ensembles in $\mathbf{C}_k(g)$ that are weakly dominated by at least one solution in $\mathbf{C}_k^{*'}$. Given two solutions $C_i \in \mathbf{C}_k^{*'}$ and $C_j \in \mathbf{C}_k(g)$, we may say that C_i covers C_j if C_i is not worse than C_j in all objective functions. The idea is to verify the Pareto improvement among generations on $\mathscr{A}$. Coverage can be denoted as

$$cov(\mathbf{C}_k^{*'}, \mathbf{C}_k(g)). \tag{2}$$

In this way, the average number of solutions in $\mathbf{C}_k^{*'}$ covering the solutions on $\mathbf{C}_k(g)$ is calculated. Thus, $cov(\mathbf{C}_k^{*'}, \mathbf{C}_k(g)) < 1$ or $cov(\mathbf{C}_k^{*'}, \mathbf{C}_k(g)) = 1$, whether $\mathbf{C}_k^{*'}$ covers the entire Pareto $\mathbf{C}_k(g)$ or not. The Pareto improvement at each generation is then measured as

$$imp(\mathbf{C}_k(g), \mathbf{C}_k^{*'}) = 1 - cov(\mathbf{C}_k^{*'}, \mathbf{C}_k(g)). \tag{3}$$

Improvement reaches its maximum ($imp = 1$) when there is no solution on $\mathbf{C}_k(g)$ covered by solutions on $\mathbf{C}_k^{*'}$ and its lowest possible value ($imp = 0$), when all solutions on $\mathbf{C}_k(g)$ are covered by those on $\mathbf{C}_k^{*'}$. Consequently, the update of $\mathscr{A}$ is dependent on the improvement obtained between these two Pareto fronts. When $imp > 0$, $\mathscr{A}$ is updated; otherwise, there is no update. The BV for MOGA is summarized in Algorithm 4.

Algorithm 4. BV for MOGA

```
1: Creates initial population $\mathbf{C}(1)$ of $w$ chromosomes
2: $\mathscr{A} = \emptyset$
3: Find $\mathbf{C}_k(1)$ and set $\mathbf{C}_k^{*'} := \mathbf{C}_k(1)$
4: Store $\mathbf{C}_k^{*'}$ in $\mathscr{A}$
5: for each generation $g$, $g \in \{1, \dots, max(g)\}$
6:    perform all genetic operators;
7:    generate $\mathbf{C}(g+1)$ and find $\mathbf{C}_k(g+1)$ as usual
8:    if $imp(\mathbf{C}_k(g+1), \mathbf{C}_k^{*'}) > 0$
9:       set $\mathbf{C}_k^{*'} := \mathbf{C}_k(g+1)$
10:       update $\mathscr{A}$ by storing in it the new $\mathbf{C}_k^{*'}$
11:   endif
12: endfor
13: return $\mathbf{C}_k^{*'}$ stored on $\mathscr{A}$ to pick up $C_j^{*'}$.
```

3.3. Global validation (GV)

There is a problem with BV, since $C_j^*(g)$ (or $\mathbf{C}_k(g)$ for MOGA) found on $\mathscr{O}$ at each generation, may not be the best solution (or Pareto front) on $\mathscr{V}$. One approach which avoids such a limitation is the global validation GV [23,35]. This approach relies on using $\mathscr{A}$ to validate the entire population $\mathbf{C}(g)$ from each generation, in the context of multi-objective optimization as follows (Algorithm 5): at each g step, all solutions are validated, and thus the set of non-dominated solutions found on $\mathscr{V}$ is stored in $\mathscr{A}$. When the optimization process is completed, two sets of non-dominated solutions are available: (1) the traditional $\mathbf{C}_k^*$ found on $\mathscr{O}$; and (2)

$\mathbf{C}_k^{*'}$, the set of non-dominated solutions found on $\mathscr{V}$. In Fig. 3(f), the solutions composing $\mathbf{C}_k^{*'}$ stored in $\mathscr{A}$ are represented by circles. As can be observed in this figure, the solutions stored in $\mathscr{A}$ are different from the solutions over $\mathbf{C}_k^{*}$.

Algorithm 5. GV for MOGA

```
1: Create initial population $\mathbf{C}(1)$ of $w$ chromosomes
2: $\mathscr{A} = \emptyset$
3: for each generation $g \in \{1, \ldots, max(g)\}$
4:   perform all genetic operators
5:   generate $\mathbf{C}(g+1)$
6:   validate all solutions in $\mathbf{C}(g+1)$ over samples contained in $\mathscr{V}$
7:   perform a non-dominated sorting to $\mathbf{C}(g+1) \cup \mathscr{A}$ to find $\mathbf{C}_k^{*'}$
8:   update $\mathscr{A}$ by storing in it $\mathbf{C}_k^{*'}$
9: endfor
10: return $\mathbf{C}_k^{*'}$ stored in $\mathscr{A}$
```

We propose to change Radtke et al.'s method [23] slightly to adapt it to single-objective optimization problems. Since no Pareto front is involved when using GA, we are interested in the best solution $C_j^{*'}$. In this case, it is sufficient to validate all solutions at each new generation, find the best solution $C_j^{*'}(g)$ and compare it to $C_j^{*'}$, which is stored in $\mathscr{A}$. In this way, we keep the solution $C_j^{*'}$ found on $\mathscr{V}$ stored in $\mathscr{A}$. Letting ϵ represent the objective function, the complete GV algorithm for GA is described in Algorithm 6.

Algorithm 6. GV for GA

```
1: Create initial population $\mathbf{C}(1)$ of $w$ chromosomes
2: $\mathscr{A} = \emptyset$
3: Validate all solutions in $\mathbf{C}(1)$ over samples contained in $\mathscr{V}$
4: Find $C_j^{*'}(1)$ from $\mathbf{C}(1)$
5: Set $C_j^{*'} := C_j^{*'}(1)$
6: for each generation $g \in \{1, \ldots, max(g)\}$
7:   perform all genetic operations and generate $\mathbf{C}(g+1)$
8:   validate all solutions in $\mathbf{C}(g+1)$
9:   find $C_j^{*'}(g+1)$
10:    if $\epsilon(\mathscr{V}, C_j^{*'})(g+1) < \epsilon(\mathscr{V}, C_j^{*'})$
11:      set $C_j^{*'} := C_j^{*'}(g+1)$
12:      update $\mathscr{A}$ by storing in it the new $C_j^{*'}$
13:    endif
14: endfor
15: return $C_j^{*'}$ stored in $\mathscr{A}$
```

An example of the GV strategy for single-objective GA is illustrated in Fig. 2f. This figure shows overfitting when comparing $\epsilon(\mathscr{V}, C_j^{*'})$ and $\epsilon(\mathscr{V}, C_j^{*})$. The ensemble $C_j^{*'}$ probably has higher generalization performance than C_j^{*} obtained in $\mathscr{O}$. In Sections 4 and 5, we analyze such an assumption experimentally.

To summarize, on the one hand PV is applied at the end of the optimization problem to validate the candidate ensembles which compose the last population of solutions (or the Pareto front in the MOGA case). On the other hand, BV and GV are applied during the optimization process. While BV validates only the best solution (or Pareto front), GV validates the whole population of classifier ensembles generated at each generation. In all of the three strategies, the best solution found by the validation process is used to classify the test samples.

4. Experiments

The parameters and the experimental protocol employed are described in Sections 4.1 and 4.2 respectively.

4.1. Parameter settings on experiments

The databases, ensemble construction methods, search algorithms and objective functions used to conduct the experiments are defined in this section.

4.1.1. Databases

It is important to note that the databases must be large enough to be partitioned into the four above-mentioned datasets: $\mathscr{T}$, $\mathscr{O}$, $\mathscr{V}$ and $\mathscr{G}$, to perform experiments using the holdout validation strategy. However, very few large databases containing real classification problems are available in the literature. The approach for selecting classifiers proposed in this paper is general enough to be conducted using small datasets by applying k-fold cross-validation. Accordingly, we performed experiments using both the holdout and 10-fold cross-validation strategies. Another important aspect taken into account in selecting the databases for our experiments is that relatively high-dimensional feature spaces are necessary for the RSS method.

Two databases were used in the holdout validation experiments: (1) NIST Special Database 19 containing digits (NIST SD19), which we call NIST-digits here; and (2) NIST SD19 containing handwritten uppercase letters, which we call NIST-letters here. We use the representation proposed by Oliveira et al. [18], which appears to be well defined and well suited to NIST SD19. The features are a combination of the concavity, contour and surface of characters. The final feature vector is composed of 132 components: 78 for concavity, 48 for contour and 6 for surface. Table 1 lists important information about these two databases and the partitions used to compose the four separate datasets. These same partitions were used in [33] for NIST-digits and in [22] for NIST-letters. We used two test datasets from NIST-digits, which we call data-test1 (60,089 samples) and data-test2 (58,646 samples) here. Data-test2 is well known to be more difficult to use for classification than data-test1 [8]. Table 2 describes the three databases used in the 10-fold cross-validation experiments: dna, texture and satimage.

4.1.2. Ensemble construction methods and base classifiers

We chose kNN and DT as the base classifiers in our experiments. The C4.5 algorithm [21] (Release 8) was used to construct the trees with pruning. In addition, we used $k = 1$ for kNN classifiers in all databases without fine-tuning this parameter in order to avoid additional experiments. BAG and RSS were applied to generate the initial pools of classifiers in our experiments. RSS was proposed by Ho in [9], and works by randomly choosing n different subspaces from the feature space. Each random subspace is used to train one individual classifier c_i. BAG is a bootstrap technique proposed by Breiman [4], which builds n replicate training datasets by randomly sampling, with replacement, from $\mathscr{T}$. Thus, each training dataset is used to train one classifier c_i.

Three initial pools of 100 classifiers were created: (1) 100 DT and (2) 100 kNN, which were generated using RSS, and (3) 100 DT, which was generated using BAG (BAG is mostly effective with unstable classifiers, such as DT). The size of the subsets of features used by RSS is shown in Table 1 for large datasets and in Table 2 for small datasets. The same subspaces are used for both kNN and DT classifiers. Majority voting was used as the combination function. Considering an ensemble of n classifiers, y_i as the output of the i-th classifier and the following set of class labels $\Omega = \{\omega_1, \omega_2 \ldots, \omega_c\}$, majority voting ($mv$) is calculated as

$$mv = \max_{k=1}^{c} \sum_{i=1}^{n} y_{i,k}. \tag{4}$$

4.1.3. Objective functions

Diversity measures and ϵ are applied as objective functions to guide the optimization processes. Kuncheva and Whitaker [13] studied 10 diversity measures. They conclude that these diversity measures may be divided into three different groups, taking into account the correlation among measures: (1) the double-fault

Table 1
Specifications of the large datasets used in the experiments.

Dataset	# of features	Training set ($\mathscr{T}$)	Optimization set ($\mathscr{O}$)	Validation set ($\mathscr{V}$)	Test set ($\mathscr{G}$)	Features RSS	Pool $\mathscr{C}$ size
NIST-digits	132	5000	10,000	10,000	test1 60,089 test2 58,646	32	100
NIST-letters	132	43,160	3980	7960	12,092	32	100

Table 2
Specifications of the small datasets used in the experiments.

Dataset	# of samples	# of features	Features RSS	Pool $\mathscr{C}$ size
dna	3186	180	45	100
texture	5500	40	20	100
satimage	6435	36	18	100

alone; (2) coincident failure diversity (also alone); and (3) the remaining eight diversity measures. Thus, in order to reduce redundancy in our experiments, we chose to employ only four diversity measures: double-fault (δ), coincident failure diversity (σ) and difficulty measure (θ), from the third group, plus ambiguity (γ) as defined in [38], which was not investigated in [13]. These measures are described below:

(1) **Ambiguity**: given the candidate ensemble $C_j = \{c_1, c_2, \ldots, c_l\}$ and its output ω_k, ambiguity is calculated as follows:

$$a_i(x) = \begin{cases} 0 & \text{if } y_i = \omega_k, \\ 1 & \text{otherwise}, \end{cases} \tag{5}$$

where $a_i(x)$ is the ambiguity of the ith classifier for sample x. Considering X as the dataset and $|X|$ its cardinality, and $|C_j|$ as the ensemble's size, the ambiguity for the whole C_j is

$$\gamma = \frac{1}{|X| \cdot |C_j|} \sum_{i \in C_j} \sum_{x \in X} a_i(x). \tag{6}$$

(2) **Coincident failure diversity**: this measure is based on the same distribution F proposed for θ. Here, however, $Y = \frac{i}{l}$ denotes the proportion of classifiers that do not correctly classify x. Given p_q as the probability that q classifiers fail when classifying x, this measure is defined as follows:

$$\sigma = \begin{cases} 0 & p_0 = 1.0, \\ \frac{1}{(1-p_0)} \sum_{q=1}^{l} \frac{q}{l} \frac{(q-1)}{(l-1)} p_q & p_0 < 1. \end{cases} \tag{7}$$

(3) **Difficulty measure**: letting F calculated from $\{\frac{0}{l}, \frac{1}{l}, \ldots, 1\}$, which represents the number of classifiers contained in C_j that correctly classify x, this measure may be calculated as

$$\theta = Var(F). \tag{8}$$

(4) **Double-fault**: let N^{ab} be the number of examples classified in X, where a, b determine whether the classifier is correct (1) or not (0). This pairwise measure is defined for a pair of classifiers c_i and c_k as

$$\delta_{i,k} = \frac{N^{00}}{N^{11} + N^{10} + N^{01} + N^{00}}. \tag{9}$$

4.1.4. Genetic algorithms

The selection of classifier ensembles was applied in this work in the context of GAs based on binary vectors. Since we used baseline ensembles composed of 100 classifiers, each individual was represented by a binary vector with a size of 100. Each bit value determines whether a classifier is active (1) or not (0). The genetic parameters were set as in [3] to avoid the need to run additional experiments using a separate dataset. Table 3 shows the parameter settings employed. The same parameters were used for both GAs in all experiments.

4.2. Experimental protocol

Our experiments were broken down into three main series. In the first and second series, the holdout and 10-fold cross-validation strategies were implemented to verify the impact of overfitting in large and small databases respectively. The validation control methods PV, BV and GV were also compared in order to determine the best method for controlling overfitting. Diversity measures were employed in pairs of objective functions combined with ϵ by NSGA-II, and only ϵ was used to guide GA in both series of experiments. Finally, in the third series, diversity measures were applied individually as single-objective functions. As a result of this last series of experiments, we show that the relationship between diversity and performance may be measured using the GV strategy.

To conduct the 10-fold cross-validation experiments, the original whole datasets were divided into 10 folds. Each time one of the 10 folds was used as $\mathscr{G}$, a second fold was used as $\mathscr{V}$, a third fold was used as $\mathscr{O}$ and the remaining 7-folds were put together to compose $\mathscr{T}$. Thus, $\mathscr{T}$, $\mathscr{O}$ and $\mathscr{V}$ were used to generate the initial pools of classifiers, to perform the optimization process and to perform the three validation strategies respectively. This process was repeated 10 times.

It is also important to mention that each optimization process with the holdout and 10-fold cross-validation strategies was replicated 30 times due to the use of stochastic search algorithms. Then, the best solution for each run was picked up according to the overfitting control strategy employed. The solutions were tested on multiple comparisons using the Kruskal–Wallis nonparametric statistical test by testing the equality between mean values. The confidence level was 95% ($\alpha = 0.05$), and the Dunn–Sidak correction was applied to the critical values. Therefore, the results for the 10-fold cross-validation experiments were obtained as the mean of the error rates over 30 replications, and the final error rates reported in the results tables as the mean of the error rates across 10 trials.

5. Results

The results obtained in each series of experiments are summarized in this section.

5.1. Holdout validation results

Table 4 shows the mean error rates obtained using both GA and NSGA-II search algorithms in **NIST-digits data-test1** and **data-**

Table 3
Genetic algorithms parameters.

Population size	128
Number of generations	1000
Probability of crossover	0.8
Probability of mutation	0.01
One-point crossover and bit-flip mutation	

Table 4
Mean and standard deviation values of the error rates obtained on 30 replications comparing selection procedures on large datasets using GA and NSGA-II. Values in bold indicate that a validation method decreased the error rates significantly, and underlined values indicate that a validation strategy is significantly better than the others.

Method	Val	GA	NSGA-II			
		Error (ϵ)	Ambiguity (γ)	Coincident (σ)	Difficulty (θ)	Double-fault (δ)
NIST-digits - test1 (100 kNN = 3.72; 100DT-RSS = 2.92; 100DT-BAG = 5.65)						
kNN-RSS	NV	3.60 (0.06)	3.66 (0.06)	3.67 (0.03)	3.64 (0.05)	3.63 (0.07)
	PV	3.60 (0.06)	3.70 (0.07)	3.68 (0.04)	3.63 (0.05)	3.64 (0.07)
	BV	3.57 (0.07)	3.70 (0.07)	3.65 (0.03)	3.63 (0.07)	3.64 (0.07)
	GV	**3.55** (0.06)	3.63 (0.06)	3.62 (0.06)	3.60 (0.08)	3.60 (0.09)
DT-RSS	NV	2.82 (0.05)	2.96 (0.10)	2.92 (0.07)	2.81 (0.03)	2.81 (0.04)
	PV	2.80 (0.05)	2.97 (0.10)	2.97 (0.11)	2.83 (0.02)	2.82 (0.05)
	BV	2.83 (0.05)	2.97 (0.10)	2.97 (0.11)	2.83 (0.02)	2.82 (0.06)
	GV	2.84 (0.06)	2.94 (0.08)	2.94 (0.09)	2.82 (0.06)	2.84 (0.07)
DT-BAG	NV	5.20 (0.07)	5.73 (0.07)	5.87 (0.12)	5.67 (0.06)	5.73 (0.08)
	PV	5.18 (0.07)	5.71 (0.08)	5.86 (0.10)	5.68 (0.05)	5.73 (0.08)
	BV	5.18 (0.05)	5.71 (0.08)	5.85 (0.10)	5.69 (0.05)	5.72 (0.09)
	GV	5.18 (0.06)	**5.54** (0.06)	**5.56** (0.11)	**5.55** (0.09)	**5.57** (0.07)
NIST-digits-test2 (100 kNN = 8.10; 100DT-RSS = 6.67; 100DT-BAG = 10.99)						
kNN-RSS	NV	7.91 (0.14)	7.90 (0.14)	8.09 (0.16)	8.12 (0.09)	8.12 (0.09)
	PV	7.89 (0.12)	7.97 (0.15)	8.11 (0.18)	8.11 (0.11)	8.14 (0.12)
	BV	7.85 (0.17)	7.97 (0.15)	8.10 (0.18)	8.11 (0.10)	8.14 (0.14)
	GV	7.80 (0.13)	7.87 (0.13)	**7.94** (0.15)	**7.93** (0.10)	**7.93** (0.13)
DT-RSS	NV	6.53 (0.08)	6.76 (0.17)	6.77 (0.14)	6.59 (0.11)	6.60 (0.10)
	PV	6.50 (0.09)	6.79 (0.18)	6.88 (0.21)	6.69 (0.06)	6.65 (0.11)
	BV	6.53 (0.09)	6.79 (0.18)	6.88 (0.21)	6.69 (0.06)	6.65 (0.12)
	GV	6.53(0.09)	6.73 (0.13)	6.76 (0.12)	6.59 (0.09)	6.63 (0.13)
DT-BAG	NV	10.16(0.15)	10.99 (0.17)	11.28 (0.28)	11.05 (0.08)	11.13 (0.08)
	PV	10.11 (0.15)	10.96 (0.17)	11.25 (0.23)	11.06 (0.11)	11.12 (0.21)
	BV	10.11(0.13)	10.94 (0.17)	11.24 (0.24)	11.06 (0.10)	11.10 (0.14)
	GV	10.09(0.13)	**10.68** (0.11)	**10.76** (0.22)	**10.76** (0.15)	**10.83** (0.07)
NIST-letter (100kNN = 6.60; 100DT-RSS = 6.06; 100DT-BAG = 7.63)						
kNN-RSS	NV	6.49 (0.11)	6.58 (0.10)	6.73 (0.25)	6.41 (0.09)	6.55 (0.13)
	PV	6.50 (0.09)	6.58 (0.11)	6.71 (0.26)	6.43 (0.11)	6.60 (0.12)
	BV	6.47 (0.12)	6.58 (0.12)	6.71 (0.26)	6.43 (0.11)	6.58 (0.12)
	GV	6.49 (0.09)	6.54 (0.14)	**6.50** (0.13)	6.33 (0.16)	**6.45** (0.12)
DT-RSS	NV	6.03 (0.07)	6.24 (0.11)	6.34 (0.19)	5.93 (0.08)	6.10 (0.12)
	PV	6.03 (0.07)	6.25 (0.11)	6.34 (0.19)	5.92 (0.06)	6.11 (0.12)
	BV	6.05 (0.09)	6.24 (0.11)	6.34 (0.19)	5.92 (0.07)	6.03 (0.13)
	GV	6.01 (0.09)	**6.08** (0.09)	**6.12** (0.14)	5.89 (0.07)	6.02 (0.10)
DT-BAG	NV	7.71 (0.10)	7.68 (0.09)	8.00 (0.14)	7.56 (0.08)	7.78 (0.12)
	PV	7.69 (0.08)	7.64 (0.07)	8.08 (0.18)	7.51 (0.05)	7.78 (0.13)
	BV	7.71 (0.10)	7.64 (0.07)	8.03 (0.16)	7.51 (0.05)	7.76 (0.12)
	GV	7.70 (0.08)	7.64 (0.07)	**7.87** (0.14)	7.54 (0.08)	**7.64** (0.12)

test2, and in **NIST-letters**. The error rates obtained by combining the initial pool of 100 classifiers are also included in this table as well as the results with no overfitting control, denoted NV.

These experiments showed the following:

(1) The results from the literature [12], which conclude that complex learning problems are more affected by overfitting, are confirmed. There are more problems presenting overfitting in **NIST-digits data-test2** than in **data-test1**. As we mentioned in Section 4.1.1, **NIST-digits data-test2** is more difficult to use for classification.
(2) NSGA-II is more prone to overfitting than GA. Even though, in the majority of the experiments using GA, at least one of the validation methods slightly decreased the error rates when compared with NV, the differences are not significant. In contrast, these differences are more likely to be significant in experiments with NSGA-II. Of 36 cases using NSGA-II, an overfitting control decreased the error rates in 30. In 17 of these experiments, the differences were significant.
(3) When overfitting was detected, GV outperformed both PV and BV. The Kruskal–Wallis test shows that, among 19 cases where overfitting control decreased the error rates significantly, GV was the best strategy in 18 problems.
(4) In terms of ensemble creation methods, our results indicate an order relation between the methods investigated for the **NIST-digits** database. BAG was more prone to overfitting than RSS. In addition, ensembles of DT were less prone to overfitting than ensembles of kNN, both generated using RSS. For the **NIST-letters** database, the results were equivalent.
(5) Although this paper does not focus on comparing ensemble creation methods in terms of performance, our results indicate that ensembles generated with BAG performed worse than ensembles generated by RSS. Also, DT ensembles performed better than kNN.
(6) Classifier ensemble selection was a better option than combining the initial pools of classifiers. For **NIST-digits**, GA using ϵ as the objective function, and for **NIST-letters**, NSGA-II using θ and ϵ as the pair of objective functions, found solutions better than the baseline composed of 100 classifiers in all three problems.

5.2. Cross-validation results

Table 5 summarizes the mean values and the standard deviation values achieved on the three small datasets presented in Table 2. We also report the error rates obtained on combining the initial pools of classifiers by majority voting.

Table 5
Mean and standard deviation values of the error rates obtained on 30 replications comparing selection procedures on small datasets using GA and NSGA-II. Values in bold indicate that a validation method decreased the error rates significantly, and underlined values indicate when a validation strategy is significantly better than the others.

Method	Val	GA	NSGA-II			
		Error (ϵ)	Ambiguity (γ)	Coincident (σ)	Difficulty (θ)	Double-fault (δ)
Texture (100 kNN = 1.11; 100DT-RSS = 2.56; 100DT-BAG = 3.60)						
kNN-RSS	NV	1.09 (0.03)	1.28 (0.03)	1.19 (0.06)	0.87 (0.03)	0.90 (0.05)
	PV	1.09 (0.04)	1.40 (0.06)	1.21 (0.07)	0.84 (0.03)	0.94 (0.06)
	BV	1.08 (0.04)	1.41 (0.06)	1.21 (0.08)	0.88 (0.04)	0.89 (0.06)
	GV	1.11 (0.03)	1.26 (0.06)	1.18 (0.08)	0.95 (0.05)	0.98 (0.09)
DT-RSS	NV	2.54 (0.07)	2.94 (0.11)	3.16 (0.15)	2.40 (0.05)	3.00 (0.14)
	PV	2.52 (0.09)	3.03 (0.14)	3.41 (0.19)	2.41 (0.06)	3.09 (0.11)
	BV	2.55 (0.08)	3.04 (0.15)	3.42 (0.15)	2.44 (0.06)	3.10 (0.10)
	GV	2.51 (0.09)	2.93 (0.15)	3.20 (0.21)	2.41 (0.09)	2.91 (0.18)
DT-BAG	NV	3.58 (0.10)	3.60 (0.09)	4.09 (0.15)	3.49 (0.09)	4.02 (0.14)
	PV	3.53 (0.08)	3.63 (0.13)	4.18 (0.20)	3.55 (0.08)	4.14 (0.19)
	BV	3.60 (0.08)	3.62 (0.11)	4.17 (0.18)	3.46 (0.12)	4.14 (0.20)
	GV	3.58 (0.08)	3.63 (0.10)	4.01 (0.15)	3.43 (0.11)	4.04 (0.22)
DNA (100 kNN = 6.87; 100DT-RSS = 5.05; 100DT-BAG = 5.02)						
kNN-RSS	NV	8.01 (0.27)	9.61 (0.36)	10.60 (0.51)	8.86 (0.22)	9.21 (0.35)
	PV	8.02 (0.2)	9.85 (0.35)	10.85 (0.55)	8.93 (0.20)	9.38 (0.39)
	BV	**7.64** (0.24)	9.82 (0.37)	10.76 (0.52)	8.85 (0.21)	9.37 (0.41)
	GV	**7.69** (0.23)	**8.60** (0.37)	**8.75** (0.45)	**8.01** (0.33)	**8.36** (0.30)
DT-RSS	NV	5.10 (0.24)	6.44 (0.25)	7.05 (0.43)	5.15 (0.18)	6.20 (0.27)
	PV	4.99 (0.18)	6.76 (0.31)	7.30 (0.46)	4.96 (0.12)	6.31 (0.30)
	BV	4.98 (0.25)	6.76 (0.30)	7.30 (0.47)	4.93 (0.14)	6.30 (0.30)
	GV	4.97 (0.16)	**5.93** (0.29)	**6.36** (0.42)	**4.77** (0.17)	**5.67** (0.33)
DT-BAG	NV	5.00(0.10)	5.35 (0.14)	5.51 (0.22)	5.13 (0.14)	5.55 (0.23)
	PV	4.99 (0.11)	5.50 (0.12)	5.69 (0.20)	5.08 (0.12)	5.35 (0.14)
	BV	4.98 (0.15)	5.51 (0.11)	5.69 (0.20)	4.99 (0.12)	5.38 (0.15)
	GV	4.99 (0.08)	5.36 (0.20)	5.41 (0.21)	4.93 (0.13)	5.54 (0.22)
Satimage (100kNN = 8.59; 100DT-RSS = 8.64; 100DT-BAG = 9.59)						
kNN-RSS	NV	8.64 (0.09)	8.76 (0.12)	9.12 (0.16)	8.81 (0.23)	9.15 (0.14)
	PV	8.62 (0.07)	8.76 (0.09)	9.25 (0.17)	8.88 (0.25)	9.28 (0.13)
	BV	8.57 (0.10)	8.75 (0.09)	9.23 (0.18)	8.88 (0.26)	9.33 (0.14)
	GV	8.58 (0.11)	**8.69** (0.10)	9.05 (0.17)	8.84 (0.18	9.15 (0.17)
DT-RSS	NV	8.83 (0.11)	8.92 (0.11)	9.44 (0.20)	8.88 (0.10)	9.25 (0.19)
	PV	8.80 (0.09)	9.00 (0.12)	9.63 (0.22)	8.93 (0.12)	9.38 (0.18)
	BV	8.81 (0.09)	9.00 (0.10)	9.61 (0.23)	8.94 (0.11)	9.39 (0.17)
	GV	8.82 (0.12)	**8.77** (0.11)	**9.19** (0.18)	**8.75** (0.13)	**9.03** (0.17)
DT-BAG	NV	9.63 (0.09)	9.81 (0.10)	10.40 (0.19)	9.91 (0.13)	10.14 (0.18)
	PV	9.61 (0.07)	9.81 (0.12)	10.55 (0.18)	9.89 (0.09)	10.34 (0.17)
	BV	9.65 (0.10)	9.82 (0.12)	10.53 (0.17)	9.91 (0.09)	10.35 (0.16)
	GV	9.63 (0.09)	**9.65** (0.14)	10.25 (0.22)	**9.71** (0.13)	10.10 (0.16)

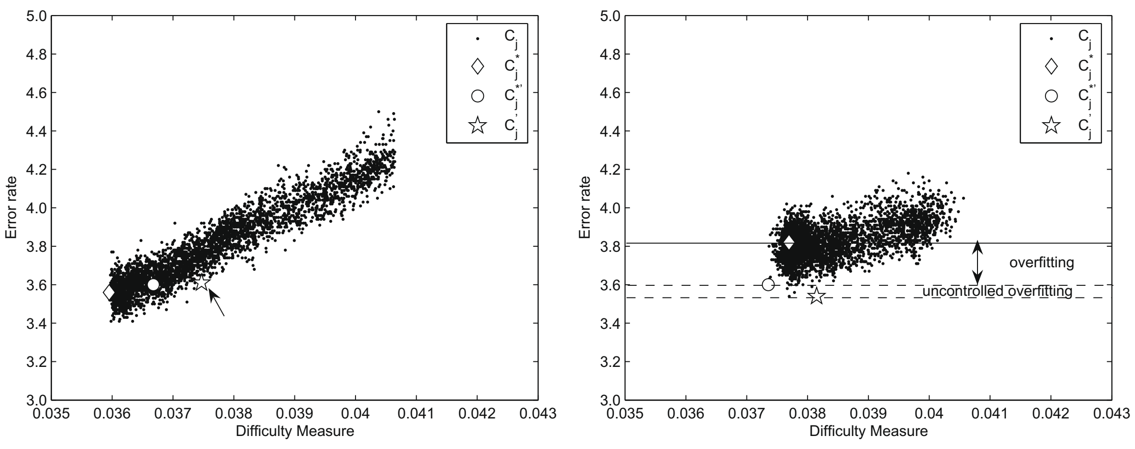
Based on these results, it may be observed that:

(1) For small databases, NSGA-II was also more prone to overfitting than GA. The overfitting control strategies significantly decreased the error rates of the solutions found by GA only in one case (**dna** with kNN-based ensembles) out of nine, while they significantly reduced the error rates of the solutions found by NSGA-II in 16 of 36 cases.
(2) In 15 cases in which overfitting was detected, the difference between the overfitting control methods was significant. GV was also the best strategy for controlling overfitting in this series of experiments.
(3) **Texture** is a highly stable database: it has the same class distribution and it can be easily classified, since the error rates can be lower than 1% (see results obtained using θ and ϵ to guide NSGA-II). Hence, it is not surprising that our results do not show overfitting in this database.
(4) For **dna**, overfitting was detected in the two problems involving ensembles generated using RSS. For **satimage**, DT-based ensembles generated using RSS were more prone to overfitting than the other two methods. Thus, unlike the **NIST-digits** results, BAG was less prone to overfitting than RSS for these two databases.
(5) RSS-based ensembles presented lower error rates than BAG-based ensembles in all three databases.
(6) The solutions found by NSGA-II guided by θ and ϵ were better than the baseline combination for **texture** and **dna**, except for ensembles of kNN for **dna**. However, for **satimage**, selection did not outperform combination.

5.3. Relationship between performance and diversity

When GA was guided by diversity measures as single-objective functions, we observed that a relationship between diversity measures and performance could be measured using the GV strategy, as illustrated in Fig. 4. The same optimization problem investigated in Section 2 is shown here. GA was guided by the minimization of θ as the objective function. It is important to mention that the optimization was only guided by θ. The reason for showing plots of ϵ versus θ in this figure is to demonstrate the relationship between this diversity measure and performance.

Fig. 4(a) shows all C_j evaluated during the optimization process (points) and C_j^* (in terms of θ values) found on $\mathcal{O}$. It is interesting to note that Fig. 4b confirms that the overfitting problem is also detected when θ is the objective function used to guide GA, since $\epsilon(\mathcal{V}, C_j^*) > \epsilon(\mathcal{V}, C_j^{*'})$. Thus, in this example, there is an overfitting measured by $\epsilon(\mathcal{V}, C_j^{*'}) - \epsilon(\mathcal{V}, C_j^*) = 0.23$. The use of GV allows us to keep $(C_j^{*'})$ stored in $\mathcal{A}$. However, Fig. 4b also shows that $C_j^{*'}$ is not the solution with the smallest ϵ among all candidate ensembles



(a) $\mathcal{O}$ dataset (b) $\mathcal{V}$ dataset

Fig. 4. Optimization using GA with θ as the objective function. In (a) all C_j evaluated during the optimization process (points), C_j^* (diamonds), $C_j^{*'}$ (circles) and C_j' (stars) are shown on $\mathcal{O}$; controlled and uncontrolled overfitting using GV are shown in (b). Arrows highlight C_j'.

evaluated. In fact, the solution denoted C_j' is the candidate ensemble with lowest ϵ.

These observations lead us to propose the use of GV to measure the relationship between diversity and performance. The assumption is that there is a relationship between these two measures when the generalization performance is increased by keeping the solution $C_j^{*'}$ stored in $\mathcal{A}$ (based on diversity values). In addition, the relationship may range from weak to strong, based on the difference (overfitting) between $\epsilon(\mathcal{V}, C_j^{*'})$ and $\epsilon(\mathcal{V}, C_j')$. For instance, since $\epsilon(\mathcal{V}, C_j') - \epsilon(\mathcal{V}, C_j^{*'}) = 0.05$ in Fig. 4b, a 0.23% overfitting was controlled using GV and a 0.05% overfitting remains uncontrolled. Because this difference is close to 0, θ is assumed to be strongly related to performance in this problem.

Taking into account that the relationship between diversity and performance has been measured in the literature using correlation measures [13] and kappa-error diagrams [5] for example, the use of GV offers a different strategy for analyzing such a key aspect in classifier ensembles literature. However, there may be a drawback to this strategy: diversity measures are not independent of ensemble size [1,17], i.e. when diversity measures are employed as single-objective functions, a minimization of the number of classifiers may result. It is expected that ensembles which are too small will perform considerably less well than other approaches, such as combining the initial pools of classifiers or selecting the best subset of classifiers using GA only guided by ϵ or MOGA guided by ϵ combined with diversity.

Table 6
Mean and standard deviation values of the error rates obtained on measuring the uncontrolled overfitting. The relationship between diversity and performance is stronger as ϕ decreases. The best result for each case is shown in bold.

Method	Strategy	Ambiguity (γ)	Coincident (σ)	Difficulty (θ)	Double-fault (δ)
Texture					
	$\epsilon(\mathcal{V}, C_j^{*'})$	1.20 (0.04)	1.12 (0.03)	**1.01** (0.03)	1.01 (0.02)
kNN-RSS	$\epsilon(\mathcal{V}, C_j')$	1.12 (0.03)	1.09 (0.04)	1.09 (0.04)	1.09 (0.05)
GA $\epsilon = 1.11$	ϕ	0.08 (0.03)	0.03 (0.04)	−0.08 (0.04)	−0.08 (0.03)
	$\epsilon(\mathcal{V}, C_j^{*'})$	2.80 (0.04)	2.50 (0.08)	**2.40** (0.06)	2.46 (0.07)
DT-RSS	$\epsilon(\mathcal{V}, C_j')$	2.63 (0.08)	2.50 (0.07)	2.43 (0.09)	2.46 (0.09)
GA $\epsilon = 2.51$	ϕ	0.18 (0.06)	0.00 (0.07)	−0.03 (0.07)	0.00 (0.08)
	$\epsilon(\mathcal{V}, C_j^{*'})$	3.59 (0.09)	3.56 (0.07)	**3.48** (0.06)	3.49 (0.06)
DT-BAG	$\epsilon(\mathcal{V}, C_j')$	3.61 (0.09)	3.61 (0.08)	3.54 (0.06)	3.57 (0.08)
GA $\epsilon = 3.58$	ϕ	−0.02(0.09)	−0.05 (0.08)	−0.06 (0.06)	−0.08 (0.07)
DNA					
	$\epsilon(\mathcal{V}, C_j^{*'})$	10.6 (0.00)	8.33 (0.21)	**7.50** (0.25)	7.89 (0.16)
kNN-RSS	$\epsilon(\mathcal{V}, C_j')$	9.05 (0.00)	8.00 (0.33)	8.44 (0.30)	7.75 (0.29)
GA $\epsilon = 7.69$	ϕ	1.56 (0.00)	0.33 (0.27)	−0.94 (0.30)	0.14 (0.20)
	$\epsilon(\mathcal{V}, C_j^{*'})$	7.40 (0.16)	5.01 (0.18)	**4.50** (0.16)	4.67 (0.10)
DT-RSS	$\epsilon(\mathcal{V}, C_j')$	5.17 (0.18)	5.09 (0.20)	5.24 (0.22)	4.91 (0.19)
GA $\epsilon = 4.97$	ϕ	2.23 (0.17)	−0.07(0.19)	−0.73 (0.19)	−0.24 (0.14)
	$\epsilon(\mathcal{V}, C_j^{*'})$	5.02 (0.11)	5.16 (0.10)	**4.93** (0.09)	4.94 (0.07)
DT-BAG	$\epsilon(\mathcal{V}, C_j')$	5.00 (0.10)	5.08 (0.10)	5.05 (0.12)	5.01 (0.09)
GA $\epsilon = 4.99$	ϕ	0.03 (0.10)	0.08 (0.10)	−0.12 (0.10)	−0.07 (0.08)
Satimage					
	$\epsilon(\mathcal{V}, C_j^{*'})$	8.60 (0.09)	**8.58** (0.11)	8.64 (0.08)	8.62 (0.10)
kNN-RSS	$\epsilon(\mathcal{V}, C_j')$	8.55 (0.07)	8.60 (0.08)	8.66 (0.12)	8.62 (0.09)
GA $\epsilon = 8.58$	ϕ	0.05 (0.08)	−0.02 (0.09)	−0.02 (0.10)	0.00 (0.09)
	$\epsilon(\mathcal{V}, C_j^{*'})$	**8.22** (0.12)	8.77 (0.11)	8.34 (0.03)	8.67 (0.11)
DT-RSS	$\epsilon(\mathcal{V}, C_j')$	8.11 (0.12)	8.72 (0.11)	8.47 (0.14)	8.71 (0.11)
GA $\epsilon = 8.82$	ϕ	0.11 (0.12)	0.05 (0.11)	−0.10 (0.08)	−0.04 (0.10)
	$\epsilon(\mathcal{V}, C_j^{*'})$	9.61 (0.09)	9.64 (0.13)	**9.61** (0.09)	9.71 (0.11)
DT-BAG	$\epsilon(\mathcal{V}, C_j')$	9.66 (0.15)	9.65 (0.13)	9.66 (0.12)	9.70 (0.10)
GA $\epsilon = 9.63$	ϕ	−0.05 (0.12)	−0.01 (0.13)	−0.05 (0.10)	0.01 (0.10)

To avoid the problem of reaching a too small ensemble size, we defined a large fixed minimum ensemble size for all diversity measures in this series of experiments. The minimum ensemble size was fixed based on the median size of the ensembles obtained using ϵ as a single-objective function calculated on the three small datasets described in Table 2. Taking into account these results, we set the minimum ensemble size to 49 classifiers.

The four diversity measures presented in Section 4.1.3 were used in this series of experiments. The objective of the experiments was to verify which measure is the most closely related to performance. The relationship between the four diversity measures and performance is measured by calculating the uncontrolled overfitting $\phi = \epsilon(\mathscr{V}, C_j') - \epsilon(\mathscr{V}, C_j^{*'})$. The lower ϕ, the stronger the relationship.

Table 6 summarizes the results obtained. We have also included the results achieved using GA guided only by ϵ, so that the solutions found by all five single-objective functions investigated in this paper can be compared. These results are the same as those shown in Table 5.

Our results indicate that the differences among the results obtained with diversity measures are small, except for **dna** in RSS-based ensembles. In fact, we confirmed the results of previous work, e.g. [27,13], that diversity measures are highly correlated. However, our experiments reveal important information about the relationship between performance and diversity. Taking into account the assumption that the closer to zero the uncontrolled overfitting, the stronger the relationship between the two measures, we can observe the following results:

(1) The diversity measure δ was more closely related to performance. The uncontrolled overfitting ϕ was either zero or as close as possible to zero in five cases out of a total of nine cases.
(2) The diversity measure γ was less closely related to performance. The results obtained using γ presented the highest ϕ in six cases.
(3) θ was highly related to performance and appears to be better than ϵ for guiding classifier ensemble selection. In eight cases, θ found solutions better than those found using ϵ. In addition, the ensembles found using θ presented the highest negative ϕ. What this means, in effect, is that guiding GV using θ is better than doing so using ϵ.
(4) Our results indicate that diversity measures can be effective objective functions for selecting high-performance classifier ensembles. The main problem is that diversity measures are critically affected by ensemble size. Since we have fixed quite a large ensemble size, i.e. 49 classifiers, diversity measure performances exceeded those of ϵ.

6. Conclusions

This work presented the experimental results of a study comparing three overfitting control strategies adapted to classifier ensemble selection performed as optimization problems using single- and multi-objective genetic algorithms. We showed that the task of selecting classifier ensembles may be prone to overfitting, we pointed out the best strategy for controlling overfitting and we used a global validation strategy to measure the relationship between diversity and performance. The following overfitting control methods were investigated: (1) partial validation, in which only the last population of solutions is validated; (2) backwarding, which relies on validating each best solution at each generation, and (3) global validation, in which all solutions at each generation are validated. Three initial pools of 100 classifiers were generated: 100 kNN and 100 decision trees using the random subspace method, and a third pool of 100 decision Trees using Bagging. Five different objective functions were applied: four diversity measures and the error rate.

The experiments were divided into three series. The three overfitting control methods were compared in two large databases (first series of experiments) and in three small datasets (second series). The error rate was employed as a single-objective function by single-objective genetic algorithm and the four diversity measures were combined with the error rate to make up pairs of objective functions to guide the multi-objective genetic algorithm (NSGA-II). Finally, in the third series of experiments, the four diversity measures were directly applied by single-objective genetic algorithm, and the global validation strategy was applied to control overfitting. The objective in this third series of experiments was to show how the global validation strategy can be used for identifying the diversity measure more closely related to performance.

The results show that overfitting can be detected in classifier ensemble selection, especially when NSGA-II is employed as the search algorithm. In response to the following question: which is the best strategy employing an archive for reducing overfitting in classifier ensemble selection when this selection is formulated as an optimization problem? posed in the introduction, the global validation may be deemed to be the best strategy. In all problems where overfitting was detected, in both large and small databases, the global validation strategy outperformed the other overfitting control methods. In response to the second question: are classifier ensembles generated by bagging and random subspace equally affected by overfitting? our results indicate that classifier ensembles generated by the random subspace method are more prone to overfitting than the bagging-based ones, even though the same behavior was not observed in all databases investigated. In terms of ensembles generated by the random subspace method, our results globally showed that kNN ensembles and decision trees ensembles are equally affected by overfitting. This aspect is totally problem-dependent.

Finally, our results outlined a relationship between performance and diversity. Double-fault and ambiguity were the diversity measures more and less closely related to performance, respectively. These results confirm what is reported in the literature, which is that double-fault is highly correlated with majority voting error [27] and that measures which do not need information about the correctness of the classifier output, such as ambiguity, are less correlated to the performance [1]. Moreover, we show that the difficulty measure can be better than the error rate as a single-objective function for selecting high-performance classifier ensembles. It is important, however, to realize that quite a large minimum ensemble size (49) was fixed for all diversity measures. Such a size was defined based on the size of the ensembles found by GA guided by the error rate.

The next stage of this research will involve determining strategies to improve the selection results, such as a post-processing level employing dynamic classifier selection. Although we were able to increase the generalization performance of the baseline initial pool of classifiers by selecting subsets of classifiers, the gain in performance can be increased still further.

References

[1] M. Aksela, J. Laaksonen, Using diversity of errors for selecting members of a committee classifier, Pattern Recognition 39 (4) (2006) 608–623.
[2] S. Bandyopadhyay, S. Pal, B. Aruna, Multiobjective gas, quantitative indices, and pattern classification, IEEE Transactions on System, Man, and Cybernetics – Part B 34 (5) (2004) 2088–2099.
[3] N. Benahmed, Optimisation de réseaux de neurones pour la reconnaissance de chiffres manuscrits isolés: Sélection et pondération des primitives par algorithmes génétiques, Master's thesis, École de technologie supérieure, 2002.
[4] L. Breiman, Bagging predictors, Machine Learning 24 (2) (1996) 123–140.
[5] T. Dietterich, An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization, Machine Learning 40 (2) (2000) 139–158.

[6] Y. Freund, R. Schapire, Experiments with a new boosting algorithm, in: Proceedings of XIII International Conference on Machine Learning, vol. 148, 1996, p. 156.
[7] H. Frohlich, O. Champelle, Feature selection for support vector machines by means of genetic algorithms, in: Proceedings of IEEE International Conference on Tools with AI, vol. 142, 2003, p. 148.
[8] P. Gother, NIST Special Database 19 – Handprited forms and characters database, National Institute of Standard and Technology – NIST:database CD documentation, 1995.
[9] T. Ho, The random subspace method for constructing decision forests, IEEE Transactions on PAMI 20 (8) (1998) 832–844.
[10] K. Deb, Multi-Objective Optimization using Evolutionary Algorithms, John Wiley & Sons Ltd., England, 2001.
[11] J. Kittler, M. Hatef, R.P. Duin, J. Matas, On combining classifiers, IEEE Transactions on PAMI 20 (3) (1998) 226–239.
[12] R. Kohavi, D. Sommerfield, Feature subset selection using the wrapper model: Overfitting and dynamic search space topology, in: Proceedings of The First International Conference on Knowledge Discovery and Data Mining, vol. 192, 1995, p. 197.
[13] L. Kuncheva, C. Whitaker, Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy, Machine Learning 51 (2) (2003) 181–207.
[14] X. Llorà, D. Goldberg, I. Traus, E.B.I. Mansilla, Accuracy, parsimony, and generality in evolutionary learning systems via multiobjective selection, in: International Workshop in Learning Classifier Systems, vol. 118, 2002, p. 142.
[15] J. Loughrey, P. Cunningham, Overfitting in wrapper-based feature subset selection: the harder you try the worse it gets, in: Proceedings of International Conference on Innovative Techniques and Applications of Artificial Intelligence, vol. 33, 2004, p. 43.
[16] T.M. Mitchell, Machine Learning, McGraw-Hill, USA, 1997.
[17] N. Narasimhamurthy, Evaluation of diversity measures for binary classifier ensembles, in: Proceedings of MCS, Seaside, USA, 2005, pp. 267–277.
[18] L. Oliveira, R. Sabourin, F. Bortolozzi, C. Suen, Automatic recognition of handwritten numerical strings: a recognition and verification strategy, IEEE Transactions on PAMI 24 (11) (2002) 1438–1454.
[19] D. Partridge, W. Yates, Engineering multiversion neural-net systems, Neural Computation 8 (4) (1996) 869–893.
[20] M. Prior, T. Windeatt. Over-fitting in ensembles of neural networks classifiers within ecoc frameworks. In Proceedings of MCS, Seaside, USA, 2005, pp. 286–295.
[21] J. Quinlan, Programs for Machine Learning, Morgan Kaufmann, 1993.
[22] P. Radtke, R. Sabourin, T. Wong, Classification system optimization with multi-objective genetic algorithms, in: Proceedings of 10th International Workshop on Frontiers in Handwriting Recognition, La Baule, France, 2006.
[23] P. Radtke, R. Sabourin, T. Wong, Impact of solution over-fit on evolutionary multi-objective optimization for classification systems, in: Proceedings of IJCNN, Vancouver, Canada, 2006.
[24] G. Ratsch, T. Onoda, K.-R. Mller, An improvement of adaboost to avoid overfitting, in: Proceedings of The Fifth International Conference on Neural Information Processing, Kitakyushu, Japan, 1998.
[25] R. Reed, Pruning algorithms – a survey, IEEE Transactions on Neural Networks 4 (5) (1993) 740–747.
[26] D. Robilliard, C. Fonlupt, Backwarding: an overfitting control for genetic programming in a remote sensing application, in: Proceedings of the Fifth European Conference on Artificial Evolution, 2000, pp. 245–254.
[27] D. Ruta, B. Gabrys, Classifier selection for majority voting, Information Fusion 6 (1) (2005) 163–168.
[28] E. Santos, Static and dynamic overproduction and selection of classifier ensembles with genetic algorithms, PhD thesis, École de technologie supérieure – University of Quebec, 2008.
[29] E. Santos, L. Oliveira, R. Sabourin, P. Maupin, Overfitting in the selection of classifier ensembles: a comparative study between pso and ga, in: Proceedings of Genetic and Evolutionary Computation Conference, 2008, pp. 1423–1424.
[30] E. Santos, R. Sabourin, P. Maupin, Pareto analysis for the selection of classifier ensembles, in: Proceedings of Genetic and Evolutionary Computation Conference, 2008, pp. 553–558.
[31] A. Sharkey, N. Sharkey, The "test and select approach to ensemble combination, in: J. Kittler, F. Roli (Eds.), Proceedings of the I International Workshop on Multiple Classifier System, 2000, pp. 30–44.
[32] K. Sirlantzis, M. Fairhurst, R. Guest, An evolutionary algorithm for classifier and combination rule selection in multiple classifier system, in: Proceedings of ICPR, 2002, 771–774.
[33] G. Tremblay, R. Sabourin, P. Maupin, Optimizing nearest neighbour in random subspaces using a multi-objective genetic algorithm, in: Proceedings of ICPR, 2004, pp. 208–211.
[34] A. Tsymbal, M. Pechenizkiy, P. Cunningham, Sequential genetic search for ensemble feature selection, in: Proceedings of International Joint Conference on Artificial Intelligence, 2005, pp. 877–882.
[35] S. Wiegand, C. Igel, U. Handmann, Evolutionary multi-objective optimization of neural networks for face detection, International Journal of Computational Intelligence and Applications 4 (3) (2004) 237–253.
[36] H. Wu, J. Shapiro, Does overfitting affect performance in estimation of distribution algorithms, in: Proceedings of the Genetic and Evolutionary Computation Conference, Seattle, USA, 2006, pp. 433–434.
[37] J. Wu, S. Azarm, Metrics for quality assessment of a multiobjective design optimization solution set, Trans. ASME Journal of Mechanical Design 123 (2001) 18–25.
[38] G. Zenobi, P. Cunningham, Using diversity in preparing ensembles of classifiers based on different feature subsets to minimize generalization error, in: Proceedings of European Conference on Machine Learning, 2001, pp. 576–587.
[39] E. Zitzler, L. Thiele, M. Laumanns, C. Fonseca, V. Fonseca, Performance assessment of multiobjective optimizers: an analysis and review, IEEE Transactions on Evolutionary Computation 7 (2) (2003) 117–132.