ISSN 2395-8618

EMiner: A Tool for Selecting Classification Algorithms and Optimal Parameters

Rayrone Zirtany Nunes Marques, Luciano Reis Coutinho, Tiago Bonini Borchartt, Samyr Béliche Vale, and Francisco José da Silva e Silva

***Abstract*—In this paper, Genetic Algorithm (GA) is used to search for combinations of learning algorithms and associated parameters with maximum accuracy. An important feature of the approach is that the GA initial population is formed by using parameter values gathered from ExpDB (a public database of data mining experiments). The proposed approach was implemented in a tool called EMiner, built on top of a grid based software infrastructure for developing collaborative applications in medicine and healthcare domains (ECADeG project). Experiments on 16 datasets from the UCI repository were performed. The results obtained have shown that the strategy of combining the data from ExpDB via GA is effective in finding classification models with good accuracy.**

***Index Terms*—Data mining, medicine and healthcare, algorithm selection, parameter optimization, genetic algorithms.**

I. Introduction

MEDICINE and healthcare are important application areas for data mining technology [1], [2], [3], [4]. Given its nature—viz. to extract information and knowledge from large amounts of raw data—data mining has much to contribute in helping physicians and other researches to analyse medical databases for improving their understanding of diseases, diagnostic skills, treatment procedures, etc. In this sense, one of the most popular and effective data mining techniques has been the use of machine learning algorithms to infer classification models from training datasets [5].

For instance, consider a dataset $\mathcal{D}$ of patient records $\{r_1, \ldots, r_m\}$. Each record r_i being a tuple $(x_1, \ldots, x_n, y)$ of attributes describing the medical condition of a patient. The attributes x_i are data items detailing symptoms, disease factors, and/or results of clinical tests collected from the patient. The attribute y summarizes the patient medical condition in terms of some disease type. In this setting, a completely correct classification model consists in a function $f_{\mathcal{D}}$ that maps the attributes x_i to y such that $f_{\mathcal{D}}(x_1, \ldots, x_n) = y$ iff $(x_1, \ldots, x_n, y) \in \mathcal{D}$. In practice—mainly due to uncertainty, or incomplete and inconsistent data—a completely correct classification model $f_{\mathcal{D}}$ is an impossible idealization. Instead, what is feasible is an approximate classification model $\hat{f}_{\mathcal{D}}$ that tries to minimize the classification error against the current available records and the records added to $\mathcal{D}$ in the future. The approximate classification model $\hat{f}_{\mathcal{D}}$ then represents the knowledge extracted from $\mathcal{D}$ relating antecedents to consequents, and can be used researchers and physicians to explain, diagnose and treat diseases.

The research in machine learning algorithms has produced many approaches for creating good classifiers [5]. On the one hand, such an abundance provides the researcher with proven techniques to mine accurate and general classification models out of raw data. On the other hand, the researcher is faced with the problem of choosing among tenths, or even hundreds, of sophisticated mining algorithms, each one having a set of (hyper)parameters that need to be adjust for the proper functioning of the algorithm [6], [7]. Considering that the medical researcher is not necessarily an expert in data mining, the right choice of algorithm and associated parameter values becomes a challenging task prone to error, or leading to sub-optimal selection based on intuitive appeal and/or reputation. As noted in [6], this suggests an important challenge for machine learning: given a dataset $\mathcal{D}$, how to automatically and simultaneously choose a learning algorithm and set its (hyper)parameters to optimize accuracy and generality; a challenge that can be named *Combined Algorithm Selection and Hyperparameter* optimization problem (or CASH problem).

In this paper, we present an approach to the CASH problem originally developed in the context of ECADeG Project[1] [8]. We put forward a methodology and a tool called EMiner. Our methodology starts with a given dataset $\mathcal{D}$ and a predefined set of available classification algorithms. In general, this set of algorithms is arbitrary. The aim is to select the best classifier, in terms of accuracy/generality, obtainable from the available algorithms with adequate parameter settings. For this, we iterate over all classification algorithms applying a Genetic Algorithm (GA) [9]. This GA is used to evolve a population of candidate classifiers, each candidate classifier consisting in the current algorithm instantiated with a possibly different set of parameter values. Traditionally, the initial population of GA is random. In our case, we propose to form the initial population by using parameter values retrieved from a public database that collects the results of data mining experiments, called ExpDB [10]. With this, i.e. by combining GA with

Manuscript received on July 14, 2015, accepted for publication on September 18, 2015, published on October 15, 2015.

The authors are with the Universidade Federal do Maranhão, Programa de Pós-Graduação em Ciência da Computação, Av. dos Portugueses 1966, Bacanga, São Luís, MA, Brazil (e-mail: rayronezirtany@gmail.com, {lrc, tiagobonini, samyr, fssilva}@deinf.ufma.br).

[1] *Enabling Collaborative Applications for Desktop Grids* is a project whose aim is to provide a grid based software infrastructure to support the development of collaborative applications in medicine and healthcare domains.

ISSN 2395-8618

parameters values coming from the experimental data available in ExpDB, we show that we can automatically reach at good classifiers outperforming default parameters settings, with less experiments than is necessary if we conduct a non-guided search over the state of possible algorithms and associated parameters.

The rest of the paper is organized as follows. Section 2 discuss related work. In Section 3 the technical concepts and specific learning algorithms used in this work are presented. Section 4 describes the proposed methodology and the EMiner tool. In Section 5 we present some of the experiments we have conducted to assess and validate our approach to the CASH problem. Finally, in Section 6 we draw our final conclusions and comment on future work.

II. Related Work

In this section we comment on related work sorted in two main themes: approaches to CASH optimization problem, and approaches employing GA to optimize parameter values.

Dealing with the CASH optimization problem, we found [6] and [7]. Thornton et al. [6] use recent innovations in Bayesian optimization to find the best parameter values for a classification algorithm. They formally define the problem of algorithm selection and parameter optimization in classification problems, naming this type of problem by the acronym CASH (*Combined Algorithm Selection Hyperparameter*) optimization problem. They demonstrate that the learning algorithm itself can be considered a parameter and that Bayesian optimization obtains high quality results in a relatively reasonable time. Finally, to fully automate the approach, they built a tool called Auto-Weka, extending/adapting the traditional tool Weka.

Leite et al. [7] propose a new technique for selecting classification algorithms called *active testing*. This technique selects the most useful algorithm using cross-validation testing on a tournament where, in each round of selection and test, it is chosen the algorithm that outperformed the algorithms that won the previous round. The most promising algorithm is then chosen based on the tournament history over similar datasets. To evaluate the approach, 292 combinations of algorithms-parameters are used to analyze 76 datasets. The results showed that active testing quickly come up with a combination algorithm-parameter whose performance is close to optimal.

Among the approaches employing GA to optmize parameter values, we found the work by Samadzadegan et al. [11] using GA to select an optimal kernel and respective parameters in the learning of *Support Vector Machines* (SVM). The results showed that the proposed method outperformed in terms of accuracy when compared to traditional *Grid Search* (brute force). Another work is [12] that also use GA to optimize SVM kernel and parameters. The author explains that the traditional combination causes a problem of premature convergence that limits the accuracy of the SVM obtained. Then he suggests new genetic operators (called IO-GA) specially designed to optimize the SVM kernel/parameters. The results showed an increase in classification accuracy over the traditional operators. Several other work in the literature address the issue of parameter optimization for classification algorithms by using GA [13], [14], [2], [4]. In general, they do not take into account previous experiments to initially guide their search for better results, and are limited to the parameter optimization of a single classification algorithm.

III. Conceptual Background

In this section we further detail the CASH optimization problem, the specific classification algorithms we have used in our experiments, the notion of genetic algorithms (GA) and the specific GA we have applied in our approach. We end the section with a description of the ExpDB project.

A. CASH Optimization Problem

To better characterize the CASH optimization problem, let us first consider the restricted problem of *parameter optimization* for a fixed machine learning algorithm $\mathcal{A}$ when this is applied to infer a classification model $\hat{f}_{\mathcal{D}}$ from a given dataset $\mathcal{D}$. Suppose that $\Lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n)$ is the list of (hyper)parameters of $\mathcal{A}$, where each λ_i is a variable defined over some domain $\{\alpha_i\}$ of values (discrete or continuous). Given $\mathcal{D} = \{(x_1, \ldots, x_n, y)\}$, the problem of optimizing the parameters of $\mathcal{A}$ consists in finding assignments $\lambda_i := \alpha_i^*$, $1 \leq i \leq n$, leading to maximum accuracy/generality of $\mathcal{A}$ over $\mathcal{D}$. Thus, denoting the set of assignments $\{\lambda_i := \alpha_i\}_{1 \leq i \leq n}$ by α and the measure of accuracy/generality of $\mathcal{A}$ (with parameter values α) over $\mathcal{D}$ by $\mathcal{M}_{ag}[\mathcal{A}(\alpha), \mathcal{D}]$, the parameter optimization problem can be formulated compactly as the search for $\alpha^* \in \underset{\alpha}{\textbf{argmax}}\ \mathcal{M}_{ag}[\mathcal{A}(\alpha), \mathcal{D}]$.

Traditionally a method of measuring accuracy/generality of a given combination of algorithm and parameter values is to perform a cross-validation test with k-folds [15]. In this method, the data set $\mathcal{D}$ is randomly divided into k mutually exclusive subsets $B_1, B_2, \ldots, B_k$ of approximately equal size called *folds*. Then, the classification algorithm $\mathcal{A}$ with appropriate parameters values α is executed k times; each time $t \in \{1, 2, ..., k\}$ the algorithm $\mathcal{A}(\alpha)$ is trained using $\mathcal{D} - B_t$ and tested against B_t. At the end, the measure of accuracy/generality is obtained as follows

$$\mathcal{M}_{ag}[\mathcal{A}(\alpha), \mathcal{D}] = \frac{1}{k} \sum_{t=1}^{k} \frac{|B_t^c|}{|B_t|} \times 100, \tag{1}$$

where $|B_t|$ is total number of records in B_t, and $|B_t^c|$ is number of records correctly classified by the model $\hat{f}_{\mathcal{D}-B_t}$ obtained at time t; a record is correctly classified when $\hat{f}_{\mathcal{D}-B_t}(x_1, \ldots, x_n) = y$ implies $(x_1, \ldots, x_n, y) \in B_t$.

In this work we use the method of k-folds cross-validation to estimate the accuracy/generality of the algorithms in our experiments. We adopt $k = 10$ because it is a value that provides good estimates as established by several research papers [15], [7], [6], [10].

 http://dx.doi.org/10.17562/PB-52-2

ISSN 2395-8618

Finally, from this formulation of the restricted problem of parameter optimization, the more general CASH optimization problem consists in the search for both algorithm $\mathcal{A}^*$ and respective parameter values α^* that lead to maximum accuracy/generality in $\mathcal{D}$; i.e., find $\mathcal{A}^*(\alpha^*) \in \underset{\mathcal{A}(\alpha)}{\textbf{argmax}}\ \mathcal{M}_{ag}[\mathcal{A}(\alpha), \mathcal{D}]$.

B. Classification Algorithms

In medical application of data mining two crucial feature are the *transparency* and the *interpretability* of the models obtained [1]. By this we mean the possibility of human analysis and validation of the models generated by a given learning technique. This is crucial because the physician or researcher using the models in general wants to be able to explain and justify its decisions when being guided by some classification model.

Taking into account these desirable features, the classification algorithms we have used in our work are divided into two categories: algorithms based on *decision trees induction* and algorithms based on *classification rule induction*. The choice of these categories is justified by the transparency and interpretability of the classifiers obtained with them, once both categories are based on human comprehensible representations, which facilitates their analysis and validation. Among the algorithms for decision trees induction we have used J48 [16], RandomTree [3] and REPTree [5]. For classification rule induction JRip [17], PART [5], and Ridor [5]. In Table I we summarize the (hyper)parameters of used classification algorithms.

TABLE I
CLASSIFICATION ALGORITHMS AND RESPECTIVE (HYPER)PARAMETERS.

Algorithm	Parameter	Type	Def. Value
J48	Use binary splits	boolean	false
	Confidence threshold for pruning	double	0.25
	Cleanup after the tree has been built	boolean	true
	Subtree raising to be performed	boolean	true
	Minimum number of instances per leaf	integer	2
	Number of folds for reduced error pruning	integer	3
	Use reduced error pruning	boolean	false
	Random number seed for reduced-error pruning	integer	1
	Use Laplace correction	boolean	false
	Unpruned tree	boolean	false
R. Tree	Minimum number of instances for leaf	double	1.0
	The number of attributes considered for a split	integer	0
	The random seed to use	integer	1
REPTree	Maximum tree depth	integer	-1
	Minimum class variance proportion of train	double	0.001
	Minimum number of instances	integer	2
	Switch off pruning	boolean	false
	Number of folds for reduced error pruning	integer	3
	Seed for random data shuffling	integer	1
JRip	Switch off checking error rate $>= 0.5$	boolean	true
	Min weights of instances within a split	double	2.0
	Switch off pruning	boolean	false
	Number of folds for reduced error pruning	integer	3
	Number of optimization runs	integer	2
	Seed for random data shuffling	integer	1
PART	Use binary splits on nominal attributes	boolean	false
	Confidence factor for rule pruning	double	0.25
	Generate unpruned decision list	boolean	false
	Minimum number of instances per leaf rule	integer	2
	Number of folds for reduced error pruning	integer	3
	Use reduced error pruning	boolean	false
	Random number seed for reduced-error pruning	integer	1
Ridor	Minimal weights of instances within a split	double	2.0
	Number of folds for reduced error pruning	integer	3
	Number of randomization shuffles	integer	10
	Use error rate of all data	boolean	false
	Use majority class as default class in each step	boolean	false

C. Genetic Algorithms

Genetic Algorithms (GA) are optimization algorithms based on principles of natural selection and genetics [9]. The main idea is that in a population, the individuals with more favorable genetic traits are more likely to survive, reproduce and give birth to an increasingly fit offspring, while less fit individuals tend to become extinct. In GA, each individual in the population (in the form of chromosomes) represents a candidate solution for a given problem. To find better individuals, the GA uses a random search strategy, favoring points of high fitness; i.e., points at which the function to be minimized or maximized assumes relatively low or high values. The search process is iterative, where each iteration is called a generation. In each generation, the mechanisms of selection and reproduction are applied to individuals of the population. Through selection, it is determined which individuals are apt to reproduce, generating a certain number of offspring. The probability of an individual being selected is proportional to its fitness; i.e., the higher fitness of an individual the greater chance of it being selected.

Depending on the problem, the chromosomes can be further divided in logical sections called genes. In this paper, chromosomes are used to represent the set of parameters values peculiar to a given classification algorithm. Thus, each gene determines the value of a specific parameter can take. Once we are using 6 different classification algorithms, each one with different parameters, we have defined 6 types of chromosomes in order to meet the particularities of each one of them. For instance, for the J48 algorithm, which has 10 parameters in total, we have defined a chromosome divided in 10 genes, each gene with a given number of bits appropriate to the parameter types.

The evaluation of the individuals is done by means of a fitness function. This is intended to measure how apt is a given individual (solution) from the current population. The aptitude of an individual influences in its survival probability. In this work, the fitness function used is the mean accuracy/generality of the classifier (algorithm + parameter values) after the process of k-fold cross-validation given by Eq. (1).

Via crossover, the information contained in the genes of two existing individuals are combined in order to create new individuals. This operator is the main mechanism of the GA to explore the search space, by creating new points obtained by the exchange of genetic information between individuals. In this work we use a crossing rate corresponding to 70% of the population and the crossover point is chosen randomly.

During the crossing operation, can be born individuals who

ISSN 2395-8618

violate constraints of dependence between parameters imposed by the classification algorithm. For example, the parameter *number of folds for reduced error pruning* are relevant if the parameter *use reduced error pruning* is true; i.e., first parameter (son) depends on second (parent). To address this problem, the chromosomes was designed to correct such violations using value of the parent parameter. In the previous example, if the parameter *use reduced error pruning* was true, then the parameter value *number of folds for reduced error pruning* would be disregarded by assigning a null value or zero depending on the domain parameter.

The mutation is a genetic operator which serves to introduce random variations in the population, or even to recover some features lost in operations such as crossover. Furthermore, mutation is used to prevent premature convergence to local optima, by sampling new points in the search space. This operator is applied with a given probability, after the crossover operator. In this work, the mutation rate varies according to the size X of the population according to the ratio $1/X$; i.e., on average 1 in X individuals is mutated [18].

Similarly crossover operator, the mutation operator can also generate individuals that violate some restrictions of parameters values of classification algorithm, therefore the same strategy adopted with crossover operator was used for handling this problem.

D. ExpDB

ExpDB[2] is a database designed to store a large number of data mining experiments, containing detailed information on the datasets, the algorithms and the parameter settings used, as well as the evaluation procedures and the obtained results [10]. Currently this database contains about $600,000$ experiments on 130 datasets, in varied subjects such as healthcare, finances, education, etc. The main idea is to facilitate large-scale experimentation, guarantee repeatability and reusability of experiments, and help to clarify the conditions under which certain results are valid. We use ExpDB to obtain already tested initial values for the parameters of the classification algorithms. These collected values will form the initial population to be improved via the application of GA.

IV. EMiner Tool

Based on GA and ExpDB, we have proposed a methodology for handling the CASH optimization problem. This methodology is divided into three phases: *initial values definition*, *parameter optimization* and *algorithm selection*. Figure 1 ilustrates the main phases and steps of the proposed methodology.

In the first phase we have the definition of classification algorithms to be used and the extraction of the initial values for their parameters from the base ExpDB. The extraction occurs by first getting raw parameter sets from ExpDB. Then a filtering is performed to remove the repetead sets. Finally, the distinct parameter sets are encoded in different chromosomes to generate the initial population of the GA.

The second phase is concerned with the optimization of the classification algorithm parameters by using GA. Here it is performed the cross-validation (Eq. (1)) of each individual (representing a classifier for a given dataset) in the population. Each cross-validation uses k folds to test an individual and calculate their mean accuracy (fitness). Then the genetic operators (selection, crossover and mutation) are applied to the current generation to give rise to a new generation. After a given number of generations the best classifier is selected.

Finally, the last phase deals with selecting the classification algorithm for a given dataset. So, it repeats the previous phases for each classification algorithm in order to built a rank of used algorithms and parameter values.

A first implementation of our approach to the CASH problem was performed in the context of the ECADeG Project. It is a prototype tool called EMiner (ECADeG Miner), available for download at the address `http://lsd.ufma.br/joomla/index.php/projetos/8-ecadeg`.

V. Experimental Evaluation

A. Computing Environment

The genetic algorithm implemented in EMiner was written in JAVA using the framework JGAP [18]. JGAP is a Genetic Algorithms and Genetic Programming component provided as a Java framework. It provides basic genetic mechanisms that can be easily used to apply evolutionary principles to problem solutions. Regarding the implementation of the decision trees and classification rules algorithms, and the k-folds cross-validation process, we have used the WEKA [19] API. WEKA is a collection of machine learning algorithms for data mining tasks that contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. All of out experiments were run on a InteGrade desktop grid formed by Linux machines with Intel Core I5, running at 3GHz and 4GB of RAM.

B. Datasets

We performed several experiments using datasets from the UCI database [20]. We have chosen datasets that have been frequently used as benchmarks for evaluating classification methods in the literature. These datasets consist of discrete and continuous features. Table II summarizes the number of classes, number of discrete and continuous attributes found in the datasets.

C. Experiments

Initially—in order to get a baseline for comparison—we conducted experiments to measure the performance of the 6 classification algorithms of Section III-B, in terms of mean accuracy (Eq. (1)), when each one of these is configured with

[2]http://expdb.cs.kuleuven.be/

 http://dx.doi.org/10.17562/PB-52-2

ISSN 2395-8618

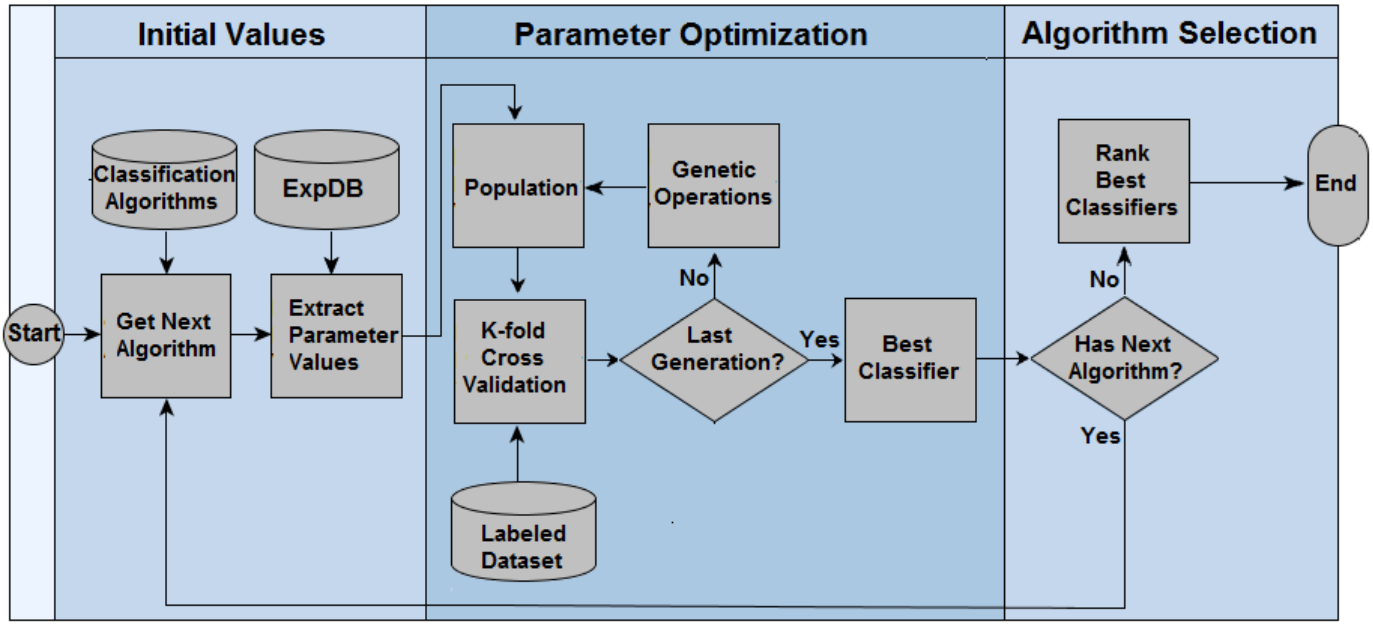

Fig. 1. Methodology for parameter optimization and algorithm selection.

TABLE II
DATASETS FROM THE UCI REPOSITORY.

ID	Name	Num. Discr.	Num. Cont.	Num. Classes	Num. Training	Num. Test
1	Abalone	1	7	28	2923	1254
2	Amazon	10000	0	50	1050	450
3	Cars	6	0	4	1210	518
4	Convex	0	784	2	50000	8000
5	Dexter	20000	0	2	420	180
6	German Credit	13	7	2	700	300
7	Gisette	5000	0	2	4900	2100
8	KDD09-Appentency	190	40	2	35000	15000
9	KR-vs-KP	37	0	2	2238	958
10	Madelon	500	0	2	1820	780
11	Secom	0	591	2	1096	471
12	Semeion	256	0	10	1115	478
13	Shuttle	9	0	7	43500	14500
14	Waveform	0	40	3	3500	1500
15	Wine Quality	0	11	11	3425	1469
16	Yeast	0	8	10	1038	446

their default parameter values. The results obtained using the trainning instances (column "Num. Training" of Table II) are summarized in the second column of Table IV. In the sixth column of Table IV, we present the results obtained using the test instances (column "Num. Test" of Table II).

Next we performed experiments by using the sets of parameter values obtained from ExpDB. The number of parameters sets considered is shown in Table III. In this case, for each combination of algorithm and dataset, we performed as many experiments as the number of parameters sets present in Table III. In Table IV, third and seventh columns, we summarize the results obtained for trainning and test instances, respectively.

Thirdly we ran the GA for the classification algorithms and datasets being considered using 20 randomly chosen parameter sets as the initial population, over 100 generations.

TABLE III
NUMBER OF PARAMETER SETS OBTAINED FROM EXPDB.

ID	Algorithm	Avaliable ExpDB Parameter Sets	Used ExpDB Parameter Sets
A1	J48	15479	500
A2	Random Tree	21	21
A3	REPTree	21	21
A4	JRip	21	21
A5	PART	21	21
A6	Ridor	21	21

The motivation for these is to show that the use of the parameter values from ExpDB are really useful in obtaining classifiers with greater performance. The results obtained are shown in Table IV, fourth and eighth columns.

As a last round of experiments, we used our approach (Section 4). As long as it combines ExpDB and GA, each experiment consisted in performing 10-folds cross-validation (Eq. (1)) in each individual of a population of candidate classifiers (algorithm + parameters) over a number of generations. The population size and number of generations were fixed in 20 and 50 respectively. The final results can be seen in Table IV, fifth and ninth columns, for the training and test instances, respectively.

D. Discussion

The results obtained in the test performance show that the proposed approach has been effective. When we analyse line by line the Table IV we note that, in general, the parameters values from ExpDB overcome the results obtained by using default values. The exception occurs in dataset 13.

ISSN 2395-8618

TABLE IV
PERFORMANCE ON BOTH 10-FOLD CROSS-VALIDATION AND TEST DATA. EXPERIMENTS USING DEFAULT VALUES, EXPDB VALUES, GENETIC ALGORITHM AND GENETIC ALGORITHM WITH EXPDB VALUES (EMINER APPROACH). WE REPORT RESULTS AS MEAN ACCURACY ACROSS ALL CLASSIFIERS TESTED.

	10-Fold C.V. Performance (%)				Test Performance (%)			
Dataset	Default	ExpDB	GA	EMiner	Default	ExpDB	GA	EMiner
1	24.56	26.16	27.84	**28.39**	23.30	24.50	26.50	**27.69**
2	35.52	36.48	36.57	**36.76**	33.11	33.78	33.11	**36.67**
3	94.79	94.79	**96.69**	**96.69**	97.10	97.10	98.91	**98.91**
4	53.88	53.91	**53.94**	**53.94**	53.86	53.86	53.79	**53.79**
5	85.48	86.67	**92.38**	**92.38**	87.22	86.67	**88.33**	**88.33**
6	75.14	75.43	**77.43**	**77.43**	72.33	72.00	73.00	**73.00**
7	94.41	94.65	**95.31**	94.90	94.48	94.71	94.52	**94.76**
8	**98.20**	**98.20**	**98.20**	**98.20**	98.26	98.26	98.26	**98.26**
9	99.15	99.29	**99.69**	**99.69**	99.06	99.37	99.37	**99.58**
10	75.71	77.80	72.64	**79.75**	79.62	80.26	72.95	**83.26**
11	93.89	93.98	**94.44**	**94.44**	92.13	94.13	**95.66**	**95.66**
12	75.99	75.99	**78.58**	**78.58**	76.10	76.10	**76.52**	**76.52**
13	99.98	99.97	**99.99**	**99.99**	99.98	99.98	**99.99**	**99.99**
14	79.40	79.40	**81.11**	80.29	78.07	78.07	**78.20**	**78.20**
15	57.25	58.33	58.73	**59.38**	59.29	58.34	**61.23**	60.54
16	59.19	61.31	**61.50**	61.31	58.43	**59.78**	59.33	**59.78**

When we compare the columns of ExpDB values and the EMiner approach, we see that the combination of ExpDB and GA obtain better results than the use of the ExpDB parameter values alone (and better than the default parameter values, by transitivity). In all databases, we obtained classifiers with greater or equal accuracy by initiating with the ExpDB parameters values and evolving these parameter values by means of the GA.

When we analyse the simple GA approach with random initialization versus EMiner approach, we observe that the gain obtained by the combination ExpDB and GA is not only due to the GA, but it is also influenced by the use of the ExpDB parameter values to form the initial population for the GA. In general, an initial population of randomly chosen parameter values fared worse than one based on the ExpDB values. In our experiments, this observation failed only in datasets 7, 14 and 16 where we obtained a better classifier by starting from a randomly initial population. In most cases, when the genetic algorithm is started with the values of ExpDB, the best configuration of the parameters is found with a smaller number of generated populations. This advantage can be seen in the left chart in Fig. 2, where we compare the average number of generated populations until find the best accuracy for each dataset. The right chart in Fig. 2 show the relative performance improvement when the GA is started with ExpDB values. Among the sixteen datasets used, only in the *convex* dataset the use of ExpDB values has not resulted in a performance gain.

Given all these results we plot on Fig. 3 a bar chart showing the distribution of algorithm selection over the four group of experiments executed. Looking at the chart we see that the decision tree algorithms (J48, RandomTree and REPTree) dominated the algorithms based on classification rules (JRip, PART and Ridor). Of all, the most frequently used algorithm was J48 and PART, which were selected 4 in the experiments involving GA with ExpDB; 6 and 3 times (resp.) in GA alone; 7 and 6 time (resp.) in ExpDB alone, and 4 times in the default values experiments. Notwithstanding this dominance, all algorithms had their change to be selected as the best one in at least one experimental setup, what shows the importance of automated solutions to the CASH optimization problem.

TABLE V
COMPARISON WITH OTHER WORK.

Dataset	EMiner	Thornton, et al. [6]
1	**27.69**	27.29
2	36.67	**62.44**
3	98.91	**100.00**
4	53,79	**77.95**
5	88.33	**92.78**
6	**73.00**	72.33
7	94.76	**97.76**
8	**98.26**	**98.26**
9	**83.26**	79.23
10	**95.66**	92.13
11	76.52	**94.97**
12	**99.99**	**99.99**
13	78.20	**85.80**
14	60.54	**66.44**
15	**59.78**	59.55

Finally, to complement the empirical evaluation of the proposed approach, we also compared the best results we obtained in Table IV with the best result found in literature [6], which use in their experiments the same datasets as we did. Table V summarizes the comparison. We observe that the proposed approach outperforms many time the results we could found on the literature.

Compared to the literature, we conclude that the main contribution of our work is the use of GA to select the most promising combination of the algorithm and parameter values for a given classification task, starting the search with experimental data retrieved from the public ExpDB repository [10]. Our approach to the CASH optimization problem is an alternative to the approaches presented in [6] and [7].

ISSN 2395-8618

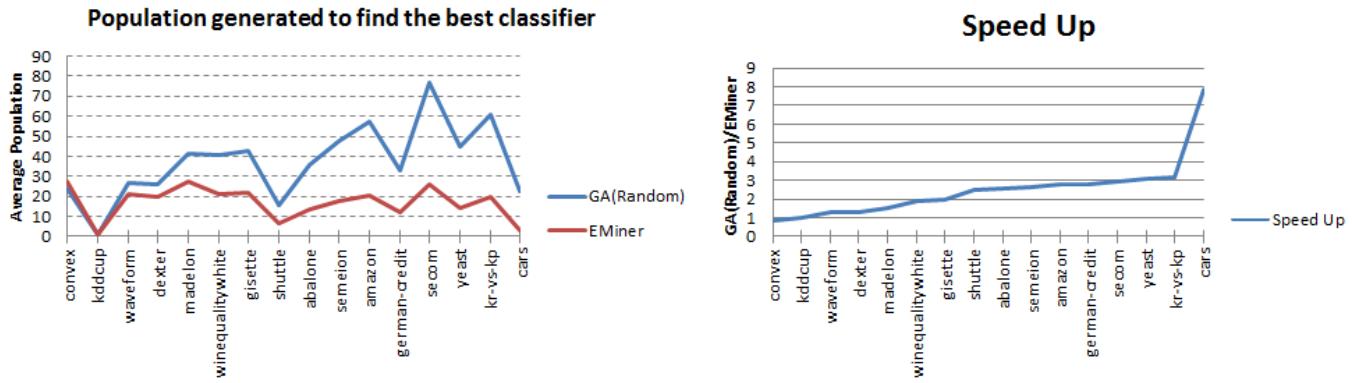

Fig. 2. Speedup comparison between simple GA approach and EMiner.

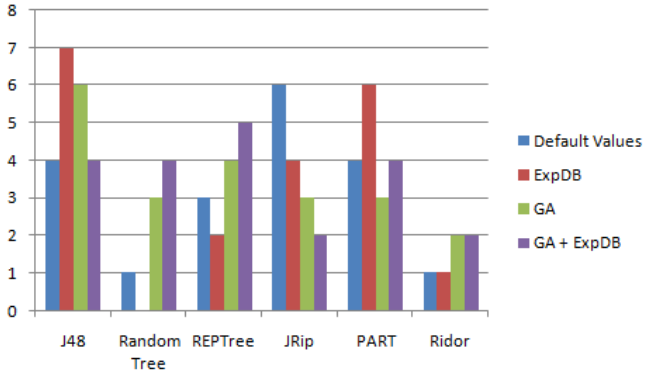

Fig. 3. Distribution of algorithm selection.

VI. Conclusion

In this work, it was shown how the problem of algorithm selection and (hyper)parameter optimization, or CASH optimization problem, can be approached in an automated manner. We have proposed an approach that combines GA with data extracted from the ExpDB to search for a set of suitable parameter values, for a given list of classification algorithms, when we want to find a classification model with maximum accuracy/generality on a specific data mining application. The classification algorithms experimented in this study were 3 algorithms for decision tree induction (J48, Random Tree and REPTree) and 3 for classification rule induction (JRip, PART and Ridor). The proposed approach was implemented in tool called EMiner, built on the context of the ECADeG project.

Regarding the CASH optimization problem, the main insight of this work is the use of GA with a non-random initial population. This non-random initial population consists of data from various experiments previously conducted in datasets in different areas, and stored in ExpDB. Regarding the application of data mining in medicine and healthcare, the main contribution is the development of an automated tool to help medical researchers to come up with good classification models, despite not being experts in data mining algorithms and techniques.

To validate our approach, several experiments have been performed. These experiments allowed us to evaluate the gain in accuracy due to the introduction of ExpDB and GA, the two basic elements of the proposed approach. Also, these experiments allowed us to compare the approach with some work reported in the literature. We have concluded that the proposed approach is effective in some of the cases analysed.

As future work we envisage several research paths. The first one is to further test the approach with other data mining algorithms. Another is to conduct experiments to assess in more detail the real effectiveness and efficiency of starting from an initial population based on data from ExpDB instead of starting from a random one. Finally, as we strive for the construction of practical tools to help researchers in the area of medicine and healthcare, we are thinking in conducting qualitative experiments to see how our EMiner tool fares when evaluated by these professionals.

Acknowledgment

The authors would like to thank FAPEMA (State of Maranhão Research Agency – Brazil) for supporting this work, grants INRIA-00114/11.

References

[1] N. Lavrac, "Selected techniques for data mining in medicine," *Artificial Intelligence in Medicine*, vol. 16, pp. 3–23, 1999.

[2] M. Adnan, W. Husain, and N. Rashid, "A hybrid approach using naïve bayes and genetic algorithm for childhood obesity prediction," in *2012 International Conference on Computer Information Science (ICCIS)*, vol. 1, June 2012, pp. 281–285.

[3] P. K. Srimani and M. S. Koti, "Medical diagnosis using ensemble classifiers – a novel machine-learning approach," in *Journal of Advanced Computing*, 1st ed. Columbia International Publishing, 2013, pp. 9–27.

[4] S. Amin, K. Agarwal, and R. Beg, "Genetic neural network based data mining in prediction of heart disease using risk factors," in *2013 IEEE Conference on Information Communication Technologies (ICT)*, April 2013, pp. 1227–1231.

[5] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Pratical Machine Learning Tools and Techniques*, 3rd ed. Elsevier, 2011.

ISSN 2395-8618

[6] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD-13. New York, NY, USA: ACM, 2013, pp. 847–855. [Online]. Available: http://doi.acm.org/10.1145/2487575.2487629

[7] R. Leite, P. Brazdil, and J. Vanschoren, "Selecting classification algorithms with active testing," in *Proceedings of the 8th International Conference on Machine Learning and Data Mining in Pattern Recognition*, ser. MLDM-12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 117–131. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-31537-4_10

[8] F. Maia, R. Araújo, L. C. Muniz, R. Zirtany, L. Coutinho, S. Vale, F. J. Silva, P. Cincilla, I. Chabbouh, S. Monnet, L. Arantes, and M. Shapiro, "A grid based distributed cooperative environment for health care research," in *Foundations of Health Information Engineering and Systems Lecture Notes in Computer Science Volume*, vol. 7789. Springer Berlin Heidelberg, 2013, pp. 142–150. [Online]. Available: http://link.springer.com/chapter/10.1007%2F978-3-642-39088-3_9

[9] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA, USA: MIT Press, 1992.

[10] H. Blockeel, J. Vanschoren, B. Pfahringer, and G. Holmes, "Experiment databases," *Machine Learning*, vol. 87, no. 2, pp. 127–158, 2012. [Online]. Available: http://dx.doi.org/10.1007/s10994-011-5277-0

[11] F. Samadzadegan, A. Soleymani, and R. Abbaspour, "Evaluation of genetic algorithms for tuning SVM parameters in multi-class problems," in *2010 11th International Symposium on Computational Intelligence and Informatics (CINTI)*, Nov 2010, pp. 323–328.

[12] J. Zhou, O. Maruatona, and W. Wang, "Parameter optimization for support vector machine classifier with IO-GA," in *2011 First International Workshop on Complexity and Data Mining (IWCDM)*, Sept 2011, pp. 117–120.

[13] K.-K. Seo, "A GA-based feature subset selection and parameter optimization of support vector machine for content-based image retrieval," in *Proceedings of the 3rd International Conference on Advanced Data Mining and Applications*, ser. ADMA-07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 594–604. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-73871-8_57

[14] A. Sureka and K. Indukuri, "Using genetic algorithms for parameter optimization in building predictive data mining models," in *Advanced Data Mining and Applications*, ser. Lecture Notes in Computer Science, C. Tang, C. Ling, X. Zhou, N. Cercone, and X. Li, Eds. Springer Berlin Heidelberg, 2008, vol. 5139, pp. 260–271. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-88192-6_25

[15] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proceedings of the 14th International Joint Conference on Artificial Intelligence – Volume 2*, ser. IJCAI-95. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995, pp. 1137–1143.

[16] J. R. Quinlan, "Improved use of continuous attributes in c4.5," *J. Artif. Int. Res.*, vol. 4, no. 1, pp. 77–90, Mar. 1996. [Online]. Available: http://dl.acm.org/citation.cfm?id=1622737.1622742

[17] A. K. Tanwani, J. Afridi, M. Z. Shafiq, and M. Farooq, "Guidelines to select machine learning scheme for classification of biomedical datasets," Islamabad, Pakistan, 2009.

[18] D.-Y. Chen, T.-R. Chuang, and S.-C. Tsai, "JGAP: A Java-based graph algorithms platform," *Softw. Pract. Exper.*, vol. 31, no. 7, pp. 615–635, Jun. 2001. [Online]. Available: http://dx.doi.org/10.1002/spe.379

[19] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: An update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, Nov. 2009.

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