Integrated Computer-Aided Engineering 0 (2015) 1–14
IOS Press

Using metaheuristics to optimize the combination of classifier and cluster ensembles

Luiz F. S. Coletta [a,*], Eduardo R. Hruschka [a], Ayan Acharya [b] and Joydeep Ghosh [b]
[a] *Department of Computer Science, University of Sao Paulo (USP) at Sao Carlos, Brazil*
E-mail: {luizfsc,erh}@icmc.usp.br
[b] *Department of Electrical & Computer Engineering, University of Texas (UT) at Austin, USA*
E-mail: aacharya@utexas.edu, ghosh@ece.utexas.edu

Abstract. We investigate how to make a simpler version of an existing algorithm, named C^{3}E, from Consensus between Classification and Clustering Ensembles, more user-friendly by automatically tuning its main parameters with the use of metaheuristics. In particular, C^{3}E based on a Squared Loss function, C^{3}E-SL, assumes an optimization procedure that takes as input class membership estimates from existing classifiers, as well as a similarity matrix from a cluster ensemble operating solely on the new target data, to provide a consolidated classification of the target data. To do so, two parameters have to be defined *a priori*, namely: the relative importance of classifier and cluster ensembles and the number of iterations of the algorithm. In some practical applications, these parameters can be optimized via time consuming grid search approaches based on cross-validation procedures. This paper shows that seven metaheuristics for parameter optimization yield classifiers as accurate as those obtained from grid search, but taking half the running time. More precisely, and by assuming a trade-off between user-friendliness and accuracy, experiments performed on twenty real-world datasets suggest that CMA-ES, DE, and SaDE are the best alternatives to optimize the C^{3}E-SL parameters.

Keywords: Classification, Clustering, Ensembles, Metaheuristics, Evolutionary Algorithms

1. Introduction

The combination of multiple classifiers to generate a single classifier has been an active area of research for the last two decades [20,48,85]. For instance, an analytical framework to quantify the improvements in classification results due to combining multiple models has been addressed in [71]. More recently, a survey of classifier ensembles — including applications of them to many difficult real-world problems such as remote sensing, person recognition, one vs. all recognition, and medicine — has been presented in [55]. In brief, several studies in many different areas have empirically shown that from independent, diversified classifiers, the ensemble created is usually more accurate than its individual components [18,64,72,75,78,79]. Analogously, just as ensemble learning has been proved to be more useful compared to single-model solutions for classification problems, many research efforts have indicated that cluster ensembles can improve the quality of results as compared to a single clustering solution — *e.g.*, see [30,37,54,69,70]. In fact, cluster ensembles have been used in a broader manner than classifier ensembles since their motivations include [31]: improvements in the quality of solutions, robust and stable results, selection of models, reuse existing knowledge, different views, and distributed computing. In a broader sense, clustering algorithms have shown to be helpful for a variety of problems [16,32,33,43,44,47,57,63,77].

*Corresponding author. E-mail: luizfsc@icmc.usp.br.

In this paper, we investigate the use of metaheuristics to estimate values for user-defined parameters of an algorithm that combines ensembles of classifiers and clusterers. Most of the motivations for combining ensembles of classifiers and clusterers are similar to those that hold for the standalone use of either classifier ensembles or cluster ensembles. However, some additional nice properties can emerge from such a combination — e.g., unsupervised models can provide a variety of supplementary constraints for classifying new (target) data [8]. From this viewpoint, the usual underlying assumption is that similar new instances in the target set are more likely to share the same class label. Several recent studies have been assuming this by using different techniques [1,21,28,29,60,67,73]. Accordingly, the supplementary constraints provided by the cluster ensemble can be useful for improving the generalization capability of the resulting classifier, specially when labeled data is scarce — thereby being convenient in semi-supervised learning scenarios [84]. Also, unsupervised models can help determine differences between training and target distributions, thus being particularly interesting for applications in which concept drift may take place [2]. Instructive examples inspired in these scenarios will be presented.

Acharya et al. [1] introduced a framework that combines ensembles of classifiers and clusterers to generate a more consolidated classification. In this framework, an ensemble of classifiers is first learned on an initial labeled training dataset. These classifiers are then used to obtain initial estimates of class probability distributions for new, unlabeled (target) data. In addition, a cluster ensemble is applied to the target data to yield a similarity matrix that, by its turn, is used to refine the initial class probability distributions obtained from the classifier ensemble. This framework is materialized through an optimization algorithm named C^{3}E that exploits properties of Bregman divergences[1] in conjunction with Legendre duality to provide a principled and scalable approach [10]. As we discuss afterwards, by using a squared loss function, this algorithm becomes simpler, and requires only two user-defined parameters — as opposed to its more general version in [1], which has three user-defined parameters. The parametric optimization of C^{3}E based on squared loss function is the main contribution of our work[2].

[1] Bregman Divergences include a large number of useful loss functions such as the well-known Squared Loss, KL-divergence, Mahalanobis Distance, and I-Divergence.

[2] A preliminary version of this work appeared in [17].

The remainder of this paper is organized as follows. Section 2 reviews the combination of classifier and cluster ensembles as done by the C^{3}E algorithm, with particular emphasis on its simpler version based on a Squared Loss function (C^{3}E-SL), which was not explicitly studied in [1]. Section 3 provides an overview on metaheuristics for parameter optimization, with particular focus on those used to estimate the C^{3}E-SL parameters, namely: three Evolutionary Algorithms — (1+1)-Evolutionary Strategy (ES) [61,65], Differential Evolution (DE) [58], and Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [35] —, a Self-adaptive DE version (SaDE) [59], a Memetic Computing-based algorithm named Re-sampled Inheritance Search (RIS) [3], and two other nature-inspired algorithms that also perform some kind of stochastic guided search — Simulated Annealing (SA) [46] and Particle Swarm Optimization (PSO) [45]. Section 4 presents experimental results from comparative analysis of the metaheuristics. In Section 5, we finally present our conclusions and suggestions for future work.

2. Combination of classifier and cluster ensembles

In [1], the authors designed a framework that combines classifiers and clusterers to generate a more consolidated classification. This framework, whose core is the C^{3}E algorithm, is depicted in Figure 1. It is assumed that a set of classifiers (consisting of one or more classifiers) have been previously induced from a training set. Such an ensemble of classifiers is employed to estimate initial class probability distributions for every instance $\mathbf{x}_i$ of a target set $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{n}$. These probability distributions are stored as c-dimensional vectors $\{\boldsymbol{\pi}_i\}_{i=1}^{n}$ (c is the number of classes) and will be refined with the help of a cluster ensemble. From this viewpoint, the cluster ensemble provides supplementary constraints for classifying the instances of $\mathcal{X}$, with the rationale that similar instances are more likely to share the same class label. In order to capture similarities between the instances of $\mathcal{X}$, C^{3}E takes as input a similarity (co-association) matrix $\mathbf{S}$, where each entry corresponds to the relative co-occurrence of two instances in the same cluster [31,69] — considering all the data partitions that form the cluster ensemble built from $\mathcal{X}$.

To sum up, C^{3}E receives as inputs a set of vectors $\{\boldsymbol{\pi}_i\}_{i=1}^{n}$ and a similarity matrix, $\mathbf{S}$, and outputs a consolidated classification for every instance in $\mathcal{X}$ — represented by a set of vectors $\{\mathbf{y}_i\}_{i=1}^{n}$, where

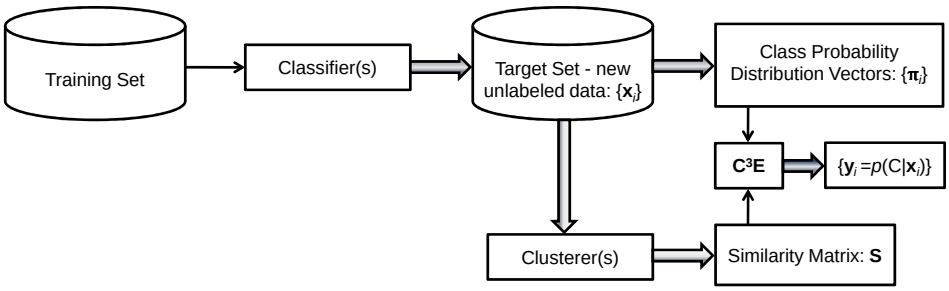

Fig. 1. Framework to combine ensembles of classifiers and clusterers using C^{3}E algorithm [1].

$\mathbf{y}_i = p(C \mid \mathbf{x}_i)$ — i.e., $\mathbf{y}_i$ is the estimated posterior class probability assignment for every instance in $\mathcal{X}$. To do so, C^{3}E solves an optimization problem whose objective is to minimize J in (1) with respect to the set of probability vectors $\{\mathbf{y}_i\}_{i=1}^n$:

$$J = \sum_{i \in \mathcal{X}} \mathcal{L}(\mathbf{y}_i, \boldsymbol{\pi}_i) + \alpha \sum_{(i,j) \in \mathcal{X}} s_{ij} \mathcal{L}(\mathbf{y}_i, \mathbf{y}_j) \quad (1)$$

The quantity $\mathcal{L}(\cdot,\cdot)$ denotes a loss function. Informally, the first term in Equation (1) captures dissimilarities between the class probabilities provided by the ensemble of classifiers and the output vectors $\{\mathbf{y}_i\}_{i=1}^n$. The second term encodes the cumulative weighted dissimilarity between all possible pairs $(\mathbf{y}_i, \mathbf{y}_j)$. The weights to these pairs are assigned in proportion to the similarity values $s_{ij} \in [0,1]$ of matrix $\mathbf{S}$, and the coefficient $\alpha \in \mathbb{R}_+$ controls the relative importance of classifier and cluster ensembles.

The C^{3}E algorithm, as proposed in [1], exploits general properties of a large class of loss functions, described by Bregman divergences [7], in conjunction with Legendre duality and a notion of variable splitting that is also used in alternating direction method of multipliers [10] to yield a principled and scalable solution. If one chooses a squared loss function, the C^{3}E algorithm gets simpler. In particular, the variable splitting approach, in which two copies of $\mathbf{y}_i$ are updated iteratively, is not needed anymore. As such, we can get rid of one of the user-defined parameters of the algorithm, λ, which is an optimization constraint to ensure that the two copies of the variables remain close during the optimization process. Also, the update equations are still available in closed form solution for each $\mathbf{y}_i$, as explained next.

2.1. C^{3}E with squared loss function

Choosing the Squared Loss (SL) function as the Bregman divergence in the optimization problem formulated in Equation (1) we obtain:

$$\begin{aligned} J_{SL} = & \frac{1}{2} \sum_{i \in \mathcal{X}} \|\mathbf{y}_i - \boldsymbol{\pi}_i\|^2 \\ & + \alpha \frac{1}{2} \sum_{(i,j) \in \mathcal{X}} s_{ij} \|\mathbf{y}_i - \mathbf{y}_j\|^2. \end{aligned} \quad (2)$$

The second term of J_{SL} shows that the variables are coupled. In order to circumvent this difficulty, and following an approach analogous to the one adopted for the more general case [1] — where any Bregman Divergence can be used — we can design an iterative update procedure of the variables to be optimized. In particular, keeping $\{\mathbf{y}_j\}_{j=1}^n \setminus \{\mathbf{y}_i\}$ fixed, we can minimize J_{SL} in Equation (2) for every $\mathbf{y}_i$ by setting:

$$\frac{\partial J_{SL}}{\partial \mathbf{y}_i} = \mathbf{0}. \quad (3)$$

Considering that the similarity matrix $\mathbf{S}$ is symmetric, and noting that $\frac{\partial \|\mathbf{x}\|2}{\partial \mathbf{x}} = 2\mathbf{x}$, we get:

$$\mathbf{y}_i = \frac{\boldsymbol{\pi}_i + \alpha' \sum_{j \neq i} s_{ij} \mathbf{y}_j}{1 + \alpha' \sum_{j \neq i} s_{ij}}, \quad (4)$$

where we set $\alpha' = 2\alpha$ for mathematical convenience. Equation (4) can be computed iteratively by using Algorithm 1, which summarizes the main steps of C^{3}E with Squared Loss function (C^{3}E-SL). Since each up-

date of $\mathbf{y}_i$ reduces J_{SL}, which is bounded from below by zero, the algorithm converges. A stopping criterion can be defined as either the maximum number of iterations, I, or a predefined threshold on the difference between values of objective function in (2) computed from two consecutive iterations of the algorithm.

The asymptotic time complexity of Algorithm 1 is $O(c \cdot n^2)$, where c is the number of class labels and n is the number of instances in the target set. Note that n, the number of new instances to be classified, is usually much less than the number of instances in the training set — used to build the classifier ensemble. Furthermore, as in the more general case addressed in [1], the resulting minimization procedure can be performed in parallel — by updating one or more variables per processor.

In practice, the choice of the Bregman divergence is dependent on the application domain. At this point, however, we can anticipate that our empirical results are very similar to those found in [1] across a variety of datasets, thereby suggesting that, as usual, simpler approaches should be tried first. Next, we make room for instructive examples that explore some of the capabilities of the C^{3}E-SL algorithm.

Algorithm 1: C^{3}E with Squared Loss (C^{3}E-SL)

Input: $\{\boldsymbol{\pi}_i\}, \mathbf{S}, \alpha$.
Output: $\{\mathbf{y}_i\}$.
1 Initialize $\{\mathbf{y}_i\}$ such that $y_{i\ell} = \frac{1}{c}$ $\forall i \in \{1, 2, ..., n\}, \forall \ell \in \{1, 2, ..., c\}$;
2 **Repeat**
3 Update $\mathbf{y}_i$ using Equation (4) $\forall i \in \{1, 2, ..., n\}$;
4 **until** *convergence*;

2.2. *Pedagogical examples*

Figure 2 shows a partition with 60 objects used as a training set by a decision tree. The supervised model induced encompasses the three known classes: star-green, circle-red, and square-blue. The decision boundaries are represented by dashed lines. As expected, the resulting classification accuracy is 100%. On the other hand, Figure 3 presents the target set to be classified. These new objects, compared to those that were used to train the classifier (which are the largest, darkest objects included for convenience in this figure), have a slightly different distribution, which causes misclassification. As a result, in this target set, the classification accuracy of the trained decision tree is 77.1%. However, by using C^{3}E-SL to refine these results — with a similarity matrix built from two cluster partitions, being one with 5 clusters and another with 10 clusters — the classification accuracy increases to 99.2% (with $\alpha = 0.3$ and $I = 15$). This suggests that the information provided by clusterers allows to design learning methods that are aware of the possible changes in the data distribution [2].

Figure 4, by its turn, presents a two-dimensional derivation of the well-known Iris dataset [26] — x_1 and x_2 are respectively the (approximate) sepal and petal areas. The three classes (Setosa, Versicolor and Virginica) are represented by circle-blue, star-green, and square-red symbols, respectively. A decision tree

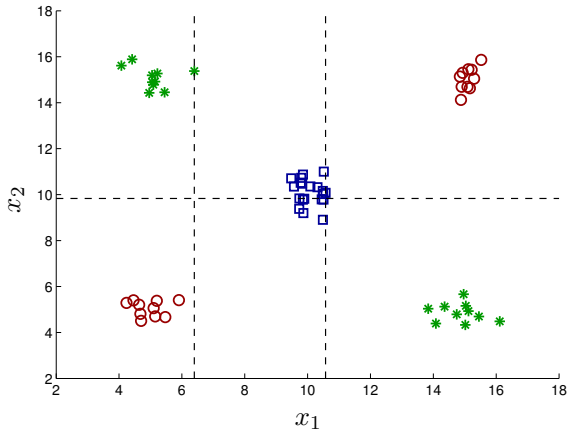

Fig. 2. Training set with 60 objects used to induce a decision tree to recognize three balanced classes — dashed lines represent the decision boundaries, which provide an accuracy of 100%.

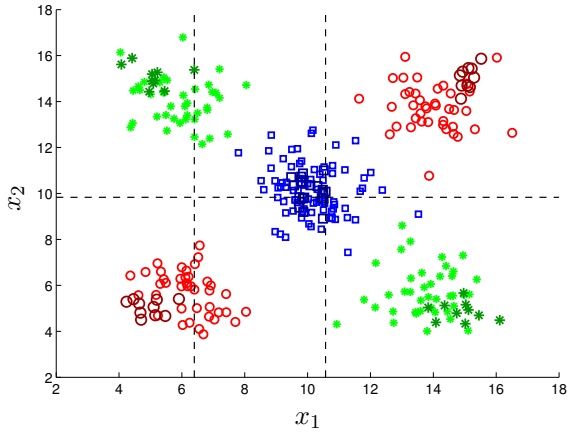

Fig. 3. Target set, with some change in the distribution of the data, extrapolating the decision boundaries. Decision tree obtained a accuracy of 77.1%, while C^{3}E-SL achieved 99.2%.

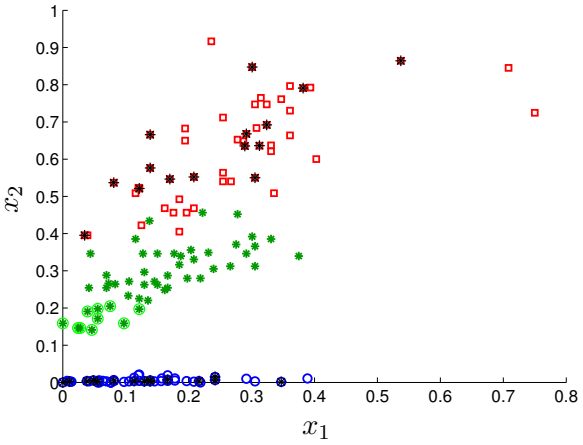

Fig. 4. A decision tree was induced on Iris 2D dataset considering only objects of Setosa and Virginica classes (which are represented as a black star).

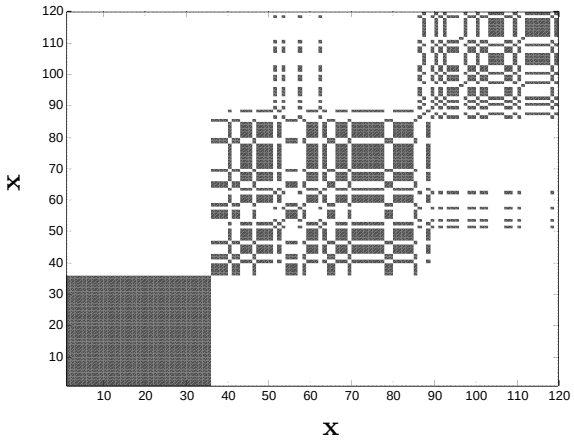

Fig. 5. The similarity matrix was able to capture the existence of three clusters and helped to refine the probability distributions of the classes provided by the decision tree.

was induced by considering 30 balanced objects, encompassing only the classes Setosa and Virginica — these objects are represented as black stars altogether with their respective symbols. As expected, the model classifies star-green objects as being square-red ones (i.e., all Versicolor flowers are classified as Virginica), since they are in overlapping regions, and star-green objects were not used to build the classification model. By refining these classifications with C^{3}E-SL (with $\alpha = 0.5$ and $I = 15$) the 10 star-green (circled) objects nearest to the circle-blue ones became the most uncertain with respect to which class they belong to. This observation can be clarified by taking into account the similarity matrix illustrated in Figure 5, which shows that three clusters were detected by the cluster ensemble. More specifically, the classes Versicolor and Virginica were (in parts) correctly separated in two clusters by the unsupervised model. This information is useful to indicate that the star-green objects may not belong to the same class of the square-red ones. In other words, the resulting classification can be (in some level) influenced by the cluster constraints, coded into the similarity matrix, thereby helping to improve the generalization capability — specially when labeled data is scarce and/or poorly sampled [84].

3. Metaheuristics for parameter optimization

Metaheuristics consist of stochastic algorithms with randomization and local search that are used for global optimization, including evolutionary-based algorithms [5,6], and also those based on thermodynamic principles [46], swarm intelligence [45], and memetic computing [14,15]. In general, metaheuristics have two main characteristics: diversification and intensification [80]. Diversification involves exploring the search space on the global scale by generating diverse random solutions, while intensification focuses on the search in a specific region of the search space by exploiting good solutions that are found in it. Several studies have shown that metaheuristics are indeed capable to find acceptable solutions for complex (optimization) problems in a reasonable time, preventing entrapment in local minima [4,23,24,27]. From this perspective, and observing that in [1] a systematic approach for parameter optimization was not used, we investigate the performance of a variety of metaheuristics to optimize the C^{3}E-SL parameters (α and I). These metaheuristics are briefly addressed next.

3.1. Conceptual overview

Over the last decades, a wide variety of nature-inspired metaheuristics have been developed. For example, genetic and evolutionary approaches have been used in image and data analysis [13,51,52,56,62], as well as to design wireless networks [53], distribution processes [42], and safety buildings [36]. Similarly, swarm intelligence techniques have been employed to classify motor imagery electroencephalogram signals [39] and to optimize paths for evacuation routes [25] and concrete transportation systems [83]. Recently, considerable efforts have been invested in algorithms that can adapt their own control parameters [11,23,50,59]. Typically, such algorithms reach better results for that they are less sensitive to parameter vari-

ations. From this, we have studied seven metaheuristics to optimize the C^{3}E-SL parameters:

(1+1)-ES: this metaheuristic is an Evolution Strategy [61,65], in which evolution is a process based on two individuals, which are candidate solutions to a given problem. In brief, from one parent (a real-valued vector) a descendant is created by means of adding normally distributed random numbers [5,6]. Then, the best of both individuals serves as the ancestor for the following generation, thus contributing to evolve better solutions. Typically, two parameters (c_d and c_i), which are based on the $1/5$-success rule of Rechenberg [61], must be provided to adjust the convergence rate. Schwefel [65] suggests using $c_d = 0.82$ and $c_i = 1.22$[3].

CMA-ES: this metaheuristic is an extension of Evolution Strategies. CMA-ES [35] — that stands for Covariance Matrix Adaptation Evolution Strategy — adapts the covariance matrix of a multi-variate normal search distribution. This adaptation consists of leaning dependencies among the parameters. The only user-defined parameter of CMA-ES is the population size. As for (1+1)-ES, by following previous studies [34,35] we used 20 individuals in our work.

Differential Evolution (DE): as (1+1)-ES and CMA-ES, DE is also an evolutionary algorithm for parameter optimization [58]. DE aims at solving an optimization problem by maintaining a population of candidate solutions (parameter vectors). From this population, new candidate solutions are created by means of perturbations (the so-called mutations) of existing solutions. In particular, consider a given parameter vector. After being mutated, the parameters of this vector are mixed with the parameters of another predetermined (target) vector to provide the so-called trial vector. If the trial vector encodes a better solution than the target vector, this gets replaced by the trial vector in the next generation. DE can be materialized by means of a variety of strategies — we used $DE/best/2/bin$ with population size $P = 20$, the scaling constant $F = 0.5$, and the crossover constant $CR = 0.9$ [58,68,74].

Simulated Annealing (SA): inspired by thermodynamic processes, SA [46,49] starts its search from an initial random solution (s) and, in each iteration, a "neighbor" s' of the current solution is randomly generated. Differences in the fitness function are captured by a parameter, Δ. In minimization problems, if $\Delta < 0$, then s' becomes the new current solution, otherwise s' is accepted with a probability $e^{-\Delta/T}$, where T is the so-called temperature, a parameter that is decreased over the iterations. This process is repeated until T is so small that no solution can be accepted anymore. In each iteration, the temperature is updated by doing $T' = r \cdot T$, where r is the cooling rate. In our study, as suggested by [46], we used $r = 0.95$.

Particle Swarm Optimization (PSO): based on swarm intelligence [22,45], PSO starts the search from a population of randomly generated candidate solutions, which are named particles. These particles are moved around in the search-space by taking into account their positions and velocities. In particular, the velocity of the particle is updated via information of the best position found by itself and the best position found so far (considering all particles). Following the general guidelines from [66], we adopted $V_\alpha = 0.3$, and $V_I = 15$. Also, we used a population formed by 20 particles and we set both the *cognition* and *social* quantities to 2. The inertia weight, w, was set to linearly decrease from 0.9 to 0.4 during the first $1{,}500$ iterations.

SaDE: is a Self-adaptive DE that samples random values for F and Cr from normal distributions $\mathcal{N}(0.5, 0.3)$ and $\mathcal{N}(CRm, 0.1)$, respectively. Initially, CRm is set to 0.5, but it is adapted at every 25 generations [59]. Similar adaptation occurs at every 50 generations to select a trial vector generation strategy (which can be "DE/rand/1/bin" or "DE/current-to-best/1/bin"). Although SaDE has a handful of preset parameters, it tends to be less sensitive to them [59]. We have used a population with 20 individuals.

RIS: consists of a memetic algorithm that gradually perturbs a single solution by means of two components [14]: a deterministic local search that exploits within the neighbourhood of the solution, and a stochastic sampling that produces random solutions to be recombined (inheritance). This recombination is based on the DE exponential crossover. The required parameters were defined as in [14]: inheritance factor $\alpha_e = 0.5$, the initial search radius $\rho = 0.4$, and the stop-threshold $\varepsilon = 10^{-6}$.

3.2. Optimizing the C^{3}E-SL parameters

In this paper, we studied how to automatically optimize the C^{3}E-SL parameters, namely the coefficient

[3] At this point of our research, we are not particularly interested in fine-tuning parameter values of the metaheuristics themselves. Instead, we are taking advantage of recommendations made in previous studies already available in the literature. This approach has also been adopted to choose parameter values for the other metaheuristics here employed.

α, which controls the relative importance of classifier and cluster ensembles, and the number of iterations (I) of the algorithm. As expected, this is a difficult (multimodal) optimization problem — as Figure 6 illustrates for the *Wine Red Quality* dataset [26]. We are particularly interested in metaheuristics capable of escaping from local optima, hopefully being able to reach the global optimum — or at least a good local optimum solution — thereby getting better classifiers. In a similar vein, the authors in [3] used PSO to optimize an Enhanced Probabilistic Neural Network (EPNN), which takes advantage of local information and non-homogeneity existing in the training data to refine a probabilistic neural network.

The metaheuristics addressed in Section 3.1 typically find good solutions for many types of problems [4,24,27]. As the *No Free Lunch* theorem indirectly suggests [40,81,82], by focusing in a particular application, with representative (real-world) problems, a more fruitful evaluation may take place. In other words, although it is very difficult to find the best metaheuristic for a given problem, we can explore reasonable ones, whose specialization matches (at some level) to the specific problem under consideration. For this reason, in practice the suitability of any metaheuristic for a particular application scenario is an empirical matter.

In the next section, we present experimental results that show that the chosen metaheuristics can efficiently optimize the C^{3}E-SL parameters, thereby providing good classifiers for a variety of datasets.

4. Empirical evaluation

The optimization of the parameters of the C^{3}E-SL algorithm (described in Section 2.1) consists of searching for values of α and I that yield to best classification accuracy. In other words, ideally we are looking for an optimal pair of values $<\alpha^*, I^*>$ that results in the most accurate classifier for a given problem. In order to estimate these values, we employ the metaheuristics addressed in Section 3.1. The details of our experimental setup, followed by the achieved empirical results, are presented next.

4.1. Experimental setup

Twenty real-world datasets from the UCI Machine Learning Repository [26] were used in the experiments. The main characteristics of these datasets are reported in Table 1.

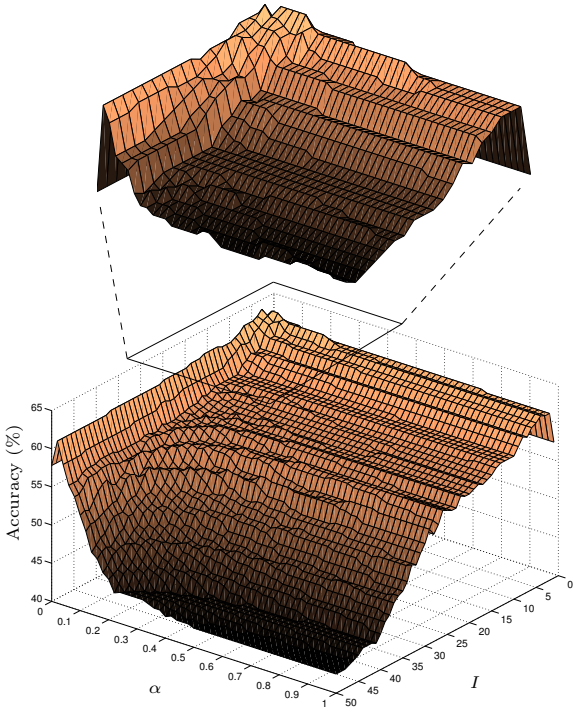

Fig. 6. C^{3}E-SL classification accuracy (%) with respect to different combinations of values for the parameters α and I.

Table 1
Main characteristics of the used datasets.

ID	Dataset	Inst.	Attrib.	Classes
Wisc	Breast Cancer Wisconsin (Original)	683	9	2
Pima	Pima Indians Diabetes	768	8	2
Yeast	Yeast	205	20	4
Spam	Spambase	4601	57	2
Iono	Ionosphere	351	33	2
Iris	Iris	150	4	3
Wred	Wine Red Quality	1599	12	6
Blood	Blood Transfusion Service Center	748	4	2
Glass	Glass Identification	214	9	7
Sonar	Con. Bench (Sonar, Mines vs. Rocks)	208	60	2
Vert	Vertebral Column	310	6	3
Seeds	Seeds	210	7	3
Ecoli	Ecoli	336	7	8
ILPD	Indian Liver Patient Dataset	579	10	2
LungC	Lung Cancer	27	52	3
Soybean	Soybean (Small)	47	35	4
Heart	Statlog Heart	270	13	2
WDBC	Breast Cancer Wisconsin (Diag.)	569	30	2
Derma	Dermatology	358	33	6
Contra	Contraceptive Method Choice	1473	9	3

Essentially, the optimal values for the parameters of C^{3}E-SL — $<\alpha^*, I^*>$ — can be estimated via cross-validation procedures, in which we usually have three datasets: training, validation, and test. In a controlled experimental setting, the new data (target set) is frequently referred to either test or validation set. These two terms have been used interchangeably in the literature, sometimes causing confusion. In this paper, it is assumed that the target/test set has not been used at all in the process of building the ensemble of classifiers — *i.e.*, it is an independent set of data not used at all to optimize any parameter of the algorithm and, as a consequence, of the resulting classifier. Thus, as usual [76], only the training and validation sets are used to optimize the parameters of the algorithm.

Figure 7 portrays the two main steps of the adopted optimization procedure. First, we split the instances into training, validation, and target/test sets. With the training set, the ensemble of classifiers is built. Then, we use metaheuristics to estimate α^* and I^* by using the validation set, where a cluster ensemble is induced. The resulting values for the parameters are finally employed to assess the classification accuracy in the target/test set where, again and as requested by C^{3}E, a cluster ensemble similar to the one built for the validation set must be induced. Provided that, in a controlled experimental setting, we do know the true class labels of all instances, we can repeat this process multiple times and compute statistics of interest from the target/test set. Thus, we used the cross-validation procedure for empirically estimating the generalization capability of the C^{3}E-SL algorithm (as further discussed next).

A straightforward (but often computationally intensive) way of searching for $<\alpha^*, I^*>$ involves running grid search [9], which has been adopted as a baseline

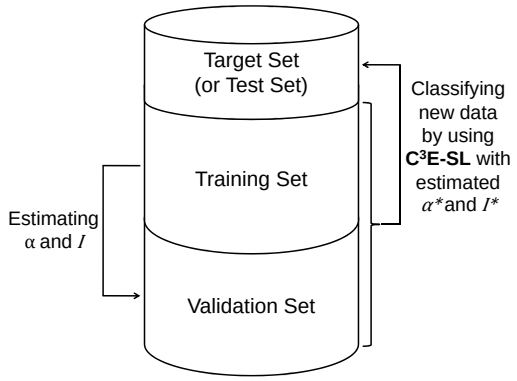

Fig. 7. Optimizing C^{3}E-SL parameters via cross-validation.

for comparison purposes in our work. Grid search usually refers to exhaustive searching through a subset of the parameter space. In our case, one of the parameters, the number of iterations (I), is naturally discrete, whereas the other parameter (α) is real-valued and thus needs to be discretized before running grid search. For both of them, lower and upper bounds must be set *a priori*. In particular, we used all different combinations of $\alpha = \{0,0.001,\ 0.002,...,1\}$ and $I = \{1,2,3,...,50\}$. From this setting, C^{3}E-SL was run $50{,}050$ times to estimate the optimal pair of values $<\alpha^*, I^*>$ for each dataset, according to (5):

$$<\alpha^*, I^*> = \underset{\alpha, I}{\textit{arg min}}\ \text{CErr}\left[\text{C}^3\text{E-SL}(<\alpha, I>)\right], \quad (5)$$

where CErr is the fitness function that returns C^{3}E-SL classification errors for different values of α and I. Although this is a two-dimensional parameter optimization, specialized algorithms are necessary to find good solutions in less time. Metaheuristics guided to minimize the misclassification rate in (5) were run from a time constraint given by grid search. In particular, for each dataset the running times spent by grid search were stored and then *50*% of these running times were given as a time limit for the metaheuristics[4]. Algorithm 2 provides an overview of the framework used for every metaheuristic. Accuracy results obtained from metaheuristics were then compared to those found via grid search. By doing that, we can evaluate the trade-off between running time and classification accuracy provided by metaheuristics. It is worth emphasizing that, as will be addressed in Section 4.2, C^{3}E-SL optimized by means of the studied metaheuristics reaches competitive accuracies in less running time, and this is the main benefit of our work.

We adopted a 5-fold cross-validation process [76], in which each fold has 20% of the dataset instances — this is precisely the size of each target/test set. Accordingly, both training and validation sets have 40% of the dataset instances. The classifier ensemble was composed of two well-known classifiers, namely: Naive Bayes (NB) and Decision Tree (DT) [76]. The similarity matrix $\mathbf{S}$ was constructed from a cluster ensemble

[4]All metaheuristics under study were implemented in Matlab, using only the necessary commands. Population initialization was made within of a common region (of α and I). This way, more uniform efficiency comparisons can be performed. The same computer (i7, 3.2 GHz, 12 Gb RAM), running only the operational system, was used for all the controlled experiments.

Algorithm 2: Optimizing C^{3}E-SL

Input: MH ← choose a metaheuristic.
Output: estimated pair of values $<\alpha^*, I^*>$.

1 **Run** MH:
2 Randomly set initial solution(s) for $<\alpha, I>$ such that $\alpha = \{0, 0.001, ..., 1\}$ and $I = \{1, 2, ..., 50\}$ — a population containing a certain number of individuals (solutions) is initialized;
3 Run C^{3}E-SL (Algorithm 1) for each individual of the population (each one using its coded solution $<\alpha, I>$ as input for the C^{3}E-SL) — the misclassification rate, as in Equation (5), is used to determine the goodness of the individuals;
4 Apply the MH operators (for details see Section 3.1);
5 Go to Step 3;
6 **until** *time limit is reached*;
7 Select the best individual (which provides the minimal misclassification rate) to be the pair of values $<\alpha^*, I^*>$;

based on data partitions found by the K-Means clustering algorithm [41]: one with k clusters and another with $2k$ clusters — where the number of clusters, k, was automatically estimated from data by using the relative validity clustering criterion known as Simplified Silhouette [12,38].

4.2. Experimental results

Before reporting the experimental results achieved by the used metaheuristics, it is instructive to compare the results obtained by C^{3}E based on Squared Loss (SL) with C^{3}E based on I-Divergence [1]. For both variants of C^{3}E, grid search was performed to optimize their parameters — α and I (as addressed in Section 4.1). Table 2 shows the average classification accuracies, along with the results obtained for the classifier ensembles. In most of the cases, both variants of C^{3}E (using grid search) provided better classification accuracies compared to the classifier ensemble. Also, the overall accuracies of C^{3}E-SL have shown to be similar to those got by C^{3}E based on I-Divergence, which is more complex.

Table 3 reports the results obtained by the studied metaheuristics. In addition, the results obtained from grid search — already reported in Table 2 — are again reproduced in the second column of Table 3 for convenience, thus making easier to compare the whole set of experimental results. In some datasets the computationally intensive grid search procedure provided the best results. However, note that with half the running time of grid search, better accuracies were obtained by RIS in 20% of the datasets, by SA in 45% of the datasets, by (1+1)-ES, CMA-ES, and PSO in 50% of the datasets and by DE and SaDE in 55% of the datasets. Also, observe that there are some ties. Considering only the results achieved by the metaheuristics, one can note that RIS obtained the best accuracy only in one dataset, CMA-ES and PSO in two datasets, DE in three, SA in four, and (1+1)-ES and SaDE performed best in five and six datasets, respectively. Although the performances of the different metaheuristics are comparable across the assessed datasets, a more careful observation suggests that DE, SaDE, (1+1)-ES, CMA-ES, and PSO have provided more consistent results compared to grid search. More specifically, DE, CMA-ES, and PSO reached better accuracies, whereas SaDE and CMA-ES showed to be more stable (see overall average in Table 3). By taking into account this observation, one may prefer to use CMA-ES or SaDE (because they are less dependent on parameters) or DE (due to implementation simplicity).

Table 2
Accuracies (%) of Classifier Ensemble (NB+DT), C^{3}E based on I-Divergence (ID) and C^{3}E based on Squared Loss (SL) — standard deviations appear within parentheses. For convenience, best results are highlighted in bold face.

Dataset	Class. Ens.	C^{3}E-ID [a]	C^{3}E-SL
Wisc	95.60 (2.8)	**96.48** (1.2)	**96.48** (1.2)
Pima	76.18 (3.1)	**77.09** (3.6)	75.92 (2.7)
Yeast	95.61 (3.2)	96.59 (2.8)	**97.56** (1.7)
Spam	81.20 (1.2)	92.20 (1.2)	**92.81** (1.1)
Iono	**88.34** (4.3)	85.20 (11.2)	87.75 (7.1)
Iris	95.33 (3.0)	**96.00** (2.8)	**96.00** (2.8)
Wred	55.91 (4.5)	**57.60** (8.7)	57.48 (9.1)
Blood	75.80 (1.9)	76.47 (1.5)	**76.60** (1.7)
Glass	**64.97** (6.7)	62.65 (9.9)	**64.97** (6.7)
Sonar	**73.60** (8.7)	71.20 (7.6)	67.38 (9.4)
Vert	**82.26** (5.9)	59.33 (15.2)	75.16 (16.1)
Seeds	88.57 (7.4)	90.00 (3.1)	**91.90** (3.2)
Ecoli	**84.83** (3.8)	**84.83** (4.5)	84.53 (4.8)
ILPD	62.69 (3.4)	**66.84** (9.4)	60.98 (8.2)
LungC	**52.68** (10.9)	**52.68** (10.9)	**52.68** (10.9)
Soybean	**80.92** (6.6)	**80.92** (6.6)	**80.92** (6.6)
Heart	80.74 (2.6)	**81.85** (2.3)	81.57 (1.7)
WDBC	**93.50** (0.8)	92.97 (0.7)	93.41 (0.5)
Derma	**95.25** (1.2)	94.49 (1.8)	90.92 (6.0)
Contra	**51.26** (1.0)	50.00 (1.5)	50.90 (1.0)

[a] Following [1] we adopted $\lambda = 1$.

Table 3
Accuracies (%) of C^{3}E-SL with parameters optimized via metaheuristics — standard deviations appear within parentheses. The best results among all metaheuristics are highlighted in bold face. Recall that metaheuristics were given just half the running time of the grid search.

Dataset	Grid Search	(1+1)-ES	CMA-ES	DE	SA	PSO	SaDE	RIS
Wisc	96.48 (1.2)	96.48 (1.2)	96.04 (1.1)	96.77 (1.5)	95.90 (0.8)	96.19 (1.0)	**96.85 (0.5)**	96.19 (0.5)
Pima	75.92 (2.7)	74.23 (2.9)	74.74 (2.8)	75.00 (2.6)	**75.53 (3.0)**	75.39 (2.6)	74.15 (1.5)	70.90 (2.4)
Yeast	97.56 (1.7)	97.07 (2.7)	97.07 (2.0)	**97.56 (1.7)**	97.07 (2.7)	96.59 (2.8)	96.59 (2.7)	96.59 (2.7)
Spam	92.81 (1.1)	92.72 (1.1)	92.72 (1.1)	92.72 (1.1)	92.74 (1.1)	**92.76 (1.1)**	91.49 (0.6)	91.09 (0.7)
Iono	87.75 (7.1)	89.20 (5.3)	89.20 (5.3)	89.20 (5.3)	88.62 (4.4)	**89.48 (5.1)**	87.39 (4.6)	76.91 (11.8)
Iris	96.00 (2.8)	90.00 (13.3)	95.33 (3.0)	95.33 (3.0)	**96.00 (2.8)**	95.33 (3.0)	**96.00 (2.8)**	94.17 (2.8)
Wred	57.48 (9.1)	**55.85 (10.1)**	55.35 (10.1)	55.41 (9.8)	55.29 (9.7)	52.16 (11.1)	43.62 (6.6)	44.50 (4.4)
Blood	76.60 (1.7)	76.47 (0.9)	76.07 (2.2)	75.80 (1.9)	**76.87 (1.0)**	76.07 (2.2)	**76.87 (0.7)**	76.40 (0.3)
Glass	64.97 (6.7)	57.53 (9.0)	**64.97 (6.7)**	**64.97 (6.7)**	57.53 (9.0)	62.65 (9.9)	51.64 (9.2)	51.76 (7.4)
Sonar	67.38 (9.4)	66.34 (11.3)	68.26 (11.3)	66.83 (13.0)	64.91 (11.6)	68.26 (11.3)	**68.75 (7.3)**	62.85 (8.2)
Vert	75.16 (16.1)	74.84 (15.9)	74.84 (15.9)	74.84 (15.9)	75.48 (16.0)	74.84 (15.9)	**78.95 (1.9)**	73.06 (5.2)
Seeds	91.90 (3.2)	87.14 (6.4)	89.52 (2.1)	89.05 (2.1)	86.67 (6.2)	88.57 (7.4)	**91.43 (2.7)**	90.00 (1.9)
Ecoli	84.53 (4.8)	**85.43 (5.4)**	84.83 (4.5)	84.24 (5.3)	84.83 (4.5)	84.53 (4.8)	83.41 (2.6)	81.18 (2.1)
ILPD	60.98 (8.2)	62.01 (9.7)	59.07 (9.6)	61.32 (8.6)	60.46 (8.5)	62.18 (8.4)	64.12 (6.2)	**64.42 (6.6)**
LungC	52.68 (10.9)	**60.13 (8.5)**	61.04 (7.3)	59.22 (10.0)	58.31 (6.5)	57.36 (6.3)	58.35 (7.7)	55.54 (12.3)
Soybean	80.92 (6.6)	89.90 (5.7)	86.74 (4.9)	87.27 (5.0)	**90.46 (6.1)**	89.40 (7.7)	87.27 (7.7)	89.90 (4.7)
Heart	81.57 (1.7)	**82.87 (2.3)**	82.13 (1.5)	82.41 (2.6)	81.76 (1.3)	82.31 (2.4)	82.41 (2.2)	82.69 (1.7)
WDBC	93.41 (0.5)	93.89 (0.6)	**93.98 (0.5)**	93.94 (0.6)	93.01 (0.4)	93.89 (0.5)	93.94 (0.6)	92.84 (0.4)
Derma	90.92 (6.0)	94.97 (1.4)	95.46 (1.2)	**95.67 (0.9)**	94.27 (1.2)	95.39 (1.1)	95.25 (1.4)	87.02 (10.1)
Contra	50.90 (1.0)	**51.65 (1.6)**	51.54 (1.1)	51.53 (1.0)	50.68 (1.1)	51.56 (1.1)	50.61 (1.2)	48.44 (0.2)
Overall Average	78.80 (5.1)	78.94 (5.8)	79.45 (4.7)	79.45 (4.9)	78.82 (4.9)	79.25 (5.3)	78.45 (3.5)	76.32 (4.3)
Win/Tie (w.r.t. Grid Search)		10	10	11	9	10	11	4

In practice, choosing an algorithm (among several alternative ones) for a particular application is usually not an easy task. For the problem being tackled, convenience aspects of the metaheuristics could be considered in order to help us to decide which ones are the most suitable ones. In this sense, we would like to answer the following question: Which of the studied metaheuristics is the preferred one for optimizing the C^{3}E-SL parameters? To do so, we took into account two aspects, namely: how *user-friendly* a particular metaheuristic is and its accuracy (recall that the running time has been fixed for all of them). At first glance, one can argue that "user-friendly" is a rather subjective criterion. For example, it could be related to the number of parameters that each metaheuristic has — actually, the more user-defined parameters, the less user-friendly. However, some metaheuristics are naturally more robust to parameter variations than others. As it can be noted from Section 3.1, PSO and SaDE have a handful of parameters, but the latter is recognized as being less sensitive to parameter variations [59]. In fact, SaDE (such as CMA-ES) adapts its control parameters during the search process, being considered a more modern and user-friendly alternative. In this context, we can state that PSO is possibly the less indicated metaheuristic for our problem. The other metaheuristics — (1+1)-ES, DE, SA, and RIS — have few parameters to be set and are subject to the same issues discussed above.

Considering the classification accuracy, the metaheuristics were ranked. Table 4 presents such ranks, which were generated according to Friedman test from the classification accuracies reported in Table 3. The Friedman test is a non-parametric counterpart of the well-known ANOVA. Essentially, tests of statistical significance are reported because they provide some reassurance about the validity and non-randomness of the obtained results. More precisely, we carried out statistical tests by following the approach described in [19]. In short, the adopted approach is aimed at comparing multiple algorithms on multiple datasets by using the Friedman test with a corresponding post-hoc test. Analyses of statistical significance showed that there is no statistical significant difference when comparing the results obtained for grid search to the metaheuristics — except for RIS (with $\alpha = 5\%$). From these analyses, one could claim that (1+1)-ES, CMA-ES, DE, SA, PSO, and SaDE can provide clas-

Table 4
Friedman test ranking produced from the average accuracies reported in Table 3.

Rank	Accuracy
1	DE
2	(1+1)-ES
3	CMA-ES
4	PSO
5	SaDE
6	SA
7	RIS

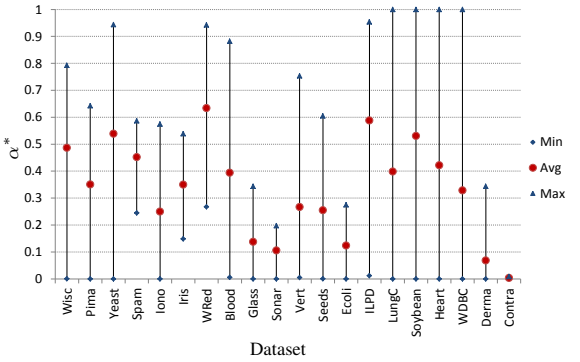

Fig. 8. Minimum, maximum, and average values for α^* (CMA-ES).

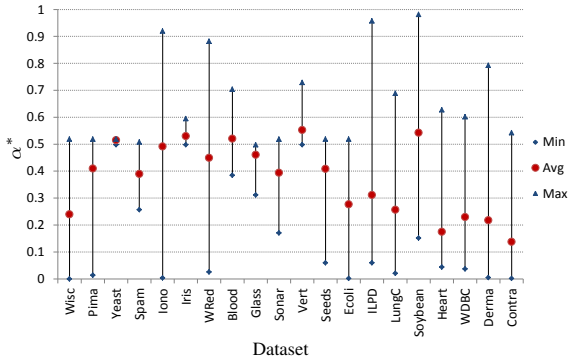

Fig. 9. Minimum, maximum, and average values for α^* (DE).

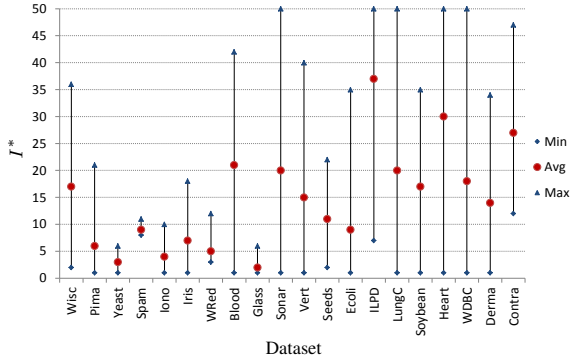

Fig. 10. Minimum, maximum, and average values for I^* (CMA-ES).

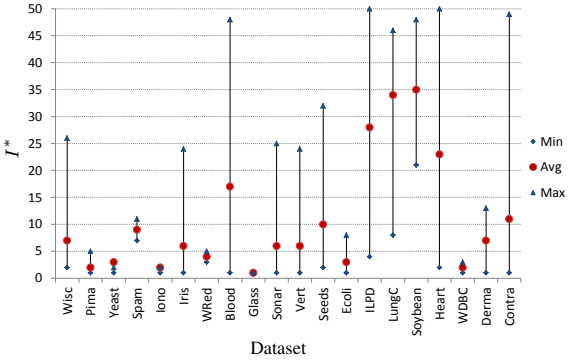

Fig. 11. Minimum, maximum, and average values for I^* (DE).

sifiers as accurate as those obtained from grid search, but taking half the running time. This is particularly relevant for real-world data mining applications where large datasets are available.

DE has shown to be the best metaheuristic according to the accuracy criterion explored in Table 4. However, it is somehow expected that there should be a trade-off between user-friendliness and accuracy. In this sense, SaDE and CMA-ES may be more indicated — but it is worth observing that DE requires setting up only two control parameters[5]. Figures 8–11 present the minimum, average, and maximum estimated values for α^* and I^* achieved with CMA-ES and DE from the cross-validation procedure. One can observe that there is a high variance both for different datasets and for different folds of the same dataset, thereby suggesting that the optimization of the C^{3}E-SL parameters is highly data dependent. We also noticed that C^{3}E-SL, with parameters optimized by CMA-ES, provided equal to or better results than those achieved by classifier ensembles in 75% of the datasets — compared to 70% and 60% for DE and SaDE, respectively. Thus, note that C^{3}E-SL with parameter optimization by means of CMA-ES, DE, or SaDE can yield to better results than classifier ensembles in less running time than grid search approaches.

[5]The population size and the number of generations required for convergence are somewhat related. From a practical viewpoint, one may expect that the more genotypes the less the required generations for convergence.

5. Conclusions

In this paper we studied how to make an existing algorithm, named C^{3}E (from Consensus between Classification and Clustering Ensembles), more convenient by automatically tuning its main parameters — the relative importance of classifier and cluster ensembles (α) and the number of iterations of the algorithm (I) — with the use of metaheuristics. Although, as expected, these parameters can be straightforwardly optimized via (time consuming) grid search approaches (based on well-known cross-validation procedures), we empirically showed that metaheuristics can be more computationally efficient alternatives. More precisely, anal-

yses of statistical significance conducted from experiments performed on twenty datasets show that six metaheuristics — (1+1)-ES, CMA-ES, DE, SA, PSO, and SaDE — can provide classifiers as accurate as those obtained from grid search, but taking half of the running time. Our work also shows that the classification accuracies of the different metaheuristics are comparable across the assessed datasets. From this viewpoint, all of them are in principle eligible to be used for optimizing the parameters of a simpler version of the C^3E algorithm based on a Squared Loss function (C^3E-SL). However, CMA-ES, SaDE, and DE could be preferred over the other metaheuristics. CMA-ES and SaDE are less dependent on user-defined parameters, whereas DE achieved the best accuracies and is possibly the simplest one to implement. In other words, from the results achieved and by assuming that there is a trade-off between user-friendliness and classification accuracy, these metaheuristics are the best alternatives for optimizing the C^3E-SL parameters. Equal to or better results than those achieved by classifier ensembles were obtained by using CMA-ES, DE, and SaDE in 75%, 70%, and 60% of the datasets, respectively.

In our future work, we are going to combine C^3E-SL and metaheuristics to solve difficult real-world problems in semi-supervised, active, and transfer learning scenarios, where we believe that the combination of supervised and unsupervised models can provide additional advantages. Besides, there are some issues that can be further investigated, namely: the impact of the type and number of classifiers used, as well as appropriate manners to obtain the similarity matrix (from cluster ensembles). In this sense, a more comprehensive study with a variety of settings, in different domains, may further clarify the capabilities (and potential limitations) of the algorithm.

Acknowledgment

We acknowledge the Brazilian Research Agencies CNPq and FAPESP for their financial support to this work. Also, this research was partially supported by ONR ATL Grant N00014-11-1-0105 and NSF Grants (IIS-0713142 and IIS-1016614).

References

[1] A. Acharya, E. Hruschka, J. Ghosh, and S. Acharyya. C3E: A framework for combining ensembles of classifiers and clusterers. In *Multiple Classifier Systems*, volume 6713 of *Lecture Notes in Computer Science*, pages 269–278. Springer Berlin Heidelberg, 2011.

[2] A. Acharya, E. R. Hruschka, J. Ghosh, and S. Acharyya. An optimization framework for combining ensembles of classifiers and clusterers with applications to nontransductive semisupervised learning and transfer learning. *ACM Trans. Knowl. Discov. Data*, **9**(1):1–35, 2014.

[3] M. Ahmadlou and H. Adeli. Enhanced probabilistic neural network with local decision circles: A robust classifier. *Integrated Computer-Aided Engineering*, **17**(3):197–210, 2010.

[4] T. Bäck, D. B. Fogel, and Z. Michalewicz. *Handbook of Evolutionary Computation*. IOP Publishing, Bristol, UK, 1997.

[5] T. Bäck, F. Hoffmeister, and H. P. Schwefel. A survey of evolution strategies. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 2–9, 1991.

[6] T. Bäck and H.-P. Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evol. Comput.*, **1**:1–23, 1993.

[7] A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh. Clustering with bregman divergences. *J. Mach. Learn. Res.*, **6**:1705–1749, 2005.

[8] S. Basu, I. Davidson, and K. Wagstaff. *Constrained Clustering: Advances in Algorithms, Theory, and Applications*. Chapman & Hall/CRC Press, 2008.

[9] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, **13**:281–305, 2012.

[10] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, **3**(1):1–122, 2011.

[11] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *Evolutionary Computation, IEEE Transactions on*, **10**(6):646–657, 2006.

[12] R. Campello and E. Hruschka. A fuzzy extension of the silhouette width criterion for cluster analysis. *Fuzzy Sets and Systems*, **157**(21):2858–2875, 2006.

[13] B. R. Campomanes-Álvarez, O. Cordón, and S. Damas. Evolutionary multi-objective optimization for mesh simplification of 3D open models. *Integrated Computer-Aided Engineering*, **20**(4):375–390, 2013.

[14] F. Caraffini, F. Neri, B. Passow, and G. Iacca. Re-sampled inheritance search: high performance despite the simplicity. *Soft Computing*, **17**(12):2235–2256, 2013.

[15] F. Caraffini, F. Neri, and L. Picinali. An analysis on separability for memetic computing automatic design. *Information Sciences*, **265**(0):1–22, 2014.

[16] T. Cheng, P. Li, S. Zhu, and D. Torrieri. M-cluster and x-ray: Two methods for multi-jammer localization in wireless sensor networks. *Integrated Computer-Aided Engineering*, **21**(1):19–34, 2014.

[17] L. Coletta, E. Hruschka, A. Acharya, and J. Ghosh. Towards the use of metaheuristics for optimizing the combination of classifier and cluster ensembles. In *Computational Intelligence and 11th Brazilian Computational Intelligence (BRICS-CCI CBIC), 2013 BRICS Congress on*, pages 483–488, 2013.

[18] K. W. De Bock and D. V. d. Poel. An empirical evaluation of rotation-based ensemble classifiers for customer churn prediction. *Expert Systems with Applications*, **38**(10):12293–12301,

2011.

[19] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, **7**:1–30, 2006.

[20] T. G. Dietterich. Ensemble methods in machine learning. In *International Workshop on Multiple Classifier Systems*, pages 1–15. Springer-Verlag, 2000.

[21] M. A. Duval-Poo, J. Sosa-Garcia, A. Guerra-Gandon, S. Vega-Pons, and J. Ruiz-Shulcloper. A new classifier combination scheme using clustering ensemble. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, volume 7441 of *Lecture Notes in Computer Science*, pages 154–161. Springer Berlin Heidelberg, 2012.

[22] Eberhart and Y. Shi. Particle swarm optimization: developments, applications and resources. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, volume 1, pages 81–86, 2001.

[23] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing (Natural Computing Series)*. Springer, 2008.

[24] P. Fleming and R. Purshouse. Evolutionary algorithms in control systems engineering: a survey. *Control Engineering Practice*, **10**(11):1223–1241, 2002.

[25] E. Forcael, V. González, F. Orozco, S. Vargas, A. Pantoja, and P. Moscoso. Ant colony optimization model for tsunamis evacuation routes. *Computer-Aided Civil and Infrastructure Engineering*, **29**:723–737, 2014.

[26] A. Frank and A. Asuncion. UCI Machine Learning Repository [`http://archive.ics.uci.edu/ml`]. University of California, Irvine, School of Information and Computer Sciences. 2010.

[27] A. A. Freitas. *A survey of evolutionary algorithms for data mining and knowledge discovery*, pages 819–845. Springer-Verlag Inc., New York, USA, 2003.

[28] H. Gan, N. Sang, R. Huang, X. Tong, and Z. Dan. Using clustering analysis to improve semi-supervised classification. *Neurocomputing*, **101**(0):290–298, 2013.

[29] J. Gao, F. Liang, W. Fan, Y. Sun, and J. Han. A graph-based consensus maximization approach for combining multiple supervised and unsupervised models. *Knowledge and Data Engineering, IEEE Transactions on*, **25**(1):15–28, 2013.

[30] R. Ghaemi, N. Sulaiman, H. Ibrahim, and N. Mustapha. A Survey: Clustering ensembles techniques. *Proceedings of World Academy of Science, Engineering and Technology*, **38**, 2009.

[31] J. Ghosh and A. Acharya. Cluster ensembles. *Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery*, **1**(4):305–315, 2011.

[32] S. Ghosh-Dastidar and H. Adeli. Wavelet-clustering-neural network model for freeway incident detection. *Computer-Aided Civil and Infrastructure Engineering*, **18**(5):325–338, 2003.

[33] N. Gonçalves, J. Nikkilä, and R. Vigário. Self-supervised mri tissue segmentation by discriminative clustering. *International Journal of Neural Systems*, **24**(01):1450004, 2014.

[34] N. Hansen. The CMA evolution strategy: A comparing review. In *Towards a New Evolutionary Computation*, volume 192 of *Studies in Fuzziness and Soft Computing*, pages 75–102. Springer Berlin Heidelberg, 2006.

[35] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evol. Comput.*, **9**(2):159–195, 2001.

[36] F. Hejazi, I. Toloue, M. Jaafar, and J. Noorzaei. Optimization of earthquake energy dissipation system by genetic algorithm. *Computer-Aided Civil and Infrastructure Engineering*, **28**(10):796–810, 2013.

[37] P. Hore, L. Hall, and D. Goldgof. A scalable framework for cluster ensembles. *Pattern Recognition*, **42**(5):676–688, 2009.

[38] E. R. Hruschka, R. J. Campello, and L. N. D. Castro. Evolving clusters in gene-expression data. *Information Sciences*, **176**(13):1898–1927, 2006.

[39] W.-Y. Hsu. Application of quantum-behaved particle swarm optimization to motor imagery EEG classification. *International Journal of Neural Systems*, **23**(06):1350026, 2013.

[40] C. Igel. No free lunch theorems: Limitations and perspectives of metaheuristics. In *Theory and Principled Methods for the Design of Metaheuristics*, Natural Computing Series, pages 1–23. Springer Berlin Heidelberg, 2014.

[41] A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, **31**(8):651–666, 2010.

[42] L. Jia, Y. Wang, and L. Fan. Multiobjective bilevel optimization for production-distribution planning problems using hybrid genetic algorithm. *Integrated Computer-Aided Engineering*, **21**(1):77–90, 2014.

[43] X. Jiang and H. Adeli. Fuzzy clustering approach for accurate embedding dimension identification in chaotic time series. *Integrated Computer-Aided Engineering*, **10**(3):287–302, 2003.

[44] X. Jiang and H. Adeli. Clustering-neural network models for freeway work zone capacity estimation. *International Journal of Neural Systems*, **14**(03):147–163, 2004.

[45] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942 –1948, 1995.

[46] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science, Number 4598*, **220, 4598**:671–680, 1983.

[47] V. S. Kodogiannis, M. Amina, and I. Petrounias. A clustering-based fuzzy wavelet neural network model for short-term load forecasting. *International Journal of Neural Systems*, **23**(05):1350024, 2013.

[48] L. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, Hoboken, NJ, 2004.

[49] P. J. M. Laarhoven and E. H. L. Aarts, editors. *Simulated annealing: theory and applications*. Kluwer Academic Publishers, Norwell, MA, USA, 1987.

[50] J. Liang, A. Qin, P. Suganthan, and S. Baskar. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *Evolutionary Computation, IEEE Transactions on*, **10**(3):281–295, 2006.

[51] J. M. Luna, J. R. Romero, C. Romero, and S. Ventura. Reducing gaps in quantitative association rules: A genetic programming free-parameter algorithm. *Integrated Computer-Aided Engineering*, **21**(4):321–337, 2014.

[52] H. D. Menéndez, D. F. Barrero, and D. Camacho. A genetic graph-based approach for partitional clustering. *International Journal of Neural Systems*, **24**(03):1430008, 2014.

[53] M. Molina-García, J. Calle-Sánchez, C. González-Merino, A. Fernández-Durán, and J. I. Alonso. Design of in-building wireless networks deployments using evolutionary algorithms. *Integrated Computer-Aided Engineering*, **21**(4):367–385, 2014.

[54] N. Nguyen and R. Caruana. Consensus clusterings. *ICDM '07: Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, **1**, 2007.

[55] N. C. Oza and K. Tumer. Classifier ensembles: Select real-world applications. *Information Fusion*, **9**(1):4–20, 2008.

[56] E. C. Pedrino, V. O. Roda, E. R. R. Kato, J. H. Saito, M. L. Tronco, R. H. Tsunaki, O. Morandin Jr, and M. C. Nicoletti. A genetic programming based system for the automatic construction of image filters. *Integrated Computer-Aided Engineering*, **20**(3):275–287, 2013.

[57] F. Peng and Y. Ouyang. Optimal clustering of railroad track maintenance jobs. *Computer-Aided Civil and Infrastructure Engineering*, **29**(4):235–247, 2014.

[58] K. Price. Differential evolution: a fast and simple numerical optimizer. In *Fuzzy Information Processing Society, 1996. NAFIPS. 1996 Biennial Conference of the North American*, pages 524–527. IEEE, 1996.

[59] A. K. Qin, V. L. Huang, and P. Suganthan. Differential evolution algorithm with strategy adaptation for global numerical optimization. *Evolutionary Computation, IEEE Transactions on*, **13**(2):398–417, 2009.

[60] A. Rahman and B. Verma. Cluster-based ensemble of classifiers. *Expert Systems*, **30**(3):270–282, 2013.

[61] I. Rechenberg. *Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution*. Number 15 in Problemata. Frommann-Holzboog Verlag, Stuttgart-Bad Cannstatt, 1973.

[62] O. Reyes, C. Morell, and S. Ventura. Evolutionary feature weighting to improve the performance of multi-label lazy algorithms. *Integrated Computer-Aided Engineering*, **21**(4):339–354, 2014.

[63] M. Rizzi, M. D'Aloia, and B. Castagnolo. A supervised method for microcalcification cluster diagnosis. *Integrated Computer-Aided Engineering*, **20**(2):157–167, 2013.

[64] F. Samadzadegan, B. Bigdeli, and P. Ramzi. A multiple classifier system for classification of LIDAR remote sensing data using multi-class SVM. In *Multiple Classifier Systems*, pages 254–263. Springer, 2010.

[65] H. P. Schwefel. *Numerical optimization of computer models*. John Wiley & Sons, Inc., Chichester, 1981.

[66] Y. Shi and R. C. Eberhart. Parameter selection in particle swarm optimization. In *Proceedings of the 7th International Conference on Evolutionary Programming VII*, EP '98, pages 591–600, London, UK, 1998. Springer-Verlag.

[67] R. G. F. Soares, H. Chen, and X. Yao. Semisupervised classification with cluster regularization. *Neural Networks and Learning Systems, IEEE Transactions on*, **23**(11):1779–1792, 2012.

[68] R. Storn and K. Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Jornal of Global Optimization*, **11**(4):341–359, 1997.

[69] A. Strehl and J. Ghosh. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, **3**:583–617, 2002.

[70] A. Topchy, A. Jain, and W. Punch. Clustering ensembles: models of consensus and weak partitions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **27**(12):1866 –1881, 2005.

[71] K. Tumer and J. Ghosh. Analysis of decision boundaries in linearly combined neural classifiers. *Pattern Recognition*, **29**(2):341–348, 1996.

[72] B. Twala. Multiple classifier application to credit risk assessment. *Expert Systems with Applications*, **37**(4):3326–3336, 2010.

[73] B. Verma and A. Rahman. Cluster-oriented ensemble classifier: Impact of multicluster characterization on ensemble classifier learning. *Knowledge and Data Engineering, IEEE Transactions on*, **24**(4):605–618, 2012.

[74] J. Vesterstrom and R. Thomsen. A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In *Evolutionary Computation. CEC 2004. Congress on*, volume 2, pages 1980–1987, 2004.

[75] E. D. Wandekokem, E. Mendel, F. Fabris, M. Valentim, R. J. Batista, F. M. Varejao, and T. W. Rauber. Diagnosing multiple faults in oil rig motor pumps using support vector machine classifier ensembles. *Integrated Computer-Aided Engineering*, **18**(1):61–74, 2011.

[76] I. H. Witten and E. Frank. *Data mining: practical machine learning tools and techniques*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, 2nd edition, 2005.

[77] J.-W. Wu, J. C. Tseng, and W.-N. Tsai. A hybrid linear text segmentation algorithm using hierarchical agglomerative clustering and discrete particle swarm optimization. *Integrated Computer-Aided Engineering*, **21**(1):35–46, 2014.

[78] R. Xia, C. Zong, and S. Li. Ensemble of feature sets and classification algorithms for sentiment classification. *Information Sciences*, **181**(6):1138–1152, 2011.

[79] X.-S. Xu, X. Xue, and Z.-H. Zhou. Ensemble multi-instance multi-label learning approach for video annotation task. In *Proceedings of the 19th ACM International Conference on Multimedia*, MM '11, pages 1153–1156, New York, NY, USA, 2011. ACM.

[80] X.-S. Yang. *Nature-inspired metaheuristic algorithms*. Luniver Press, 2nd edition, 2010.

[81] X.-S. Yang. Swarm-based metaheuristic algorithms and no-free-lunch theorems. In *Theory and New Applications of Swarm Intelligence*, pages 1–17. InTech, 2012.

[82] X. Yu and M. Gen. General discussion on performance evaluation. In *Introduction to Evolutionary Algorithms*, pages 101–115. Springer Science & Business Media, 2010.

[83] Z. Zeng, J. Xu, S. Wu, and M. Shen. Antithetic method-based particle swarm optimization for a queuing network problem with fuzzy data in concrete transportation systems. *Computer-Aided Civil and Infrastructure Engineering*, **29**(10):771–800, 2014.

[84] Z.-H. Zhou. When semi-supervised learning meets ensemble learning. *Frontiers of Electrical and Electronic Engineering in China*, **6**(1):6–16, 2011.

[85] Z.-H. Zhou. *Ensemble methods: foundations and algorithms*. CRC Press, 2012.