Handling Imbalanced Classification Problems with Support Vector Machines via Evolutionary Bilevel Optimization

Alejandro Rosales-Pérez, Salvador García, Francisco Herrera, *Senior Member, IEEE*

Abstract—Support vector machines are popular learning algorithms to deal with binary classification problems. They traditionally assume equal misclassification costs for each class; however, real-world problems may have an uneven class distribution. This paper introduces EBCS-SVM: Evolutionary Bilevel Cost-sensitive Support Vector Machines. EBCS-SVM handles imbalanced classification problems by simultaneously learning the support vectors and optimizing the SVM hyper-parameters, which comprise the kernel parameter and misclassification costs. The resulting optimization problem is a bilevel problem, where the lower-level determines the support vectors and the upper-level the hyper-parameters. This optimization problem is solved using an evolutionary algorithm at the upper-level and Sequential Minimal Optimization at the lower-level. These two methods work in a nested fashion, i.e., the optimal support vectors help guide the search of the hyper-parameters, and the lower-level is initialized based on previous successful solutions. The proposed method is assessed using 70 datasets of imbalanced classification and compared with several state-of-the-art methods. The experimental results, supported by a Bayesian test, provided evidence of the effectiveness of EBCS-SVM when working with highly imbalanced datasets.

Index Terms—Support Vector Machines, Imbalanced Classification, Data Preprocessing, Evolutionary Algorithms, Bilevel Optimization.

I. INTRODUCTION

SUPPORT vector machines (SVMs) [1] are among the most popular supervised learning algorithms, with strong theoretical foundations and high effectiveness in real-world problems. The idea behind SVMs is to find the hyper-plane that maximizes the separation margin between two categories. In their canonical form, SVMs assume an equal cost for each class. This assumption works well when the number of instances for each class is roughly similar. However, real-world problems seldom have a balanced class distribution.

The imbalanced classification problem refers to an uneven class distribution [2]–[4], i.e., there is an over-represented class, known as the majority class, and an under-represented class, known as the minority class. Imbalanced classification problems further have the characteristic that the minority class is the one of interest. Therefore, accurately recognizing the minority class becomes crucial in several applications, such as medical diagnosis, fraud detection, and face recognition.

There are two main approaches to deal with imbalanced classification problems with SVMs: data-level and algorithm-level methods [5]. The first approach aims to balance the dataset through oversampling the minority class [3], [6]–[8] or undersampling the majority class [3], [9]–[11]. Then, SVM learns from the edited dataset [12]. Although data-level methods are flexible, they ignore the particularities of learning algorithms. Conversely, algorithm-level methods modify the learning algorithm to be robust to uneven class distributions. For SVMs, common modifications comprise hyper-plane shifting [13], [14], kernel adaptation [15], and cost-sensitive [16]. Algorithm-level methods often offer better performance than data-level ones [16]; however, they need to define a set of hyper-parameters, such as the extent of shifting compensation or the correct costs[1] for each class. Therefore, methods for imbalanced classification must not only learn when class distributions are unequal, but their hyper-parameters must also be tuned to get peak performance.

Bilevel optimization (BLO) arises as an alternative for hyper-parameter optimization. BLO differs from traditional optimization in that the optimization problem has as part of its constraints a second optimization problem. In our context, the principal optimization or upper-level problem is the hyper-parameter optimization, and the second or lower-level problem is the learning of support vectors. These two problems interplay, i.e., the definition of the hyper-parameters influences the optimal set of support vectors, and this set of support vectors defines the model to predict unseen cases. However, BLO problems are computationally challenging because of their non-convexity

A. Rosales-Pérez (alejandro.rosales@cimat.mx) is with the Centro de Investigación en Matemáticas (CIMAT), Monterrey, N.L., Mexico.

S. García (salvagl@decsai.ugr.es) and F. Herrera (herrera@decsai.ugr.es) are with the Department of Computer Science and Artificial Intelligence, Andalusian Research Institute in Data Science and Computational Intelligence (DaSCI), University of Granada, 18071, Granada, Spain.

[1]The misclassification costs are the weights applied to errors incurred by classifying positive or negative samples.

and non-linearity [17], [18].

Evolutionary algorithms (EAs) are powerful search tools capable of solving complex optimization problems, such as BLO problems. Recently, the interest in using EAs to address machine learning problems is growing fastly [19]–[29]. For imbalanced learning, EAs have been used for data sampling [30], [31] and cost-sensitive learning [32]. Although recent studies address the problem of determining the optimal misclassification costs [32], [33], they have paid little attention to considering the hyper-parameters of the learning algorithm, along with exploiting the hierarchical nature of parameter and hyper-parameter learning to guide search. Furthermore, taking advantage of the properties of the learning algorithm to estimate the classification performance efficiently in imbalanced problems is almost unexplored.

In the light of the above mentioned, this paper introduces EBCS-SVM: Evolutionary Bilevel Cost-Sensitive Support Vector Machines. EBCS-SVM combines an EA and the Sequential Minimal Optimization (SMO) algorithm in a nested manner. The EA optimizes the cost of hyper-parameters, which are the costs of each class and the kernel parameters, and the SMO learns the optimal support vectors. These two optimizers interact such that information from one level is used by the other to improve search and convergence capabilities. We summarize the main contributions of this paper as follows:

- We propose EBCS-SVM, which allows learning SVMs in imbalanced classification problems and automatically sets the misclassification costs and kernel parameter.
- EBCS-SVM uses the information from the lower-level to guide the search to the upper-level and takes advantage of the previous successful hyper-parameters to initialize the set of support vectors.
- The bilevel formulation that jointly learns parameters and hyper-parameters.
- The definition of the upper-level objective function that allows estimating the performance of the SVM without performing cross-validation.

The performance of EBCS-SVM was assessed using a suite of 70 benchmark datasets of imbalanced classification and compared with the state-of-the-art methods. The experimental evaluation revealed an outstanding efficacy of EBCS-SVM when faced with problems with a high disproportion of classes. The hierarchical Bayesian test supported the main findings.

The rest of this paper is organized as follows. Section II introduces the related works on imbalanced classification problems, hyper-parameter optimization, and bilevel optimization. Section III describes the general bilevel formulation for cost-sensitive SVM. Next, Section IV describes the proposed EBCS-SVM. Section V details the datasets, reference methods, and performance measures, while Section VI presents the experimental results. Finally, Section VII discusses the main conclusions.

II. Related Work

This section presents the preliminaries. Section II-A describes the main approaches for imbalanced classification. Then, Section II-B describes the hyper-parameter optimization problem, and Section II-C presents the main concepts related to bilevel optimization.

A. Methods for Imbalance Classification

Most learning algorithms may face difficulties when dealing with imbalanced classification problems, as they can favor the majority class, leading to an ineffective classification model. Two main approaches to dealing with imbalanced datasets are sampling strategies and algorithm adaptation [2]–[4]. The former works with training data by modifying its class distribution, while the latter adjusts the training algorithm or inference process to consider the imbalance. We describe these two approaches below.

1) Data-Level Preprocessing Methods: This approach aims to reduce the effect of class imbalance by adding or removing samples from training data to balance the class distribution [34]. There are two primary sampling strategies:

- **Oversampling** methods attempt to balance the dataset by replicating or creating samples from the minority class. Random oversampling (ROS) [3] is the simplest method for data balancing that replicates samples from the minority class. SMOTE [6] is a popular method for generating artificial samples through a linear interpolation of samples from the minority class. Variants of SMOTE include SVMSMOTE [7] and ADASYN [8].
  The major criticism of oversampling methods is that the synthetic samples can cause overfitting of the classification model.
- **Undersampling** methods balance the dataset by removing instances from the majority class. Random undersampling (RUS) [3] is an uninformed method that removes instances from the majority class at random. The condensed nearest neighbor (CNN) [9] is an informed method that eliminates examples distant from the boundary decision.
  The major criticism of undersampling methods is that they can discard meaningful instances and lead to loss of information.

It is unclear whether oversampling is better than undersampling or vice versa [35], [36]. Both methods are effective in handling imbalanced classification problems.

2) Algorithm-Level Methods: This approach aims to adapt the way a particular classifier learns in such a manner that it can deal with imbalanced problems. For SVMs, there are three main approaches of adaptation:

- **Cost-sensitive** methods consider different costs to each class during learning, such that minority class errors have a higher penalization than those of the majority class. SVMs can work in a cost-sensitive framework by using different regularization parameters for positive and negative samples [37]. Also, an

instance can be weighted based on the density of its neighborhood [38]. However, the proper setting of the costs is unknown.

- **Kernel adaptation** methods adapt the kernel function or kernel matrix to reduce the bias towards the majority class. For example, WK-SMOTE [15] expands the kernel matrix by incorporating the dot products of artificial samples generated in the feature space. Then, the SVM training algorithm uses the modified kernel matrix to learn a model.

- **Hyper-plane shifting** methods shift the separating hyper-plane to enlarge the margin around minority class [14].

The major criticism of the algorithm-level methods is that they are algorithm-specific and require in-depth knowledge of the classifier. However, these methods are more accurate than data-level methods [2].

B. Hyper-Parameter Optimization

Hyper-parameter optimization refers to the problem of automatically setting the hyper-parameter configuration of a learning algorithm to optimize the performance. Hyper-parameter optimization is a complicated problem with several challenges. The challenges include computationally expensive evaluations of the objective function, a complex and non-convex search space, hyper-parameters that cannot be differentiable, and a finite amount of data that may limit the estimation of the generalization performance [39]. Formally, the hyper-parameter optimization problem can be stated as follows [39]:

$$\theta^* = \arg\min_{\theta \in \Theta} \mathbb{E}_{D_{train}, D_{val}} \mathcal{L}\left(\mathcal{A}_\theta, D_{train}, D_{val}\right) \quad (1)$$

where $\mathcal{L}\left(\mathcal{A}_\theta, D_{train}, D_{val}\right)$ is a loss function that measures the loss of the model learned by algorithm $\mathcal{A}$ with hyper-parameters θ that is trained with D_{train} and is validated with D_{val}.

Global optimization techniques are commonly adopted to face the non-convex nature of the hyper-parameter optimization problem. For SVMs, the works on hyper-parameter optimization can be categorized into:

- **Model-free optimization** includes classical techniques such as Grid Search [40], Random Search [40], Evolutionary Algorithms [41], [42], and Particle Swarm Optimization [21].

- **Model-based optimization** includes Bayesian optimization [43], [44] and surrogate-assisted optimization [27].

Works on SVMs hyper-parameter optimization have also considered the optimization of the model pipeline (data preprocessing + learning algorithm) [42], [45], the training set selection problem [28], [46], [47], or more recently a combination of feature and training set selection together with hyper-parameter optimization [48]. However, these studies neglected the imbalanced classification problem.

C. Evolutionary Bilevel Optimization

Bilevel optimization is a hierarchical optimization problem with two levels: the upper-level, also known as the leader, and the lower-level, also called the follower. Formally, a bilevel optimization problem is stated as follows [17], [18]:

$$\begin{aligned} &\min f_u\left(\mathbf{v}_u, \mathbf{v}_l\right) \\ &\text{s.t. } \mathbf{v}_l \in \{\arg\min f_l\left(\mathbf{v}_u, \mathbf{v}_l : g_l\left(\mathbf{v}_u, \mathbf{v}_l\right) \le 0\right)\} \quad (2) \\ &\quad g_u\left(\mathbf{v}_u, \mathbf{v}_l\right) \le 0 \end{aligned}$$

where $\mathbf{v}_u$ and $\mathbf{v}_l$ are the upper-level and the lower-level variables, respectively, f_u and f_l represent the objective functions for the upper-level and lower-level, and g_u and g_l are the set of constraints for upper-level and lower-level, respectively.

The lower-level is a constraint to the upper-level optimization problem. Therefore, the lower-level solution partially determines the upper-level solution. The nested structure leads to several difficulties, such as non-linearity, non-convexity, and disconnectedness. These difficulties can be present even for the simplest bilevel problems [17], [18], [49].

EAs have shown success when dealing with complex optimization problems. Thus, EAs emerge as an alternative to deal with bilevel optimization problems. Most of the current EAs proposed to handle these problems are nested in nature. These approaches have two optimization algorithms, where one algorithm runs within the other. Overviews of evolutionary bilevel algorithms can be found in [17], [18].

Supervised learning can be treated as a bilevel problem, in which the upper-level optimizes the hyperparameters that minimize the expected generalization error, and the lower-level learns the parameters [50]. Next, we explain the bilevel formulation for learning parameters and hyper-parameters for an SVM with cost-sensitive.

III. Bilevel Cost-Sensitive Support Vector Machine: Optimization Problem

The bilevel formulation breaks the problem down into two levels: the upper-level concerned with the hyper-parameters configuration and the lower-level with the SVM training. In this section, we explain the optimization objectives at each level. First, Section III-A defines the lower-level that finds the optimal separating hyper-plane when training the SVM. Next, Section III-B explains the objective function for the upper-level, which optimizes the classification performance considering the uneven class distributions for a given hyper-parameter configuration.

A. Lower-Level – Optimizing Parameters

The lower-level problem focuses on finding the support vectors that define the hyper-plane for an imbalanced classification problem. A cost-sensitive SVM penalizes the errors differently for positive and negative classes. This is formulated as follows [37]:

$$\min \frac{1}{2} \parallel \mathbf{w} \parallel^2 + C^+ \sum_{i:y=1} \xi_i + C^- \sum_{i:y=-1} \xi_i$$
$$\text{subject to } y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b \right) \geq 1 - \xi_i \qquad (3)$$
$$\xi_i \geq 0$$

where $\langle \cdot, \cdot \rangle$ represents the dot product, $\mathbf{w}$ is the separating hyper-plane, and C^+ and C^- are the costs for the positive and negative samples, respectively.

The dual problem obtained through Lagrange multipliers is as follows:

$$\max \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j}^{n} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$
$$\text{subject to } \sum_{i=1}^{n} y_i \alpha_i = 0 \qquad (4)$$
$$0 \leq \alpha_i \leq C^+, i : y = +1$$
$$0 \leq \alpha_i \leq C^-, i : y = -1$$

For learning nonlinear functions, the dot product $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ is replaced by a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$. The Radial basis function (RBF) is a kernel function that is highly effective and theoretically supported [51], [52]. The RBF kernel is defined as:

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma \parallel \mathbf{x}_i - \mathbf{x}_j \parallel^2} \qquad (5)$$

where γ is an adjustable parameter given by the upper-level.

Section III-B explains the upper-level optimization problem that optimizes the hyper-parameters C^+, C^-, and the γ value for the RBF kernel.

B. Upper-Level – Optimizing Hyper-Parameters

The upper-level optimizes the hyper-parameters used in the lower-level to learn the support vectors. Let λ be a vector that encodes the hyper-parameters C^+, C^-, and γ. The goal of the upper-level is to find the set of hyper-parameters that gets the minimum generalization error on imbalanced classification problems, which is estimated using the balanced error rate (BER) score. The BER is defined as:

$$BER(\lambda, \alpha^*, b) = \frac{1}{2} \left(\frac{FN}{TP + FN} + \frac{FP}{FP + TN} \right) \qquad (6)$$

where TP and TN are, respectively, the number of positive and negative samples correctly classified; and FN and FP are, respectively, the number of positive and negative samples incorrectly classified.

Computing BER using the training set can lead to overfitting. K-fold cross-validation is commonly used to assess the expected performance and to reduce the risk of overfitting. However, this procedure can become computationally inefficient, as it implies solving the lower-level problem K times for each configuration of hyper-parameters λ. We face that disadvantage by approximating the bound on the

leave-one-out cross-validation for the upper-level. Based on the lower-level solution α^*, the predicted value for the jth training sample is given by:

$$f(\mathbf{x}_j) = \sum_{i=1}^{n} \alpha_i^* y_i K(\mathbf{x}_j, \mathbf{x}_i) + b \qquad (7)$$

where b is the bias term and is set to satisfy the Karush-Kuhn-Tucker condition.

Eliminating a non-support vector from the training set does not affect the model; therefore, we focus on support vectors, as they can contribute to the error. Assuming that the set of support vectors remains the same during the leave-one-out procedure, we can approximate the output when removing the jth support vector from the training set as:

$$\hat{f}(\mathbf{x}_j) = \sum_{i=1}^{n} \alpha_i^* y_i K(\mathbf{x}_j, \mathbf{x}_i) + b - \alpha_j^* y_j K(\mathbf{x}_j, \mathbf{x}_j) \qquad (8)$$

In the case of the RBF kernel, the term $K(\mathbf{x}_j, \mathbf{x}_j)$ equals one. Simplifying (8) and multiplying it by y_j, an instance $\mathbf{x}_j$ is incorrectly classified if:

$$y_j \left(\sum_{i=1}^{n} \alpha_i^* y_i K(\mathbf{x}_i, \mathbf{x}_j) + b - \alpha_j^* y_j \right) < 0 \qquad (9)$$

After determining the incorrectly classified instances with (9), the BER of each individual at the upper-level is determined using (6).

After solving the bilevel optimization problem, the optimal support vectors are used to classify a new instance using (7).

IV. EBCS-SVM: Evolutionary Bilevel Cost-Sensitive Support Vector Machines

EBCS-SVM aims to build an optimal SVM model for handling imbalanced classification problems through bilevel optimization. EBCS-SVM determines the hyper-parameters values, i.e., the costs of each class and the kernel parameters that minimize BER at the upper-level, and the lower-level finds the optimal separating hyper-plane. Fig. 1 graphically depicts the bilevel interaction between the lower-level and upper-level in EBCS-SVM.

Algorithm 1 describes EBCS-SVM, and it works as follows:

1) In line 1, a population for the upper-level is randomly created with u_s individuals and three variables representing the costs for the positive and the negative class, and the γ value for RBF kernel, based on the bounds given in Section IV-B.
2) In lines 2–5, each upper-level solution is evaluated. To this end, the following steps are carried out:
   a) The lower-level problem described in Section III-A is solved using the SMO algorithm and the given hyper-parameters.
   b) The resulting SVM model is evaluated by computing the BER, as described in Section III-B.
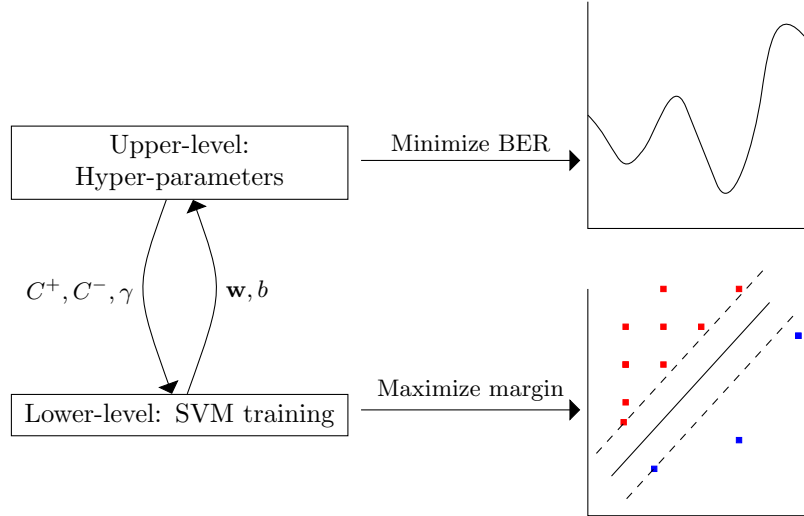
Fig. 1: Bilevel scheme to learn support vectors machine in a cost-sensitive approach. The figure depicts the dependency between hyper-parameters and support vectors, optimizing the margin with different penalties.

Algorithm 1 EBCS-SVM

Require: $\mathcal{X}$, the set of samples,
    y, the set of classes labels,
    u_s, upper population size,
    m_e, maximum number of evaluations at the upper-level,
    τ, tolerance threshold for lower-level,
    m, maximum number of iterations for lower-level.
Ensure: The set of support vectors
 1: Generate randomly an initial population of hyper-parameters, $\mathcal{P}_u$, with u_s individuals
 2: **for** each individual p_u in $\mathcal{P}_u$ **do**
 3:    Use SMO with the hyper-parameters defined in p_u to find the optimal support vectors.
 4:    Compute the BER for the upper-level individual using the optimal support vectors.
 5: **end for**
 6: **while** a stopping criterion is not met **do**
 7:    Apply evolutionary operator to produce an upper-level offspring population, $\mathcal{O}_u$.
 8:    **for** each individual o_u in $\mathcal{O}_u$ **do**
 9:       Find the nearest neighbors of o_u in $\mathcal{P}_u$.
10:       Warm starting the set of support vectors of the lower-level based on the nearest neighbors.
11:       Use SMO with the hyper-parameters defined in o_u to find the optimal support vectors.
12:       Compute the BER for the upper-level offspring using the optimal support vectors.
13:    **end for**
14: **end while**
15: Choose the solution with the lowest BER as the optimal SVM for handling imbalanced datasets. If there are more than one solution with similar BER to the lowest one, choose the one with the lower number of support vectors.

---

  3) In lines 6–14, the evolutionary process takes place.

    a) In line 7, a new population of hyper-parameters is created by applying evolutionary operators over $\mathcal{P}_u$ using the adaptation of DE operators proposed by SHADE [53].

    b) In line 9, the nearest neighbors in $\mathcal{P}_u$ are found for each individual in $\mathcal{O}_u$. In line 10, the set of support vectors for the lower-level is warm started by considering the nearest neighbors of the upper-level to determine how probable an instance is a support vector, as described in Section IV-A.

    c) Line 11 solves the lower-level optimization problem with the SMO algorithm and, line 12 computes the BER score for the given hyper-parameters.

A. Lower-level Initialization

When solving the lower-level using the hyper-parameter of the initial upper-level population, all training instances are initially assumed to be a support vector, and the SMO solves the lower-level problem. The solutions obtained by the initial configuration of hyper-parameters are stored to perform a warm starting of the lower-level.

The warm starting takes place after the first generation of the upper-level and works as follows. First, for a new given hyper-parameters configuration, its m nearest-neighbors are found among the hyper-parameters from the current population. After, the set of support vectors is retrieved and used to determine the chance of an instance becoming a support vector, based on the relative frequency of its m nearest-neighbors. Finally, the SMO algorithm solves the lower-level based on previous initialization. The premise for using this initialization is that similar configurations of hyper-parameters lead to a similar set of support vectors.

The value of m is determined during the search. For doing so, the number of neighbors is randomly selected between $[1, u_s]$ with uniform probability. Then, the probability of each value of m is updated based on the normalized frequency of success of such value, i.e., an improvement in the BER score.

B. Representation and Evolutionary Operators

At the upper-level, individuals encode the hyper-parameters as a real-value vector. Initially, C^+ and C^- are randomly generated in the range $\left[2^{-5}, 2^{10}\right]$ and γ between $\left[2^{-10}, 2^5\right]$. Individuals at this level are then evolved using DE [54], [55]. However, the classical DE requires the definition of two parameters: the differential weights (F) and the crossover rate (CR). For this reason, we adopt a self-adaptative variant called SHADE [53], which uses history information to adapt the values of F and CR during the search and a greedy current-to-best mutation strategy. Therefore, these parameters are learned during the search.

After a child solution of hyper-parameters is created and its BER value is computed, it competes with the current individual to determine which one is better and therefore survives. A child solution is better if its BER value is lower than the BER value of the current individual, or if the BER values of both solutions are equivalent, but the child has fewer support vectors.

V. Experimental Settings

In this section, we describe the configuration setup for our experimental study. In Section V-A, we present the set of benchmark datasets considered for experimentation. Section V-B provides the metrics used to assess the performance of all methods and the test to analyze them. Finally, Section V-C details the state-of-the-art techniques used to compare the performance and their respective tuning of hyper-parameters.

A. Datasets

The performance of EBCS-SVM is assessed using a benchmark of 70 datasets from the KEEL repository [56]. Table I details the characteristics of the datasets, including the number of instances (Inst.), the number of features (Feat.), and the imbalance ratio (IR)[2]. These datasets are diverse in the IR, number of samples, and number of features. Thus, the performance is assessed using problems with different characteristics. We divided the datasets into three groups based on the degree of IR: ten datasets with small IR, i.e., the IR is less than or equal to three; 35 datasets with medium IR, that is an IR higher than three and less than or equal to 20; and 25 datasets with high IR, i.e., the IR is higher than 20.

Datasets were partitioned using the 10×5-fold cross-validation. In the 5-fold cross-validation, the dataset is randomly split into five disjoint subsets. In each fold, a subset is used as the test set and the remaining as the training set. Then, 5-fold cross-validation is repeated ten times, with a different split each time. Thus, each dataset is tested 50 times.

[2]IR is defined as the ratio between the number of samples in the majority class and the number of samples in the minority. Therefore, the higher the IR, the greater the imbalance.

TABLE I: Description of datasets based on the number of instances (Inst.), number of features (Feat.), and the imbalance ratio (IR).

ID	Dataset	Inst.	Feat.	IR
	Small IR			
1	ecoli-0_vs_1	220	7	1.857
2	glass0	214	9	2.057
3	glass1	214	9	1.816
4	haberman	306	3	2.778
5	iris0	150	4	2.000
6	pima	768	8	1.866
7	vehicle1	846	18	1.029
8	vehicle2	846	18	2.881
9	vehicle3	846	18	1.029
10	wisconsin	683	9	1.858
	Medium IR			
11	abalone9-18	731	8	16.405
12	cleveland-0_vs_4	173	13	12.308
13	dermatology-6	358	34	16.900
14	ecoli-0-1-4-6_vs_5	280	6	13.000
15	ecoli-0-1-4-7_vs_2-3-5-6	336	7	10.586
16	ecoli-0-1-4-7_vs_5-6	332	6	12.280
17	ecoli-0-1_vs_2-3-5	244	7	9.167
18	ecoli-0-3-4-7_vs_5-6	257	7	9.280
19	ecoli-0-3-4_vs_5	200	7	9.000
20	ecoli-0-6-7_vs_5	220	6	10.000
21	ecoli1	336	7	3.364
22	ecoli2	336	7	5.462
23	ecoli3	336	7	8.600
24	ecoli4	336	7	15.800
25	glass-0-1-2-3_vs_4-5-6	214	9	3.196
26	glass-0-1-5_vs_2	172	9	9.118
27	glass-0-1-6_vs_5	184	9	19.444
28	glass-0-4_vs_5	92	9	9.222
29	glass-0-6_vs_5	108	9	11.000
30	glass2	214	9	11.588
31	glass4	214	9	15.462
32	glass6	214	9	6.379
33	led7digit-0-2-4-5-6-7-8-9_vs_1	443	7	8.630
34	new-thyroid1	215	5	5.143
35	newthyroid2	215	5	5.143
36	page-blocks-1-3_vs_4	472	10	15.857
37	segment0	2308	19	6.015
38	shuttle-c0-vs-c4	1829	9	13.870
39	vehicle0	846	18	3.251
40	vowel0	988	13	9.978
41	yeast-0-3-5-9_vs_7-8	506	8	9.120
42	yeast-0-5-6-7-9_vs_4	528	8	9.353
43	yeast-1_vs_7	459	7	14.300
44	yeast-2_vs_4	514	8	9.078
45	yeast3	1484	8	8.104
	High IR			
46	abalone-17_vs_7-8-9-10	2338	8	39.310
47	abalone-20_vs_8-9-10	1916	8	72.692
48	abalone-21_vs_8	581	8	40.500
49	abalone-3_vs_11	502	8	32.467
50	abalone19	4174	8	129.438
51	car-good	1728	6	24.043
52	flare-F	1066	11	23.791
53	glass5	214	9	22.778
54	poker-8-9_vs_5	2075	10	82.000
55	poker-8-9_vs_6	1485	10	58.400
56	poker-8_vs_6	1477	10	85.882
57	poker-9_vs_7	244	10	29.500
58	shuttle-6_vs_2-3	230	9	22.000
59	winequality-red-3_vs_5	691	11	68.100
60	winequality-red-4	1599	11	29.170
61	winequality-red-8_vs_6-7	855	11	46.500
62	winequality-red-8_vs_6	656	11	35.444
63	winequality-white-3-9_vs_5	1482	11	58.280
64	winequality-white-3_vs_7	900	11	44.000
65	yeast-1-2-8-9_vs_7	947	8	30.567
66	yeast-1-4-5-8_vs_7	693	8	22.100
67	yeast-2_vs_8	482	8	23.100
68	yeast4	1484	8	28.098
69	yeast5	1484	8	32.727
70	yeast6	1484	8	41.400

B. Performance Metrics and Statistical Tests

We assessed the performance of the methods using a set of metrics well-suited for imbalanced classification problems [57]. Let sensitivity (sen) and specificity (spe) be defined as:

$$sen = \frac{TP}{TP + FN}$$
$$spe = \frac{TN}{TN + FP} \tag{10}$$

where TP and TN are, respectively, the number of positive and negative samples correctly classified; and FN and FP are, respectively, the number of positive and negative samples incorrectly classified.

The metrics described in [57] can be defined in terms of specificity and sensitivity. These metrics are listed below:

- **BAR**, the balanced accuracy rate is equivalent to 1-BER and is defined as the average between sensitivity and specificity, i.e.,

$$\text{BAR} = \frac{1}{2}\left(sen + spe\right) \tag{11}$$

- **BMI**, the bookmarker informedness is defined as the sum of sensitivity and specificity minus one, i.e.,

$$\text{BMI} = sen + spe - 1 \tag{12}$$

- **GM** indicates the geometric mean between sensitivity and specificity, i.e.,

$$\text{GM} = \sqrt{sen \cdot spe} \tag{13}$$

- **uF1** [57] is the unbiased version of the F1 score and is defined as two times the sensitivity between the sum of two plus sensitivity minus specificity, i.e.,

$$\text{uF1} = \frac{2 \cdot sen}{2 + sen - spe} \tag{14}$$

- **uMCC** [57] is the unbiased version of the Matthews correlation coefficient, and is defined as follows:

$$\text{uMCC} = \frac{sen \cdot spe - 1}{\sqrt{1 - (sen - spe)^2}} \tag{15}$$

The Bayesian hypothesis tests are used to analyze EBCS-SVM regarding reference methods. They allow comparing the difference in the results achieved by two algorithms, estimating the posterior probabilities that one algorithm is better than the other and that both are practically equivalent. These tests are not affected by the number of datasets. Moreover, they can provide more information than the null hypothesis significance test, even when the latter does not reject the null hypothesis [58], [59]. We used the hierarchical Bayesian test to analyze the results, as it considers both the mean and the variance through the cross-validation partitions for each dataset. In the analysis, we considered that two methods are equivalent if the difference is below 0.01.

C. Reference Methods

We compared EBCS-SVM with several state-of-the-art techniques for imbalanced classification. To this end, we considered methods for both data-level preprocessing and algorithm-level. Specifically, the comparative study considers the following methods.

- **SVM**. The standard SVM (BL) is used on the dataset without preprocessing or modification to weight the class distributions.
- **Data-level methods (DL)**. The methods in this group are used to preprocess the data. Then, the edited dataset is used to train an SVM. This group consists of ROS [3], SMOTE [6], SVMSMOTE [7], RUS [3], and CNN [9].
- **Algorithm-level methods (AL)**. This group encompasses SVMDC [37], uNBSVM [14], WK-SMOTE [15], CSSVM [60], and RBI-LP-SVM [16].

Reference methods require defining a set of hyper-parameters to use in training. Properly selecting hyper-parameters is a crucial step to compare classification algorithms [51], [61]. We adopted the RBF kernel because of its effectiveness with SVM. For the sake of a fair comparison, the hyper-parameters of each method were optimized for each dataset independently by optimizing the BER, computed through an internal stratified 5-fold cross-validation on the training set. The set of hyper-parameters includes: the γ value optimized in the range of $\left[2^{-10}, 2^5\right]$ for all methods; the regularization parameter C in the range of $\left[2^{-5}, 2^{10}\right]$ for SVM, data-level methods, and CSSVM; the regularization parameter for positive (C^+) and negative (C^-) class optimized between $\left[2^{-5}, 2^{10}\right]$ for both WK-SMOTE and SVMDC; the regularization parameter for synthetic samples (C^s) optimized between $\left[2^{-5}, 2^{10}\right]$ for WK-SMOTE; the weight factor for positive (ω_p) and negative (ω_n) class in the range of $[0, 1]$ for uNBSVM; the cost-sensitive parameter (κ) ranges from zero to one and the margin violation weight (C_1) is the range of $[1, 10]$ for CSSVM; the amount of sampling is searched in the range of $\left[\frac{n_m+1}{n_M}, 1\right]$, with n_m and n_M as the number of samples in the minority and majority class, respectively, for data-level methods and WK-SMOTE, and the number of neighbors is between $\left[1, \frac{n_m}{2}\right]$ for SMOTE, SVMSMOTE, and WK-SMOTE. SHADE was also used to optimize the hyper-parameters, which ran with a population size equals to 30 and the stopping criteria considered performing 1,000 fitness function evaluations or that the standard deviation of fitness values of the population is below 0.001.

We provide the implementation of EBCS-SVM, datasets, splits, and detailed results in each partition as supplementary material. The supplementary material can be downloaded at www.cimat.mx/~alejandro.rosales/resources/EBCSSVM.tar.gz.

VI. Experimental Results and Discussion

This section presents the experimental results reported by EBCS-SVM and reference methods. Section VI-A shows the results obtained with datasets with small IR.

Next, Section VI-B considers the 35 datasets with medium IR, and Section VI-C presents the results using the 25 datasets with high IR. Finally, Section VI-D compares the training time required by each method.

A. Experiments on Small Imbalance Ratio

In this section, we focused on comparing the performance of EBCS-SVM against reference methods. Table I shows the results on the ten datasets with a small IR. The reported results correspond to the average and standard deviations for BAR, BMI, GM, uF1, and uMCC scores. Fig. 2 graphically depicts barycenter plots for the posterior probabilities reported by the hierarchical Bayesian test for the BAR score. Each point on the barycenter plot represents an estimate of the probability that it belongs to each region. Based on the results and the analysis carried out by the hierarchical Bayesian test, the following observations are highlighted:

- Most methods showed a competitive performance when dealing with small IR.
- For metrics BAR, GM, and uF1, most methods reported scores above 0.820. The exceptions were WK-SMOTE and RBI-LP-SVM, which obtained performances below 0.800. On the other hand, for metrics BMI and MCC, most methods reported scores above 0.710, except for WK-SMOTE, nNBSVM, and RBI-LP-SVM.
- CSSVM obtained the highest performance in all metrics. SVMDC was the second-best position for BAR, BMI, GM, and uMCC, and the third-best for uF1. On the other hand, EBCS-SVM ranked second-best for uF1; it was in the seventh position for BAR, BMI, and uMCC; and in the eighth position for GM.
- The hierarchical Bayesian tests provided strong evidence on the practical equivalence between CSSVM and EBC-SVM. We can observe similar behavior when EBCS-SVM is compared with SVM, ROS, SMOTE, SVMSMOTE, RUS, CNN, and SVMDC. Thus, EBCS-SVM exhibited a performance practically similar to that of CSSVM, the best-ranked method.
- Among data-level methods, ROS had the best average performance over the ten datasets with a small IR for BAR, BMI, GM, and uMCC, while for uF1 CNN was the best data-level approach. Conversely, CNN was the worst data-level method for BAR, DMI, GM, and uMCC, while SVMSMOTE was the worst for uF1.
- RBI-LP-SVM showed the lowest performance among algorithm-level methods. The hierarchical Bayesian test reported probabilities above 0.999 in the region of EBCS-SVM for all metrics.
- Algorithm-level methods generally reported better performance than data-level methods.

EBCS-SVM, uNBSVM, CSSVM, and ROS were highly effective methods to classify problems with a small IR. In the next section, we delved into our analysis when the IR increases.

B. Experiments with Medium Imbalance Ratio

In this section, we analyzed the performance of EBCS-SVM and reference methods when using the 35 benchmark datasets with medium IR. Table II presents the average results obtained by each method, and Fig. 3 shows the posterior plots for the BAR score when EBCS-SVM is compared with reference methods. From these, the following is pointed out:

- Most methods reported results above 0.800 for BAR, GM, and uF1 scores, except for SVM for GM and uF1; WK-SMOTE for BAR, GM, and uF1; uNBSVM for GM and uF1; and RBI-LP-SVM for BAR, GM, and uF1.
- Regarding BMI and uMCC, most methods reported performances above 0.700. The exceptions were SVM for BMI and WK-SMOTE, uNBSVM, and RBI-LP-SVM for BMI and uMCC.
- ROS achieved the highest performance for BAR, BMI, GM, and uMCC and the second-best performance for uF1. RUS achieved the best performance for uF1 and the second-best performance for BAR, BMI, GM, and MCC. EBCS-SVM ranked the third-best for all metrics.
- The hierarchical Bayesian analysis revealed a probability above 0.900 in the region of practical equivalence when EBCS-SVM was compared with ROS, SMOTE, SVMSMOTE, and RUS for all metrics. Thus, there is evidence in favor of the competitiveness of these methods for handling medium IR problems. We can observe this behavior in Fig. 3, where we can note that the center mass is in the region of practical equivalence. Although for CNN and SVMDC, the center mass fell in the practically equivalent region, the posterior plot distribution was spread throughout the area of EBCS-SVM, providing at some extent evidence in favor of EBCS-SVM.
- ROS stood out as the most effective data-level method.
- Among algorithm-level methods, EBCS-SVM obtained the best performance, and CSSVM was the second-best. Conversely, RBI-LP-SVM ranked in the last position.

For datasets with a medium IR, EBCS-SVM, ROS, and RUS were the most superior methods. ROS is highlighted as a prominent method for problems with medium IR. CSSVM remained a competitive method. In the next section, methods are evaluated when handling datasets with a high IR.

C. Experiments with High Imbalance Ratio

In this section, we considered the 25 datasets with an IR above 20 to analyze the performance of EBCS-SVM and reference methods. Table III reports the average results for all metrics, and Fig. 4 shows the posterior probabilities obtained with the hierarchical Bayesian test. Based on these results, we remark the following:

TABLE I: Obtained results on small IR datasets for BAR, BMI, GM, uF1, and uMCC scores.

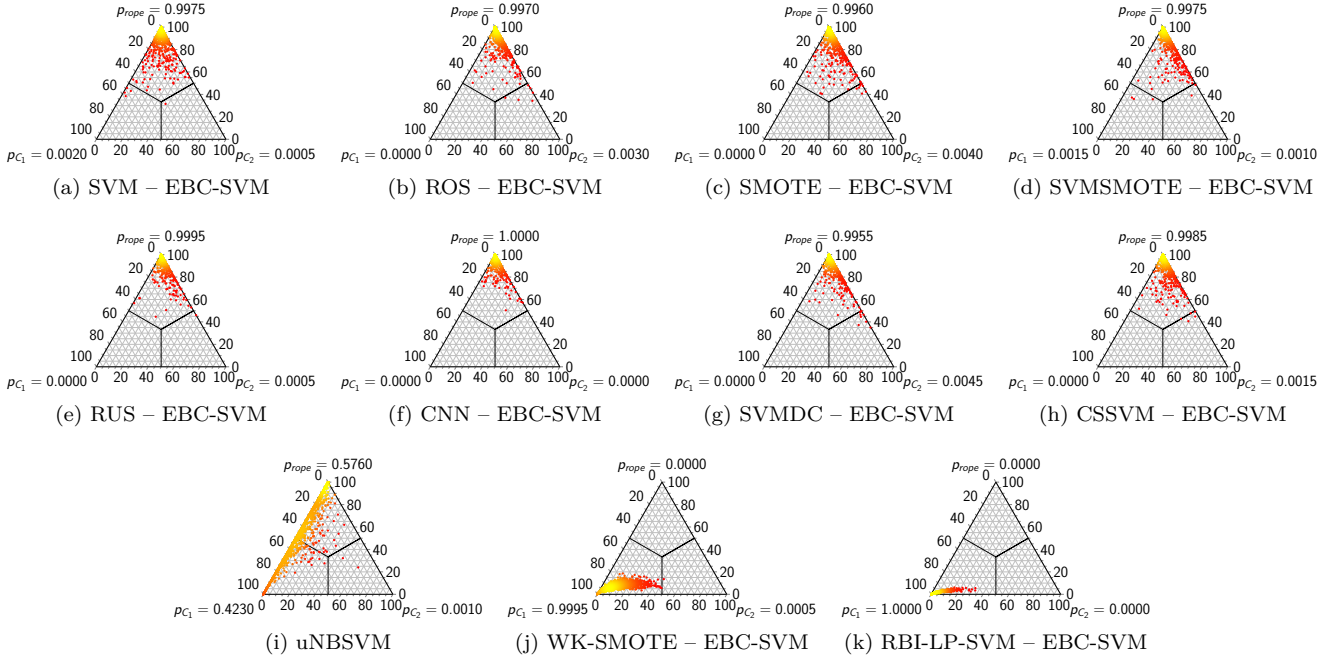| Fam.	Method	BAR	BMI	GM	uF1	uMCC
BL	SVM	0.8626 ± 0.1461	0.7251 ± 0.2922	0.8494 ± 0.1651	0.8312 ± 0.1904	0.7343 ± 0.2831
DL	ROS	0.8726 ± 0.1321	0.7452 ± 0.2643	0.8643 ± 0.1430	0.8515 ± 0.1600	0.7528 ± 0.2554
DL	SMOTE	0.8719 ± 0.1328	0.7439 ± 0.2656	0.8632 ± 0.1447	0.8502 ± 0.1628	0.7512 ± 0.2569
DL	SVMSMOTE	0.8723 ± 0.1345	0.7447 ± 0.2690	0.8627 ± 0.1487	0.8499 ± 0.1681	0.7519 ± 0.2601
DL	RUS	0.8712 ± 0.1335	0.7424 ± 0.2670	0.8636 ± 0.1442	0.8548 ± 0.1560	0.7489 ± 0.2593
DL	CNN	0.8688 ± 0.1345	0.7377 ± 0.2690	0.8616 ± 0.1448	0.8586 ± 0.1506	0.7441 ± 0.2613
AL	SVMDC	0.8730 ± 0.1324	0.7460 ± 0.2649	0.8644 ± 0.1446	0.8552 ± 0.1570	0.7529 ± 0.2569
AL	CSSVM	$\mathbf{0.8740 \pm 0.1267}$	$\mathbf{0.7480 \pm 0.2533}$	$\mathbf{0.8689 \pm 0.1327}$	$\mathbf{0.8674 \pm 0.1357}$	$\mathbf{0.7535 \pm 0.2481}$
AL	WK-SMOTE	0.7322 ± 0.1137	0.4643 ± 0.2275	0.6194 ± 0.1743	0.5753 ± 0.1930	0.5069 ± 0.2181
AL	uNBSVM	0.8419 ± 0.1325	0.6839 ± 0.2650	0.8257 ± 0.1479	0.8415 ± 0.1310	0.6960 ± 0.2582
AL	RBI-LP-SVM	0.5293 ± 0.0568	0.0587 ± 0.1136	0.0676 ± 0.1136	0.6724 ± 0.0474	0.0604 ± 0.1135
AL	EBCS-SVM	0.8696 ± 0.1310	0.7391 ± 0.2619	0.8585 ± 0.1473	0.8622 ± 0.1419	0.7468 ± 0.2536

Fig. 2: Posterior distribution for BAR when comparing EBCS-SVM with the reference methods on small IR datasets. The region of the bottom-left represents EBCS-SVM, the region at the top is for rope, and the region in the bottom-right represents the reference method.

- EBCS-SVM excelled in dealing effectively with highly imbalanced datasets in all metrics. Ergo, the superiority of EBCS-SVM is stressed when the IR is increased.
- Among data-level methods, RUS had the best score in all metrics and ranked second-best position globally. ROS was the second-best data-level method and ranked third-best position globally.
- Among algorithm-level methods, EBCS-SVM was the best one, followed by SVMDC. However, when observing the posterior probabilities reported by the hierarchical Bayesian test, we noted that the center mass is in the region of EBCS-SVM. Furthermore, the posterior odds revealed strong evidence in favor of EBCS-SVM. We observed this behavior for all metrics.
- Posterior probabilities plots also showed that, for most reference methods, the center mass fell in the region of EBCS-SVM. The exception was RUS, whose center mass is in the region of practical equivalence;

however, its distribution spreads in the regions of both EBCS-SVM and RUS.

As the IR increased, the performance of EBCS-SVM stood out over the reference methods, regardless of the adopted metric. The hierarchical Bayesian test reported high probabilities in favor of EBCS-SVM that supported these observations.

D. Computational Time

In this section, we analyze the training time for each method. We used the performance profile [62], which represents the cumulative distribution on a performance metric. The performance profile is constructed for the necessary training time required by each method to optimize the hyper-parameters and learn the classification model. Fig. 5 depicts the performance profile for all methods. The $y-$axis represents the probability $(\rho(\tau))$ that a method can learn a model within a factor τ times the

TABLE II: Obtained results on medium IR datasets for BAR, BMI, GM, uF1, and uMCC scores.

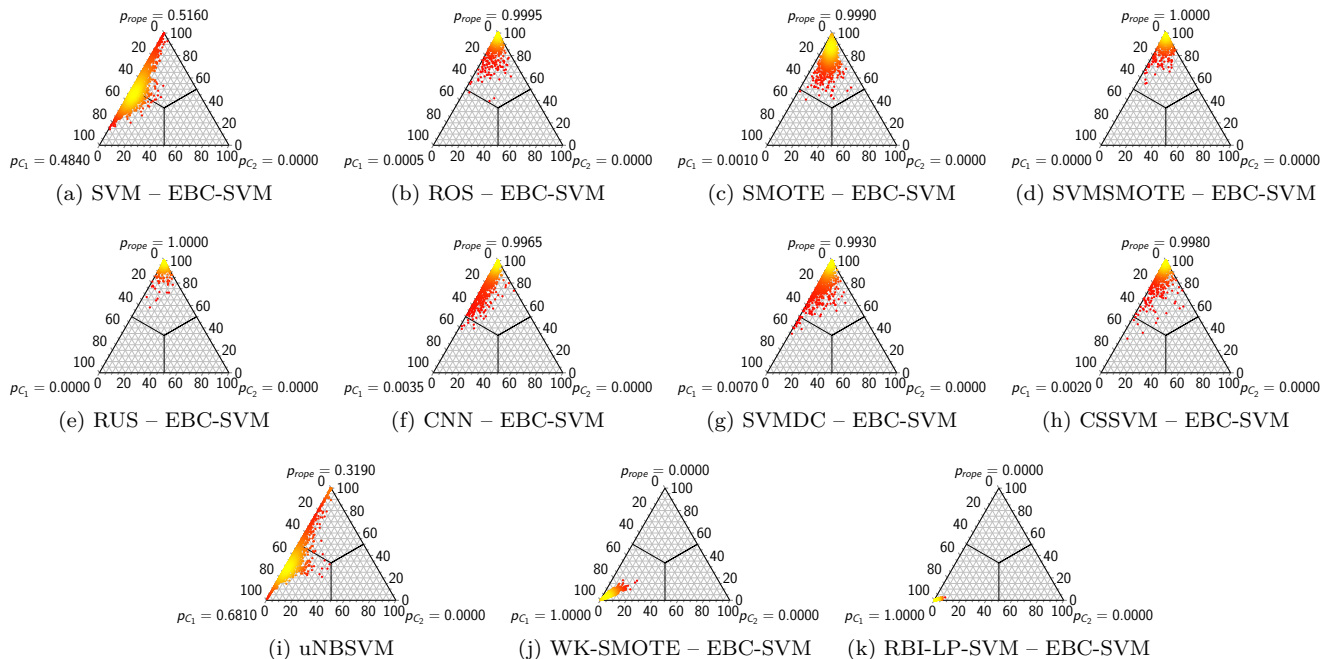| Fam.	Method	BAR	BMI	GM	uF1	uMCC
BL	SVM	0.8420 ± 0.1226	0.6841 ± 0.2453	0.7931 ± 0.1929	0.7724 ± 0.2107	0.7090 ± 0.2314
DL	ROS	$\mathbf{0.8832 \pm 0.0728}$	$\mathbf{0.7664 \pm 0.1457}$	$\mathbf{0.8693 \pm 0.0808}$	0.8579 ± 0.0892	$\mathbf{0.7813 \pm 0.1395}$
DL	SMOTE	0.8738 ± 0.0974	0.7475 ± 0.1948	0.8455 ± 0.1656	0.8345 ± 0.1687	0.7613 ± 0.1921
DL	SVMSMOTE	0.8678 ± 0.0942	0.7356 ± 0.1884	0.8329 ± 0.1549	0.8216 ± 0.1599	0.7496 ± 0.1845
DL	RUS	0.8793 ± 0.0858	0.7586 ± 0.1716	0.8652 ± 0.1025	$\mathbf{0.8603 \pm 0.1073}$	0.7690 ± 0.1672
DL	CNN	0.8626 ± 0.0974	0.7252 ± 0.1948	0.8388 ± 0.1262	0.8254 ± 0.1395	0.7395 ± 0.1894
AL	SVMDC	0.8635 ± 0.0808	0.7271 ± 0.1616	0.8334 ± 0.1181	0.8198 ± 0.1244	0.7454 ± 0.1558
AL	CSSVM	0.8689 ± 0.0795	0.7378 ± 0.1590	0.8435 ± 0.1141	0.8307 ± 0.1207	0.7548 ± 0.1536
AL	WK-SMOTE	0.7651 ± 0.1131	0.5302 ± 0.2263	0.6523 ± 0.2231	0.6250 ± 0.2274	0.5646 ± 0.2269
AL	uNBSVM	0.8315 ± 0.0856	0.6629 ± 0.1712	0.7927 ± 0.1216	0.7884 ± 0.1253	0.6852 ± 0.1664
AL	RBI-LP-SVM	0.5427 ± 0.1139	0.0855 ± 0.2278	0.0923 ± 0.2294	0.6926 ± 0.0752	0.0870 ± 0.2279
AL	EBCS-SVM	0.8784 ± 0.1051	0.7569 ± 0.2102	0.8581 ± 0.1440	0.8522 ± 0.1502	0.7671 ± 0.2052

Fig. 3: Posterior distribution for BAR when comparing EBCS-SVM with the reference methods on medium IR datasetss. The region of the bottom-left represents EBCS-SVM, the region at the top is for rope, and the region in the bottom-right represents the reference method.

fastest method, and the x−axis represents the τ factor. Thus, $\rho(1)$ indicates the probability where a given method achieves the lowest training time among all methods.

From Fig. 5, we can observe that both RUS and EBCS-SVM exhibited the best training times. From the value of $\rho(1)$, it is observed that EBCS-SVM had the highest probability (0.58) of being the fastest one, while RUS had the second-highest probability (0.30). Furthermore, algorithm-level methods generally required less training time than data-level methods, which can be observed in the near-zero probability of oversampling techniques with a value of τ equals one. The outstanding performance of RUS in training time is because by removing training instances, the SVM algorithm works with a reduced number of samples and reduces the computational time. Although CNN is also an undersampling method, the way in how instances to remove are selected slows down the training time. On the other hand, EBCS-SVM exploits the information of both levels to feedback with the information of previous

support vectors of similar solutions and improving the convergence in solving the lower-level. Finally, RBI-LP-SVM showed the worst performance profile.

VII. CONCLUSIONS

This paper introduced EBCS-SVM for learning an SVM in imbalanced scenarios. EBCS-SVM formulated the optimization of hyperparameters and support vectors as a bilevel optimization problem and showed to be able to handle imbalanced classification problems effectively and freed practitioners from defining the optimal cost to each class. To this end, an EA at the upper-level and the SMO at the lower-level interplay, such that lower-level solutions impact the BER of the upper-level solutions, and previous hyper-parameters help initialize the set of support vectors. Thus, there is a dual enrichment in the two levels.

The efficacy of EBCS-SVM was assessed using 70 benchmark datasets and compared with those of state-of-the-art

TABLE III: Obtained results on high IR datasets for BAR, BMI, GM, uF1, and uMCC scores.

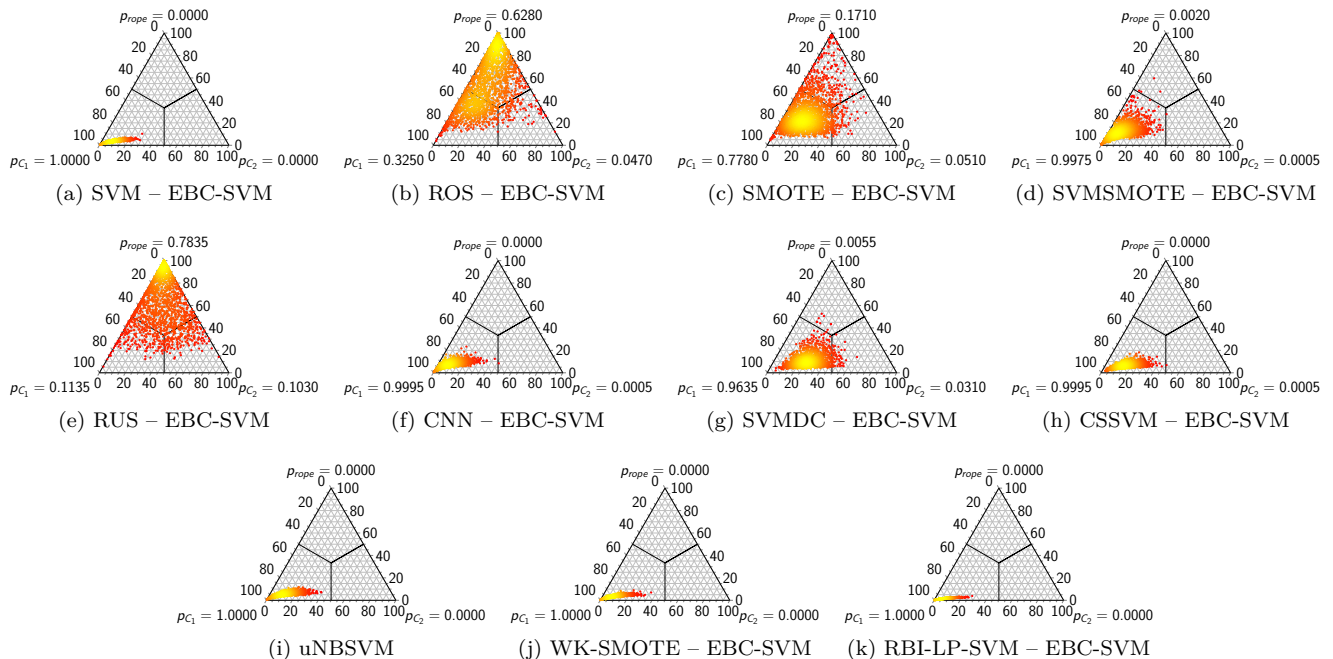| Fam.	Method	BAR	BMI	GM	uF1	uMCC
BL	SVM	0.6712 ± 0.1557	0.3423 ± 0.3114	0.4413 ± 0.3099	0.4087 ± 0.3149	0.3725 ± 0.3105
DL	ROS	0.7754 ± 0.1119	0.5508 ± 0.2239	0.6984 ± 0.1770	0.6762 ± 0.1844	0.5665 ± 0.2244
DL	SMOTE	0.7644 ± 0.1315	0.5288 ± 0.2629	0.6665 ± 0.2387	0.6452 ± 0.2426	0.5436 ± 0.2649
DL	SVMSMOTE	0.7331 ± 0.1437	0.4661 ± 0.2873	0.5907 ± 0.2766	0.5652 ± 0.2811	0.4856 ± 0.2880
DL	RUS	0.7803 ± 0.1132	0.5606 ± 0.2264	0.7220 ± 0.1726	0.7113 ± 0.1716	0.5692 ± 0.2274
DL	CNN	0.7203 ± 0.1412	0.4406 ± 0.2824	0.6030 ± 0.2297	0.5740 ± 0.2391	0.4535 ± 0.2899
AL	SVMDC	0.7340 ± 0.1500	0.4680 ± 0.3000	0.5825 ± 0.3144	0.5655 ± 0.3118	0.4849 ± 0.3037
AL	CSSVM	0.7031 ± 0.1640	0.4062 ± 0.3280	0.5003 ± 0.3371	0.4773 ± 0.3394	0.4257 ± 0.3309
AL	WK-SMOTE	0.6604 ± 0.1415	0.3208 ± 0.2830	0.4119 ± 0.3016	0.3964 ± 0.3024	0.3392 ± 0.2877
AL	uNBSVM	0.7054 ± 0.1321	0.4108 ± 0.2642	0.5492 ± 0.2659	0.5410 ± 0.2642	0.4278 ± 0.2678
AL	RBI-LP-SVM	0.5470 ± 0.1034	0.0940 ± 0.2069	0.1440 ± 0.2709	0.6410 ± 0.1298	0.0977 ± 0.2153
AL	EBCS-SVM	$\mathbf{0.7972 \pm 0.1234}$	$\mathbf{0.5944 \pm 0.2468}$	$\mathbf{0.7482 \pm 0.1732}$	$\mathbf{0.7430 \pm 0.1757}$	$\mathbf{0.6079 \pm 0.2431}$

Fig. 4: Posterior distribution for BAR when comparing EBCS-SVM with the reference methods on high IR datasetss. The region of the bottom-left represents EBCS-SVM, the region at the top is for rope, and the region in the bottom-right represents the reference method.

techniques. Experimental results revealed a leading performance of EBCS-SVM. The traditional SVM was unable to deal effectively with imbalanced datasets. On the other hand, data-level methods showed excellent performance, although in most cases required larger training times than algorithm-level methods. SVMSMOTE and CNN were the worst among data-level methods. The low performance of both can be because the implicit mapping of the kernel function is not taken into account to select boundary instances when sampling. EBCS-SVM considered the particularities of SVM to learn a model.

The most competitive method was RUS, a data-level technique that randomly subsamples the majority class. Since it does not require further information, RUS is fast. However, this method was outperformed by EBCS-SVM as the imbalanced ratio increased and EBCS-SVM required lower training time. Moreover, EBCS-SVM employed a self-adapted EA to adjust the evolutionary parameters during the learning. Therefore, EBCS-SVM is accurate and does not require fine-tuning of the SVM hyperparameters.

REFERENCES

[1] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[2] A. Fernández, S. García, M. Galar, R. C. Prati, B. Krawczyk, and F. Herrera, *Learning from Imbalanced Data Sets.* Springer International Publishing, 2018.

[3] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE T. Knowl. Data En.*, vol. 21, no. 9, pp. 1263–1284, 2009.

[4] B. Krawczyk, "Learning from imbalanced data: open challenges and future directions," *Prog. Artif. Int.*, vol. 5, no. 4, pp. 221–232, 2016.

[5] Z. Zhu, Z. Wang, D. Li, Y. Zhu, and W. Du, "Geometric structural ensemble learning for imbalanced problems," *IEEE T. Cybernetics*, vol. 50, no. 4, pp. 1617–1629, 2020.

[6] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Int. Res.*, vol. 16, no. 1, pp. 321–357, 2002.

[7] H. M. Nguyen, E. W. Cooper, and K. Kamei, "Borderline oversampling for imbalanced data classification," *Int. J. Knowl. Eng. Soft Data Paradigm.*, vol. 3, no. 1, pp. 4–21, 2011.
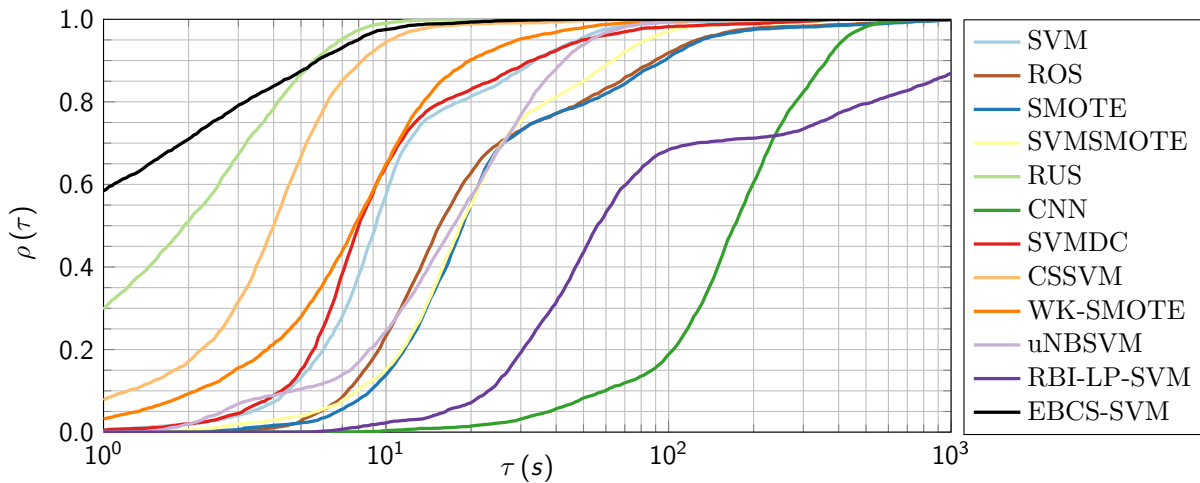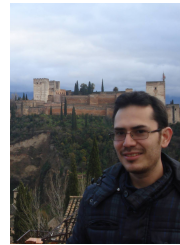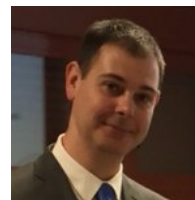
Fig. 5: Performance profile of the training time for each method.

[8] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *IEEE IJCNN*, 2008, pp. 1322–1328.

[9] P. Hart, "The condensed nearest neighbor rule," *IEEE T. Inform. Theory*, vol. 14, no. 3, pp. 515–516, 1968.

[10] Q. Kang, X. Chen, S. Li, and M. Zhou, "A noise-filtered under-sampling scheme for imbalanced classification," *IEEE T. Cybernetics*, vol. 47, no. 12, pp. 4263–4274, 2017.

[11] S. Xia, Y. Zheng, G. Wang, P. He, H. Li, and Z. Chen, "Random space division sampling for label-noisy classification or imbalanced classification," *IEEE T. Cybernetics*, pp. 1–14, 2021.

[12] J. Nalepa and M. Kawulok, "Selecting training sets for support vector machines: a review," *Artif. Intell. Rev.*, vol. 52, no. 2, pp. 857–900, 2019.

[13] T. Imam, K. M. Ting, and J. Kamruzzaman, "z-SVM: An SVM for improved classification of imbalanced data," in *Lect. Notes Artif. Int.*, A. Sattar and B.-h. Kang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 264–273.

[14] S. Datta and S. Das, "Near-bayesian support vector machines for imbalanced data classification with equal or unequal misclassification costs," *Neural Netw.*, vol. 70, pp. 39 – 52, 2015.

[15] J. Mathew, C. K. Pang, M. Luo, and W. H. Leong, "Classification of imbalanced data by oversampling in kernel space of support vector machines," *IEEE T. Neural Net. Lear.*, vol. 29, no. 9, pp. 4065–4076, 2018.

[16] S. Datta and S. Das, "Multiobjective support vector machines: Handling class imbalance with pareto optimality," *IEEE T. Neural Net. Lear.*, vol. 30, no. 5, pp. 1602–1608, 2019.

[17] A. Sinha, P. Malo, and K. Deb, *Evolutionary Bilevel Optimization: An Introduction and Recent Advances.* Springer International Publishing, 2017, pp. 71–103.

[18] A. Sinha, P. Malo, and K. Deb, "A review on bilevel optimization: From classical to evolutionary approaches and applications," *IEEE T. Evol. Comput.*, vol. 22, no. 2, pp. 276–295, 2018.

[19] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, "Choosing multiple parameters for support vector machines," *Mach. Learn.*, vol. 46, no. 1, pp. 131–159, 2002.

[20] M. L. Dantas Dias and A. R. R. Neto, "Evolutionary support vector machines: A dual approach," in *IEEE CEC*, 2016, pp. 2185–2192.

[21] S. Li and M. Tan, "Tuning SVM parameters by using a hybrid CLPSO–BFGS algorithm," *Neurocomputing*, vol. 73, no. 10, pp. 2089 – 2096, 2010, subspace Learning / Selected papers from the European Symposium on Time Series Prediction.

[22] A. D. Martinez, J. Del Ser, E. Villar-Rodriguez, E. Osaba, J. Poyatos, S. Tabik, D. Molina, and F. Herrera, "Lights and shadows in evolutionary deep learning: Taxonomy, critical methodological analysis, cases of study, learned lessons, recommendations and challenges," *Inform. Fusion*, vol. 67, pp. 161–194, 2021.

[23] I. Mierswa, "Evolutionary learning with kernels: A generic solution for large margin problems," in *Proceedings of GECCO.* ACM, 2006, pp. 1553–1560.

[24] A. Rosales-Pérez, S. Garcia, H. Terashima-Marin, C. A. Coello Coello, and F. Herrera, "MC^{2}ESVM: Multiclass classification based on cooperative evolution of support vector machines," *IEEE Comput. Intell. M.*, vol. 13, no. 2, pp. 18–29, 2018.

[25] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," *CoRR*, vol. abs/1802.01548, 2018.

[26] E. Real, C. Liang, D. R. So, and Q. V. Le, "AutoML-Zero: evolving machine learning algorithms from scratch," *CoRR*, vol. abs/2003.03384, 2020.

[27] A. Rosales-Pérez, J. A. Gonzalez, C. A. Coello Coello, H. J. Escalante, and C. A. Reyes-Garcia, "Surrogate-assisted multiobjective model selection for support vector machines," *Neurocomputing*, vol. 150, pp. 163 – 172, 2015.

[28] A. Rosales-Pérez, S. García, J. A. Gonzalez, C. A. Coello Coello, and F. Herrera, "An evolutionary multiobjective model and instance selection for support vector machines with pareto-based ensembles," *IEEE T. Evol. Comput.*, vol. 21, no. 6, pp. 863–877, 2017.

[29] K. O. Stanley, J. Clune, J. Lehman, and R. Miikkulainen, "Designing neural networks through neuroevolution," *Nat. Mach. Intell.*, vol. 1, no. 1, pp. 24–35, Jan 2019.

[30] H. L. Le, D. Landa-Silva, M. Galar, S. Garcia, and I. Triguero, "EUSC: a clustering-based surrogate model to accelerate evolutionary undersampling in imbalanced classification," *Appl. Soft Comput.*, vol. 101, p. 107033, 2021.

[31] J. Li, Q. Zhu, Q. Wu, Z. Zhang, Y. Gong, Z. He, and F. Zhu, "SMOTE-NaN-DE: addressing the noisy and borderline examples problem in imbalanced classification by natural neighbors and differential evolution," *Knowl.-Based Syst.*, vol. 223, p. 107056, 2021.

[32] C. Zhang, K. C. Tan, H. Li, and G. S. Hong, "A cost-sensitive deep belief network for imbalanced classification," *IEEE T. Neur. Net. Learn.*, vol. 30, no. 1, pp. 109–122, 2019.

[33] N. Wang, R. Liang, X. Zhao, and Y. Gao, "Cost-sensitive hypergraph learning with f-measure optimization," *IEEE T. Cybernetics*, pp. 1–12, 2021.

[34] A. Fernández, S. García, F. Herrera, and N. V. Chawla, "SMOTE for learning from imbalanced data: Progress and challenges, marking the 15-year anniversary," *J. Artif. Int. Res.*, vol. 61, no. 1, pp. 863–905, 2018.

[35] A. Estabrooks, T. Jo, and N. Japkowicz, "A multiple resampling method for learning from imbalanced data sets," *Comput. Intell.*, vol. 20, no. 1, pp. 18–36, 2004.

[36] V. López, A. Fernández, J. G. Moreno-Torres, and F. Herrera, "Analysis of preprocessing vs. cost-sensitive learning for imbalanced classification. open problems on intrinsic data characteristics," *Expert Syst. Appl.*, vol. 39, no. 7, pp. 6585 – 6608, 2012.

[37] F. R. Bach, D. Heckerman, and E. Horvitz, "Considering cost asymmetry in learning classifiers," *J. Mach. Learn. Res.*, vol. 7, pp. 1713–1741, 2006.

[38] B. B. Hazarika and D. Gupta, "Density-weighted support vector machines for binary class imbalance learning," *Neural Comput. Appl.*, vol. 33, no. 9, pp. 4243–4261, May 2021.

[39] M. Feurer and F. Hutter, *Hyperparameter Optimization*, F. Hutter, L. Kotthoff, and J. Vanschoren, Eds. Springer International Publishing, 2019.

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

[41] F. Friedrichs and C. Igel, "Evolutionary tuning of multiple SVM parameters," *Neurocomputing*, vol. 64, pp. 107 – 117, 2005, trends in Neurocomputing: 12th European Symposium on Artificial Neural Networks 2004.

[42] A. Rosales-Pérez, J. A. Gonzalez, C. A. Coello Coello, H. J. Escalante, and C. A. Reyes-Garcia, "Multi-objective model type selection," *Neurocomputing*, vol. 146, pp. 83 – 94, 2014.

[43] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," in *Adv. Neu. In.*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 2962–2970.

[44] 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 ACM SIGKDD*. ACM, 2013, pp. 847–855.

[45] A. M. Al-Zoubi, M. A. Hassonah, A. A. Heidari, H. Faris, M. Mafarja, and I. Aljarah, "Evolutionary competitive swarm exploring optimal support vector machines and feature weighting," *Soft Comput.*, vol. 25, no. 4, pp. 3335–3352, Feb 2021.

[46] G. Acampora, F. Herrera, G. Tortora, and A. Vitiello, "A multi-objective evolutionary approach to training set selection for support vector machine," *Knowl.-Based Syst.*, vol. 147, pp. 94–108, 2018.

[47] F. Cheng, J. Chen, J. Qiu, and L. Zhang, "A subregion division based multi-objective evolutionary algorithm for svm training set selection," *Neurocomputing*, vol. 394, pp. 70–83, 2020.

[48] W. Dudzik, J. Nalepa, and M. Kawulok, "Evolving data-adaptive support vector machines for binary classification," *Knowl.-Based Syst.*, vol. 227, p. 107221, 2021.

[49] A. Sinha and V. Shaikh, "Solving bilevel optimization problems using kriging approximations," *IEEE T. Cybernetics*, pp. 1–16, 2021.

[50] Y. Dhebar and K. Deb, "Interpretable rule discovery through bilevel optimization of split-rules of nonlinear decision trees for classification problems," *IEEE T. Cybernetics*, pp. 1–12, 2020.

[51] A. J. Bagnall and G. C. Cawley, "On the use of default parameter settings in the empirical evaluation of classification algorithms," *CoRR*, vol. abs/1703.06777, 2017.

[52] H. Q. Minh, "Some properties of gaussian reproducing kernel hilbert spaces and their implications for function approximation and learning theory," *Constr. Approx.*, vol. 32, no. 2, pp. 307–338, Oct 2010.

[53] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," in *IEEE CEC*, 2013, pp. 71–78.

[54] K. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization (Natural Computing Series)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005.

[55] R. Storn and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, 1997.

[56] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, and S. García, "Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework," *J. Mult.-Valued Log. S.*, vol. 17, no. 2-3, pp. 255–287, 2011.

[57] A. Luque, A. Carrasco, A. Martín, and A. de las Heras, "The impact of class imbalance in classification performance metrics based on the binary confusion matrix," *Pattern Recogn.*, vol. 91, pp. 216 – 231, 2019.

[58] A. Benavoli, G. Corani, J. Demšar, and M. Zaffalon, "Time for a change: A tutorial for comparing multiple classifiers through bayesian analysis," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 2653–2688, 2017.

[59] G. Corani, A. Benavoli, J. Demšar, F. Mangili, and M. Zaffalon, "Statistical comparison of classifiers through bayesian hierarchical modelling," *Mach. Learn.*, vol. 106, no. 11, pp. 1817–1837, 2017.

[60] A. Iranmehr, H. Masnadi-Shirazi, and N. Vasconcelos, "Cost-sensitive support vector machines," *Neurocomputing*, vol. 343, pp. 50–64, 2019.

[61] J. Wainer and G. Cawley, "Empirical evaluation of resampling procedures for optimising SVM hyperparameters," *J. Mach. Learn. Res.*, vol. 18, no. 15, pp. 1–35, 2017.

[62] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," *Math. Program.*, vol. 91, no. 2, pp. 201–213, Jan 2002.

Alejandro Rosales-Pérez received the B.S. degree in electronic engineering from the Instituto Tecnológico de Tuxtla Gutiérrez, Chiapas, Mexico in 2008, and the M.Sc. and Ph.D. degrees in computer science from INAOE, located at Puebla, Mexico in 2011 and 2016, respectively. He is currently with CIMAT Monterrey. His doctoral thesis was awarded by the National Association of Education Institutions in Information Technology (ANIEI) in 2016 and also by the Mexican Society on Artificial Intelligence (SMIA) in 2016. He is member of the Mexican National System of Researchers (SNI). His current research interests mainly include evolutionary computation and machine learning.

Salvador García received the B.S. and Ph.D. degrees in Computer Science from the University of Granada, Granada, Spain, in 2004 and 2008, respectively. He is currently a Full Professor in the Department of Computer Science and Artificial Intelligence, University of Granada, Granada, Spain. Dr. García has published more than 100 papers in international journals (more than 70 in Q1), h-index 54. He is an associate editor in chief of "Information Fusion" (Elsevier), and an associate editor of "Swarm and Evolutionary Computation" (Elsevier) and "AI Communications" (IOS Press) journals. His research interests include data science, data preprocessing, Big Data, evolutionary learning, Deep Learning and metaheuristics. He belongs to the list of the Highly Cited Researchers in the area of Computer Sciences (2014-2020): http://highlycited.com/ (Clarivate Analytics).

Francisco Herrera (SM'15) received his M.Sc. in Mathematics in 1988 and Ph.D. in Mathematics in 1991, both from the University of Granada, Spain. He is currently a Professor in the Department of Computer Science and Artificial Intelligence at the University of Granada and Director of DaSCI Institute (Andalusian Research Institute in Data Science and Computational Intelligence). His current research interests include among others, Computational Intelligence (including fuzzy modeling, computing with words, evolutionary algorithms and deep learning), information fusion and decision making, and data science (including data preprocessing, prediction and big data). He has been the supervisor of more than 50 Ph.D. students. He has published more than 500 journal papers, receiving more than 98,000 citations (Google Scholar, H-index 152). He is co-author of several books, and the Editor in Chief of Information Fusion (Elsevier). He is been selected as a Highly Cited Researcher (in the fields of Computer Science and Engineering, 2014 to present, Clarivate Analytics)