This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2019.2895298, IEEE Transactions on Evolutionary Computation

Optimal-margin evolutionary classifier

Mohammad Reza Bonyadi, *Senior Member, IEEE*, David C. Reutens

***Abstract*—We introduce a novel approach for discriminative classification using evolutionary algorithms. We first propose an algorithm to optimize the total loss value using a modified 0-1 loss function in a one-dimensional space for classification. We then extend this algorithm for multi-dimensional classification using an evolutionary algorithm. The proposed evolutionary algorithm aims to find a hyperplane which best classifies instances while minimizes the classification risk. We test particle swarm optimization, evolutionary strategy, and covariance matrix adaptation evolutionary strategy for optimization purpose. After parameter selection, we compare our results with well-established and state-of-the-art classification algorithms, for both binary and multi-class classification, on 23 benchmark classification problems, with and without noise and outliers. We also compare these methods on a seizure detection task for 12 epileptic patients. Results show that the performance of the proposed algorithm is significantly (Wilcoxon test) better than all other methods in almost all problems tested. We also show that the proposed algorithm is significantly more robust against noise and outlaiers comparing to other methods. The running time of the algorithm is within a reasonable range for the solution of real-world classification problems.**

***Index Terms*—Evolutionary algorithms, Supervised learning, Discriminative classification.**

I. INTRODUCTION

THE main goal of a supervised classification algorithm is to identify the class to which each instance belongs based on a given set of correctly labeled instances. There are two types of classifiers, generative and discriminative [1]. While generative classifiers learn a joint distribution between inputs and class labels, discriminative classifiers learn the exact class label from the training dataset. It has been shown [1], [2] that "discriminative classifiers are almost always to be preferred".

A discriminative classifier [1] is defined as follows:

Definition 1. *(Discriminative classifier) Let $Y = \{0, ..., c-1\}$, a set of class labels, and $S, S_0, S_1, ..., S_{c-1}$ be sets of instances in each class such that, $\forall i, j \in Y$, $i \neq j$, $|S_i| = m_i$, $|S| = m$, $S_i \cap S_j = \emptyset$, and $\cup_{i=0}^{c-1} S_i = S$. A classifier $\psi_\beta : S \to Y$, aims to guarantee*

$$\forall i \in Y, \forall x \in S_i, P(\psi_\beta(x) = i|x) = 1,$$

where β is a set of configurations for the procedure $\psi_\beta(x)$, and P is the probability measure.[1]

The classifier $\psi_\beta(\cdot)$ is usually a combination of an optimization problem, Ω_β, a transformation $\mathcal{M}_\beta : S \to \hat{S}$, and a discriminator $\mathcal{D} : \hat{S} \to Y$. The solution to Ω_β yields β that transforms any given instance x to $\hat{x}$ through $\hat{x} = \mathcal{M}_\beta(x)$, which is finally mapped into the class label by $\mathcal{D}(\hat{x})$. In reality, the true class of only a subset of S is known (the training set). It is hence challenging to find a transformation β for which $\psi_\beta(\vec{x})$ is the true class of all $\vec{x}$ in S, including the ones that are not in the training set (unseen instances). Therefore, the generality of an optimized β depends on the assumptions made to formulate Ω_β and the given training set itself. We assume that $\mathcal{M}_\beta(\vec{x})$ is linear ($\beta =< \omega, b >$ and $\mathcal{M}_\beta(\vec{x}) = \vec{x}\omega^T + b$). For binary classification ($y_i \in \{-1, 1\}$ for all i), $\omega : \mathbb{R}^n \to \mathbb{R}$ and $b \in \mathbb{R}$ represent the normal vector and the intercept of a hyperplane that separates the two classes, the *separator hyperplane*. Note that non-linear classification is possible by adding a kernel to a linear classifier [3]. Also, the algorithms for binary classification can be extended to work with multi-class classification through the one-vs-one approach [4].

Motivation: One way to formulate the optimization problem Ω_β for discriminative classification is through minimization of the number of misclassified instances, i.e., minimization of the total loss of the 0-1 loss function [5] (see Section II-A). Optimization of the total loss over the 0-1 loss function leads to finding the optimal separator hyperplane that is robust against outliers [5]. As this optimization problem is NP-hard [6], existing algorithms to find such a hyperplane are impractical for real-world classification [5]. In addition, the original definition of the 0-1 loss function leads to inefficiencies such as sensitivity to class imbalance and ignorance of classification risk. Several alternative formulations have been proposed [2], [7] that turn this formulation into a smooth convex one and use gradient descent to solve it efficiently. These alternative formulations, however, are sensitive to noise and outliers.

In this paper, we propose an alternative loss function to address the shortcomings associated with the 0-1 loss

[1]Throughout this paper, we consider a special case of classification problems where all members of S are in $\mathbb{R}^n$ (so called feature space), and each instance in S is represented by a vector. We also assume that feasible values for x_j (called a variable throughout this paper), the j^{th} element of the instance $\vec{x}$, are ordered by the operator "$\leq$" (i.e. x_j is not categorical). Note, however, that these categorical variables can be encoded into continuous space variables, ordered by "$\leq$", by an encoding method such as the one-hot-encoding.

All authors are with the Centre for Advanced Imaging (CAI), the University of Queensland, Brisbane, QLD 4072, Australia. M. R. Bonyadi (reza@cai.uq.edu.au, rezabny@gmail.com) is also with the PACE Analytics, RioTinto, Brisbane, QLD 4000, Australia, and with the Optimisation and Logistics Group, The University of Adelaide, Adelaide, SA 5005, Australia.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2019.2895298, IEEE Transactions on Evolutionary Computation

function (sensitivity to imbalances in the number of instances in each class and ignorance of classification risk) while keep its advantages (robustness against noise and outliers). We then propose an efficient algorithm to find the optimal solution to the one-dimensional discriminative classification problem using this loss function. We extend that algorithm to handle multi-dimensional classification problems using an evolutionary algorithm. Three evolutionary algorithms (particle swarm optimization, evolutionary strategy, and covariance matrix adaptation evolutionary strategy) are tested and compared. We finally extend the algorithm to work with multi-class classification problems.

We structure the paper as follows: Section II provides a background on the classification methods used for comparison. Section III details our proposed method. Section IV reports and discusses the results of the comparisons between our method and 6 other classification methods based on 23 standard benchmark classification problems and 12 seizure detection datasets. Section V concludes the paper and points to potential future directions.

II. Background

This section provides background information on loss functions, optimization of the 0-1 loss function, and state-of-the-art classification methods.

A. Loss function and total loss

One of the most frequently-used definitions for the optimization problem Ω_β is based on "total loss" and "loss function":

$$\Omega_\beta : \min_\beta \sum_{i=1}^{m} h(y_i, x_i, \beta) + \alpha \mathcal{R}(\omega) \quad (1)$$

where $\beta =< \omega, b >$ and $h(.)$ is a *loss function* that measures the difference between the class label y_i and the class label calculated by the classification algorithm ($\mathcal{D}(\mathcal{M}_\beta(\vec{x}_i))$), and $\sum_{i=1}^{m} h(y_i, x_i, \beta)$ is the *total loss*. The function $\mathcal{R} : \mathbb{R}^n \to \mathbb{R}$ is the *regularization function* and α is the *regularization factor*. The aim is then to minimize the total loss, given the regularization factor α. The role of regularization is to control the balance between the bias and the variance of the model [8]. Frequently-used regularization functions with this settings are L_1 ($\mathcal{R}(\omega) = ||\omega||_1$, also known as least absolute shrinkage and selection operator, LASSO, [9]) and L_2 ($\mathcal{R}(\omega) = ||\omega||_2$, also known as Tikhonov regularization [8]). Given $\beta =< \omega, b >$, the simplest loss function is the *0-1 loss function* that generates a '1' for a misclassified instance and a '0' for a correctly classified instance.

$$h(y, x, \beta) = \begin{cases} 1 & y(x\omega^T + b) \leq 0 \\ 0 & y(x\omega^T + b) > 0 \end{cases} \quad (2)$$

If y_i and $\vec{x}_i\omega^T + b$ have the same sign then the instance $\vec{x}_i$ has been classified correctly using ω and b. The optimal ω and b lead to the minimum total loss, i.e., minimum number of misclassified instances in X (number of instances $\vec{x}_i$ for which $y_i(\vec{x}_i\omega^T + b) < 0$). The best solution to this optimization problem provides the normal vector (ω) and the intercept (b) of the optimal separator hyperplane. This problem is, however, NP-hard [6] mainly because of the definition of $h(.)$. Hence, many studies have proposed other differentiable smooth variations of this loss function so that the final optimization problem is solvable using a gradient descent algorithm. For example, the hinge loss, defined by $h(y_i, \vec{x}_i, < \omega, b >) = max(0, 1 - y_i(\vec{x}_i\omega^T + b))$, has been used in support vector machines [7] and the log loss, defined by $h(y_i, \vec{x}_i, < \omega, b >) = \log(1 + e^{-y_i(\vec{x}_i\omega^T + b)})$ has been used in the logistic regression [2]. An alternative to the log loss is the sigmoid loss, $\frac{1}{1+e^{-ky(\vec{x}\omega^T+b)}}$, where k is a constant (Fig. 1 shows some well-known loss functions).

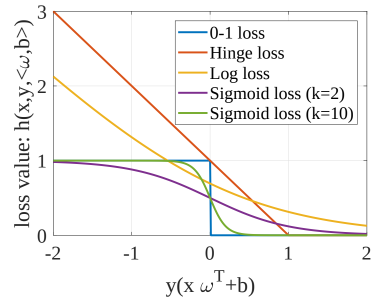

Fig. 1. Loss functions frequently used for classification.

Using the 0-1 loss function to calculate the total loss leads to identification of a hyperplane that, unlike its smooth alternatives, is known to be robust to outliers [5]. The reason is that the loss value does not scale with the misclassified instances in the 0-1 loss function while it does scale with them in hinge and log loss, i.e., the smaller the value of $y_i(\vec{x}_i\omega^T + b)$, the larger the impact on the total loss.

B. Direct optimization of 0-1 loss function

Direct optimization of the total loss with the 0-1 loss function, although leads to an optimal separator hyperplane, is an NP-hard problem. In [5], the authors proposed multiple algorithms to solve this problem to optimality and near optimality. For example, they used the branch and bound method, in combination with linear programming, to solve the problem to optimality. The time complexity of the method, however, was exponential, making it impractical for real-world classification. Another method proposed was based on the fact that an optimal hyperplane can be built upon n instances (n is the number of dimensions) from the training set. Hence, they proposed that all possible combinations of n instances be tested to find the combination upon which the best separator hyperplane can be built. The time complexity of this method is also exponential with the number of instances and dimensions. The fastest method proposed was based on optimization of the sigmoid

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2019.2895298, IEEE Transactions on Evolutionary Computation

loss with variable k. As larger values for the k leads to a better approximation of the 0-1 loss function, the sigmoid loss with larger k would lead to convergence to the optimal hyperplane. However, larger values for k cause the search space to become non-convex, making it difficult to solve the problem to optimality. Hence, it was proposed to increase k from 2 to 200 and the total loss function optimized locally for each k, given the solution found for the previous value of k. The method, called Smooth Logistic Algorithm (SLA), was shown to effectively solve binary classification problems to near optimality with a much better time complexity compared to the branch and bound and point selection methods. Although SLA outperformed support vector machines and logistic regression, its running time was still impractical even for a small number of dimensions.

C. Classification approaches with smoothed 0-1 loss function

We describe in brief popular classification methods used herein for benchmarking and comparison.

1) Support vector machine (SVM) and its recent extensions: SVM aims to find the normal vector (ω) and the intercept (b) of the separator hyperplane such that the distance between the closest instances from each class (i.e., support vectors) to the hyperplane is maximized (margin) [10] while the instances are classified correctly. The balance between the separation accuracy and the margin is adjusted by a variable λ. SVM can be formulated by a hinge loss plus the margin term and optimized by a gradient descent. The separation is determined by the sign of $\vec{x}\omega^T + b$ (T is transpose) indicating the side of the hyperplane to which the instance, $\vec{x}$, belongs. Note that this approach prioritizes hyperplanes that provide minimum empirical risk (maximum margin) measured by the inverse of the distance between the hyperplane and the closest instances from each class. This risk is of utmost importance especially in classification of noisy datasets [10].

Twin SVM (TSV) [11], [12], a recent variant of SVM, seeks a pair of hyperplanes, one of which is closer to the instances from class -1 than instances from class 1 and the other is closer to the instances from class 1 compared to the instances from class -1. The discriminator is then used to calculate which hyperplane is closer to an instance. A recent extension [13] of TSV is based on the introduction of a weighted linear loss to the formulation of TSV (and hence is called WSV throughout the paper) instead of the hinge loss, reducing the quadratic problem to a linear one. There are also online version of SVM, recently developed [14].

2) Minimax probability machine (MPM) and its recent extensions: Minimax probability machine (MPM) is a type of discriminative classifier that aims to minimize the maximum misclassification probability of instances [15]. MPM attempts to find a generalizable margin by paying attention to the distributions within classes rather than the instances themselves. Evidence [16] suggests that the structure of the instances in different classes provides important information for the design of generalizable transformations for classification. Structural minimax probability machine (SMP) [17] makes use of the structural information, approximated by two finite mixture models in each class, in the context of MPM for classification of instances. This idea has been shown to be very effective on a set of standard datasets.

3) Logistic regression (LR) and its recent extensions: Logistic regression (LR) [18] provides a smooth estimation of the 0-1 loss function using the sigmoid function (see Fig. 1), usually with $k = 2$. Elastic-net extends LR by incorporating the LASSO and Tikhonov regularization terms. For classification purposes, elastic net usually performs optimization in relation to the true class labels, restricting the algorithm to binary targets. This restriction was resolved in [19] (Discriminative Elastic Least-square, DEL) where a term was used to relax the class labels. The optimization problem associated with this classification process was then introduced and solved via an iterative procedure.

4) Linear discriminant analysis (LDA) and its recent extensions: A geometrical interpretation for ω (rather than being the norm of a separator hyperplane) is a transformation from an n-dimensional space to a one-dimensional space. With this interpretation, the intercept b is just a threshold that separates the transformed instances. LDA uses this interpretation and aims to find $\omega : \mathbb{R}^n \to \mathbb{R}$ such that, in the transformed space, the distance between the centers of the classes is maximized while the spread of instances within the class is minimized. The early version of LDA [20] was proposed mainly for classification. In that version, it was assumed that the conditional probabilities $P(\vec{x}^{(i)}|y^{(i)} = -1)$ and $P(\vec{x}^{(i)}|y^{(i)} = 1)$ ($y^{(i)}$ is the label of $\vec{x}^{(i)}$) are both normally distributed with mean and covariance parameters $(\vec{\mu}_1, \Sigma_1)$ and $(\vec{\mu}_2, \Sigma_2)$. Given these assumptions, Fisher [20] proved that $\omega = (\Sigma_1 + \Sigma_2)^{-1}(\vec{\mu}_2 - \vec{\mu}_1)$ and $k = \frac{1}{2}\vec{\mu}_2^T\Sigma_2^{-1}\vec{\mu}_2 - \frac{1}{2}\vec{\mu}_1^T\Sigma_1^{-1}\vec{\mu}_1$ leads to maximizing of $\frac{W^T S_B W}{W^T S_W W}$, where $S_W = (\Sigma_1 + \Sigma_2)$ and $S_B = (\vec{\mu}_2 - \vec{\mu}_1)(\vec{\mu}_2 - \vec{\mu}_1)^T$. In this case, ω is considered to be the norm of a hyperplane that discriminates the two classes and k shifts the hyperplane to be between the two classes, i.e., $\vec{x}\omega^T > k$ if the instance $\vec{x}$ belongs to class 1 (i.e., $\Omega_{\omega,k}$ seeks to maximize $\frac{W^T S_B W}{W^T S_W W}$, and $\mathcal{D}(\vec{x}) = sign(\mathcal{M}_{\omega,k}(\vec{x})) = \vec{x}\omega^T - k$).

MBA [21] is a recent extension of LDA that aims to find a set of transformations so that the between-class scatter is maximized between each pair of the classes. The algorithm was shown to be effective for multiclass classification. For binary classification however, the algorithm is exactly the same as LDA.

D. Evolutionary algorithms for classification

Evolutionary algorithms, EAs, (e.g., particle swarm optimization [22] and evolutionary strategy [23]) work based on a population of candidate solutions that

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2019.2895298, IEEE Transactions on Evolutionary Computation

are evolved according to some rules until they converge to an optimum solution. Each evolutionary algorithm has specific properties that confer advantages/disadvantages for specific applications. These methods aim to use information coded in each individual in the population (with the size λ) and update them to find better solutions. For example, evolutionary strategy (ES) generates new individuals using a normal distribution with the mean of the current location of the individual and an adaptive variance, calculated based on the distribution of "good" solutions. Covariance matrix adaptation evolutionary strategy (CMAES) employs a similar idea but updates the covariance matrix of the normal distribution (rather than the variance alone) to generate new instances, accelerating convergence to local optima. This idea takes into account non-separability of dimensions during optimization and hence is more successful when variables are interdependent. Particle swarm optimization (PSO) follows a different approach where individuals (particles) combine their experiences with others and calculate movement directions in the search space. See [22], [23] for detail of these methods.

Evolutionary algorithms have been used for classification in the past. For example, different evolutionary methods have been used for neural network training or generative classification [24], [25]. In these methods, either the architecture [26] or the weights [27] of a neural network is optimized using an evolutionary algorithm. Most of these methods work with the multi-layer perceptrons that is a generative classifier.

Hyperparameters of classification methods play an important role in their final accuracy and generalization ability. The optimal values for these parameters, however, could be different for different problems, hence, an automatic setting algorithm could be used to ensure the optimal performance of the method on the given problem. Evolutionary algorithms have been also used to set the hyperparameters of classification methods. In [28] for example, authors used an evolutionary algorithm to set the parameters of SVM (called EVM throughout the paper) through a cross-validation search procedure. This procedure is, however, very time consuming as the method is run against different hyperparameter values and cross-validated to ensure high accuracy. Bayesian Optimization [29] is another method that has been used to set SVM (called BVM throughout the paper) or LDA (called BDA throughout the paper) parameters.

Feature selection is one of the richest areas for which Evolutionary Algorithms have been used [30]–[32]. Feature selection requires binary selection of a set of features that cannot be optimized by classical optimization methods such as gradient descent. The selected feature using this method are then used with a classifier.

III. Proposed evolutionary-based classifier

In this section we describe Ω_β and $\mathcal{D}$ for our proposed classifier (note that $\mathcal{M}_\beta$ is linear), Optimal-margin Evolutionary Classifier (OEC). We first modify the definition of the 0-1 loss function in Eq. 2 to overcome its sensitivity to imbalances in the number of instances in different classes. We then propose an algorithm that, in a one-dimensional space, finds a hyperplane minimizing the total loss using the modified 0-1 loss function while maximizing the margin between the hyperplane and the instances from each class (minimum empirical risk). Finally, we propose an evolutionary-based algorithm to extend the use of this algorithm to multi-dimensional classification.

A. Shortcomings of the 0-1 loss function

The total loss calculation using the 0-1 loss function (Eq. 2) is sensitive to imbalance in the number of instances, a significant issue in real-world classification problems [33]. The reason is that the number of misses (function $h(.)$ in Eq. 2) are counted without taking into account the total number of instances in each class. Consider, for example, that there are two classes, the first class contains 200 instances and the second class contains only 10 instances. Also assume that hyperplane $\mathcal{H}$ misclassifies all instances in the second class and classifies all instances in the first class correctly while hyperplane $\mathcal{H}'$ misclassifies more than 10 instances from the first class and classifies all instances from the second class correctly. Using the 0-1 loss function and the total loss, hyperplane $\mathcal{H}'$ is evaluated to be worse than $\mathcal{H}$ because the number of misclassified instances is larger for $\mathcal{H}'$, ignoring the fact that $\mathcal{H}$ cannot classify the second class at all. In addition, two hyperplanes that provide the same total loss using the 0-1 loss function are evaluated to be the same, ignoring any empirical risk considerations.

B. Modified one-dimensional 0-1 loss function

To fix the sensitivity to imbalance in the 0-1 loss function, we minimize the summation of the ratio of misclassified instances rather than the *number* of misclassified instances. As this ratio is in $[0,1]$ for all classes, it is independent of the number of instances, hence, this approach leads to removing the sensitivity to imbalance in the number of instances in the classes. We call this the 0-1 loss function with priors (LFP-0-1), defined by:

$$h(\vec{x}, y, <\omega, b>) = \begin{cases} \frac{1}{n_1} & y(x\omega^T + b) \le 0 \text{ and } y = 1 \\ \frac{1}{n_{-1}} & y(x\omega^T + b) \le 0 \text{ and } y = -1 \\ 0 & y(x\omega + b) > 0 \end{cases} \tag{3}$$

The total loss value is calculated by Eq. 1, given α and and the regularization function $\mathcal{R}(.)$.

Let us assume that the number of dimensions of the given instances (X) is one, $n = 1$. In this case, the optimal hyperplane that separates the classes (i.e., minimizes the total loss with LFP-0-1 loss function) is expressed by $s \in \{-1, 1\}$ and a threshold (intercept) $b = t$. We define the *optimal threshold*, optimal t, as the threshold that minimizes the total loss with LFP-0-1 loss function.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2019.2895298, IEEE Transactions on Evolutionary Computation

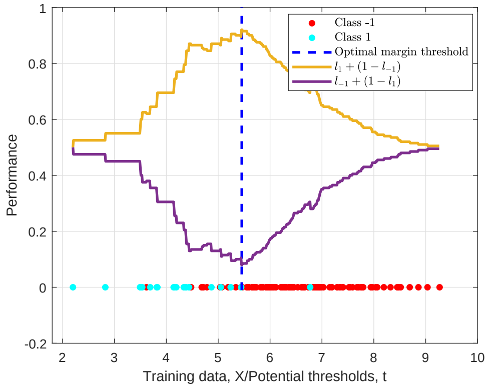

Fig. 2. Value of $l_{-1}(t)+(1-l_1(t))$ and $l_1(t)+(1-l_{-1}(t))$ (vertical axis) vs the threshold t (horizontal axis). The horizontal axis also shows the coordinate of the training data, X, in a one-dimensional space. Note that the value "0" for the vertical coordinate of the instances is arbitrary and it is for demonstration purposes only.

The value of s may negate the order of instances while the threshold t differentiates between them. We prove the following Remark, that will be used for our further discussions.

Remark 1. *Let X be an $m \times 1$ matrix of one dimensional instances, and Y is the same size representing the class labels of the instances. Also, for a given threshold, $t \in \mathbb{R}$, let the ratio of the instances in X from class -1 on the left hand side of t (i.e., the number of instances that are smaller than t) be $l_{-1}(t)$ and the ratio of the instances from class 1 on left hand side of t be $l_1(t)$. The optimal threshold has the minimum value of $l_{-1}(t)+(1-l_1(t))$ or $l_1(t)+(1-l_{-1}(t))$, over all possible possible t.*

Proof. There are two cases: the instances on the left hand side of t are classified as class -1 or 1. In the first case, the instances to the left hand side of t (inclusive) are classified to class -1, $1-l_{-1}(t)$ and $l_1(t)$ are the ratio of misclassification of the instances from class -1 and 1, respectively, at the threshold t. Hence, $(1-l_{-1}(t))+l_1(t)$ is the total ratio of misclassification of instances belong to the class -1 plus the ratio of the misclassification of instances in the class 1 at the threshold t. In the second case, the instances on the left hand side of t are classified to class 1, $l_{-1}(t)+(1-l_1(t))$ is the ratio of misclassification of instances in the class 1 plus the ratio of the misclassification of instances in the class 1 at the threshold t. Thus, if either $l_{-1}(t)+(1-l_1(t))$ or $(1-l_{-1}(t))+l_1(t)$ is minimum at a given t then that t is an optimal threshold. □

Figure 2 shows the value of $l_{-1}(t)+(1-l_1(t))$ and $l_1(t)+(1-l_{-1}(t))$ for different t.

The optimal threshold that optimizes the total loss with LFP-0-1 loss function might not be unique. This can be observed in Fig. 2, where $l_1(t)+(1-l_{-1}(t))$ indicates many plateaus for different values of t. The reason is that any value for t between two consecutive instances (recall that X is one-dimensional) would lead to a constant $l_1(t)$ and $l_{-1}(t)$, hence, the value of $l_{-1}(t)+(1-l_1(t))$ and $l_1(t)+(1-l_{-1}(t))$ are constant between each instance and the next (i.e., $l_{-1}(x_i)+(1-l_1(x_i)) = l_{-1}(t)+(1-l_1(t))$ for all $t \in [x_i, x_{i+1})$). Therefore:

1) To find an optimal t, we only need to test the value of $l_{-1}(t)+(1-l_1(t))$ and $l_1(t)+(1-l_{-1}(t))$ at the location of instances, i.e., for $t = x_i$ for all i, where x_i is the i^{th} instance in X.
2) The optimal threshold with minimum risk in the one-dimensional space is half-way between the instance that provides the minimum total loss ($l_{-1}(t)+(1-l_1(t))$ or $l_1(t)+(1-l_{-1}(t))$) and the next instance. The reason is that this threshold is the furthest from the closest instances from each class. This is called the *optimal-margin threshold*, t', throughout the paper, and is unique.

Algorithm 1 Optimum margin threshold **Input**: X: the training set, $m \times 1$; Y: the class labels, $m \times 1$ **Output**: t': the optimal margin threshold; s: a coefficient; p: the performance; r: size of margin

X=sort(X), and update Y accordingly
$l_{-1}(x_0) = 0$, $l_1(x_0) = 0$, $p = 0$, $t' = -\inf$
k_{-1} = number of instances in class -1
k_1 = number of instances in class 1
for i=1 to number of instances -1 **do**
  if x_i (the ith instance in X) is in class -1 **then**
    $l_{-1}(x_i) = l_{-1}(x_{i-1}) + \frac{1}{k_{-1}}$
  else
    $l_1(x_i) = l_1(x_{i-1}) + \frac{1}{k_1}$
  end if
  $a_{-1} = l_{-1}(x_i)+(1-l_1(x_i))$, $a_1 = l_1(x_i)+(1-l_{-1}(x_i))$
  if $a_{-1} > p$ **then**
    $t = \frac{x_i+x_{i+1}}{2}$, $s = 1$, $p = a_{-1}$
    $r = \frac{|x_i-x_{i+1}|}{2}$
  end if
  if $a_1 > p$ **then**
    $t = -\frac{x_i+x_{i+1}}{2}$, $s = -1$, $p = a_1$
    $r = \frac{|x_i-x_{i+1}|}{2}$
  end if
end for

We now propose Algorithm 1 to obtain the optimal-margin threshold, t, and $s \in \{-1, 1\}$ to distinguish between two classes in a one dimensional space. The main idea is to iterate over all instances ($x_i \in X$, where X is sorted), sorted in the one dimensional space, and calculate $l_{-1}(x_i)$ and $l_1(x_i)$. Ultimately, the goal is to find the "best instance" that, if used as a threshold (t), the total misclassification percentage is minimized, that is either $l_{-1}(t)+(1-l_1(t))$ or $l_1(t)+(1-l_{-1}(t))$ depending on the order of classes. To maximize the margin, r, we set t' to half way between the best instance and the next instance. The time complexity of this algorithm is in $O(m\log(m))$ (to sort the instances), where m is the number of instances.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2019.2895298, IEEE Transactions on Evolutionary Computation

Algorithm 2 Objective function **Input**: X: the $m \times n$ training data; Y: the class labels, $m \times 1$; β: the transformation; α: L_1 regularization factor **Output**: z: objective value

```
X̂ = Xω^T
[t', s, p, r] = Algorithm 1 (X̂, Y)
if p < 1 then
    z = p
else
    z = 1 + r
end if
z = z + α||β||_1
```

This algorithm is able to find t' and s in a one dimensional space that optimizes Eq. 2 with $h(.)$ defined by Eq. 3 [2]. The purpose of s is to make sure that the instances from class +1 are always at the right hand side of the threshold, t, and ultimately t'. This is arbitrary and does not have any impact on the final results, however, it standardizes the expectation of where the instances of different classes would place after the optimization. The value of p indicates the average performance in terms of the ratio of correctly classified instances. A new instance x is classified to class -1 if $sx \leq t'$, and to class $+1$ otherwise.

C. *Evolutionary classifier: binary classification*

By definition, we seek $\omega : \mathbb{R}^n \to \mathbb{R}$ so that $X\omega^T$ is separable by $\mathcal{D}(.)$. Geometrically, ω is a line that passes through the center of the coordinates system (intercept equal to zero). As $X\omega^T$ represents the projected instances on the line ω, we seek the best ω such that the projection of instances on it ($X\omega^T$, that is one-dimensional) maximizes the separability, measured by $\mathcal{D}(.)$. Because $X\omega^T$ is one-dimensional, we use Algorithm 1 as our $\mathcal{D}(.)$ to discriminate between classes. Hence, the aim is to find an ω for which the performance and margin calculated by Algorithm 1 using $X\omega^T$ and Y (the classes) is maximized. For this objective, finding the best ω is not a convex optimization problem, hence, an evolutionary algorithm is a good choice for optimization purposes. The objective value for evolutionary algorithm is calculated by Algorithm 2.

If the transformed instances are separable, then the performance p increases to 1. In this case, the final performance is set to $1+r$ to ensure that, among all possible ω that separate the instances, the ω which maximizes the margin is selected. The maximum margin provides a lower risk on classification of unseen instances.

In the cases where the instances are linearly separable, the margin r may increase only because $||\omega||_2$ increases. The reason is that, by increasing $||\omega||_2$, the distance between all instances increases, leading to an "illusion" of a larger margin, r. Hence, in all of our EA algorithms, we normalize the candidate solutions at each iteration to ensure that the solutions do not grow unbounded, that is equivalent to L_2 regularization. We call this algorithm the Optimal-margin Evolutionary Classifier (OEC). The L_1 regularization has been considered in our calculations by optimizing $z + \alpha||\omega||_1$ rather than z itself.

[2]The proof is trivial and can be achieved by contradiction.

Algorithm 3 Multiclass OEC **Input**: X: the $m \times n$ training data, Y: class labels **Output**: ω: the set of best transformations; t': the set of best thresholds; e: set of transformation coefficients; p: the average estimated performance

```
for i=0 to number of classes - 2 do
    for j=i+1 to number of classes - 1 do
        X_{i,j} = instances in X that belong to class i or j
        Y_{i,j} = all class labels of instances in classes i or
        j, translated to labels −1 and 1
        Use an evolutionary algorithm to find the best
        ω_{i,j}, a n × 1 transformation, with objective in
        Algorithm 2(X_{i,j}, Y_{i,j}, ω_{i,j})
        Calculate X̂ = Xω_{i,j}^T,
        [t'_{i,j}, s_{i,j}, p_{i,j}, r_{i,j}] = Algorithm 1(X̂, Y_{i,j}),
    end for
end for
```

After optimization of ω, an instance $\vec{x}$ is classified to class -1 if $s\vec{x}\omega^T < t'$, and to class 1 otherwise. Thus, we may assume that $s\vec{x}\omega^T - t'$ is the equation for the separator hyperplane found by our algorithm, where $s\omega$ is the normal vector of the hyper plane and $-t'$ is the intercept [3].

D. *Evolutionary classifier: multi-class classification*

We use the one-vs-one method to extend OEC to work with multi-class classification because it is more effective comparing to other methods in the multi-class classification task [4]. Here, we classify the instances in each pair of classes as a binary classification problem. For each class pairs i and j, OEC finds a transformation $\omega_{i,j}$, intercept $t'_{i,j}$, and coefficient $s_{i,j}$, that best separates those classes (see Algorithm 3). Algorithm 3 provides exactly $\frac{c(c-1)}{2}$ transformations, each optimally (using the objective Algorithm 2) calculated to separate one pair of classes.

To classify a new instance, we use a voting system in which the instance is transformed by all transformations (all $\omega_{i,j}$) and the class is calculated based on the votes received by each $t'_{i,j}$ and $s_{i,j}$ (see Algorithm 4).

E. *A visual overview through an example*

We generated a synthetic dataset for a binary classification problem for the purpose of demonstrating how OEC works. We used a multivariate normal distribution,[4] with covariance matrix of $\begin{bmatrix} 0.87 & -0.5 \\ 1.5 & 2.6 \end{bmatrix}$ and

[3]OEC source codes in Java, Matlab, and Python are available online at https://github.com/rezabonyadi/LinearOEC

[4]The covariance matrices and the means have been selected by some trial to illustrate the procedure of OEC as clear as possible and they do not have any other specific characteristics.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2019.2895298, IEEE Transactions on Evolutionary Computation

Algorithm 4 Get Class **Input**: x: the $1 \times n$ training data; ω: A $c \times c$ matrix of transformations, $n \times 1$ each; t': A $c \times c$ matrix of thresholds; s: A $c \times c$ matrix, each value in $\{-1, 1\}$ **Output**: the class label e

```
E =< 0, ..., 0 > with the length c.
for i=0 to number of classes - 2 do
    for j=i+1 to number of classes - 1 do
        if s_{i,j} x ω^T_{i,j} < t' then y = −1, else y = 1.
        if y = 1 then E_i = E_i + 1 else E_j = E_j + 1.
    end for
end for
e = argmax_i {E_i}.
```

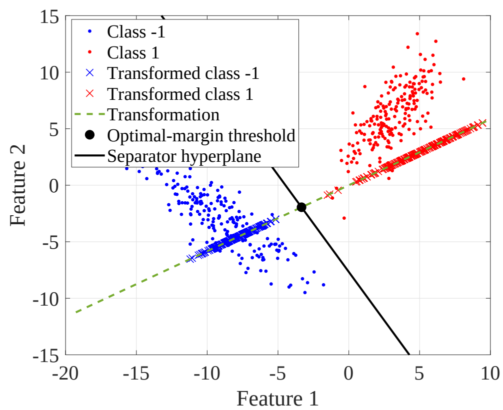

Fig. 3. After the transformation, the instances are mapped into a line, represented by ω. Alternatively, they are separable by the separator hyperplane.

$\mu =< 3, 6 >$ for class 1, and with covariance matrix $\begin{bmatrix} 2.83 & -2.83 \\ 0.71 & 0.71 \end{bmatrix}$ and $\mu =< -9, -3 >$ for class -1, and characteristics are shown in Figure 3.

We then found the optimal ω by an EA (CMAES in this example), as well as s and t' by Algorithm 1, described in Section III-C. Note, ω found by this method transforms instances from an n-dimensional space to a one-dimensional space where the instances are separable by a discriminator threshold, t'. An alternative view is that the instances are separable by a hyperplane, $s\vec{x}\omega^T - t'$. Figure 3 shows the transformation and the separator hyperplane.

The histogram of the transformed instances after optimizing ω is shown in Fig. 4. The optimal margin threshold has been also shown in the figure. It is clear that the proposed OEC is able to find a high-quality separator hyperplane to distinguish between classes.

IV. Experiments and Results

We compare OEC with six other classification methods over 35 classification problems in this section.

A. Outline of comparisons

Here we introduce the datasets, pre-processes, and algorithm specific settings used in the comparisons.

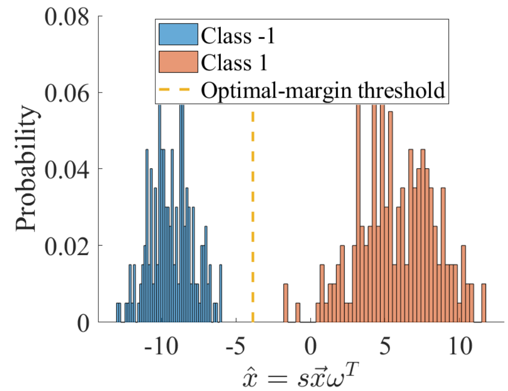

Fig. 4. The distribution of the instances after optimization of ω. The yellow dashed line indicates the optimal-margin threshold found by Algorithm 1.

1) Algorithm settings: We compared the performance of OEC with DEL, WSV, SLA, SVM, MBA, and SMP. We also compare the results against two other classification methods, Random Forest (RFT) [34] and Extreme Gradient Boosting (XGB) [35], used frequently in real-world applications recently. We used MATLAB 2017a for implementations and tests. The hyper-parameter values for these methods were as follows:

- DEL: 400 iterations of optimization, $\lambda = .5$ (regularization parameter), $\mu = 1e - 8$ (tolerance).
- WSV: $c_1 = 0.1$, $c_2 = 0.1$, $c_3 = 0.1$, $c_4 = 0.1$, suggested in the source code.
- SLA and SMP: none (no parameters).
- SVM: $\lambda = 1$ (the margin/accuracy balance factor).
- MBA: $\lambda = 0.5$ (the regularization factor).
- RFT: number of subtrees was set set experimentally.
- XGB: maximum depth and number of estimators were set experimentally.
- OEC: number of individuals $\lambda = 4 + 3\log(n)$, number of offspring (in ES and CMAES) was $\mu = \lambda/4$, maximum number of iterations $150\log(n + 1)$. The number of individuals was taken from the original source code of the CMAES while the number of iterations and offspring was set experimentally (see Section IV-B).

The parameters of RFT (number of subtree) and XGB (maximum depth and number of estimators) were set by a 5-fold cross-validation procedure and grid search for each dataset on the selected training data for each run. The search range for number of subtrees in RFT was between 100 to 600, step size of 100. This search range for the maximum depth of XGB was between 2 to 8, step size of 1, and the number of estimators was set between 200 to 800, step size of 100.

2) Datasets and performance measures: We used 15 real-world binary classification datasets (namely, Breast cancer (BC) [36], Crab gender (CG), Glass chemical (GC), Parkinson (PR) [37], Ionosphere (IS) [38], Pima Indians diabetes (PF) [39], German credit card (GR), Vote (VT), Madelon (MD) [40], Hill-Valley without noise (HV),

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2019.2895298, IEEE Transactions on Evolutionary Computation

TABLE I
THE DATASETS USED FOR COMPARISON PURPOSES IN THIS PAPER.

Dataset name	n	c	Number of instances in each class
BC	9	2	$< 458, 241 >$
CG	6	2	$< 100, 100 >$
GC	9	2	$< 51, 163 >$
PR	22	2	$< 48, 147 >$
IS	32	2	$< 225, 126 >$
PD	8	2	$< 268, 500 >$
GR	24	2	$< 300, 700 >$
VT	16	2	$< 192, 242 >$
NO	59	2	$< 365, 635 >$
SF	170	2	$< 59000, 1000 >$
MD	500	2	$< 1300, 1300 >$
HV	100	2	$< 606, 606 >$
HN	100	2	$< 606, 606 >$
SR	178	2	$< 9200, 2300 >$
PI	50	2	$< 95565, 36499 >$
SD	*	2	*
IR	4	3	$< 50, 50, 50 >$
IW	13	3	$< 59, 71, 48 >$
TF	21	3	$< 166, 368, 6666 >$
YD	8	10	$< 463, 5, 35, 44, 51, 163, 244, 429, 20, 30 >$
RQ	11	6	$< 10, 53, 681\ , 638, 199, 18 >$
WQ	11	7	$< 20, 163, 1457, 2198, 880, 175, 5 >$
HD	784	10	$< 6903, 7877, 6990, 7141, 6824, 6313, 6876, 7293, 6825, 6958 >$
HDC	500	10	Same as HD

* See text for details.

Hill-valley with noise (HN), Seizure recognition (SR) [41], Particle Identification (PI) [42], News Objectivity (NO) [43], Scania Failure (SF) [44]) and seven multi-class classification datasets (Iris (IR), Italian wine (IW), Thyroid function (TF), Yeast dataset (YD), Red wine quality (RW), White wine quality (WW), and Hand-written Dataset (HD), Handwritten digits-convolutional (HDC))[5] to compare methods. The main characteristics of these datasets have been provided in Table I. These datasets are used frequently as standard benchmarks in machine learning studies. In this table, n is the number of variables and c is the number of classes in each dataset. The number of instances in each class has been reported in the last column. We used the one-hot-encoding to encode categorical variables to binary values in datasets which include categorical variables.

The dataset Handwritten digits-convolutional (HDC) has been generated by characterizing the HD dataset using a convolutional neural network (CNN) [45]. Two convolutional layers with ReLU activation functions were used in the CNN (5×20 for the first layer and 3×20 for the second layer), each were then proceeded by a max pooling layer with strides of 2×2. A Softmax layer was then used for final classification. After the training of CNN, this Softmax layer was replaced by OEC, WSV, SVM, MBA, XGB, and RFT for comparison. This enables these methods to use features optimized by CNN in the convolutional layers for classification. The number of dimensions of the last layer before the Softmax was 500, hence, each classifier faced a 500 dimensional problem for classification.

We also used a dataset [46] on seizure detection introduced recently that includes characterized signals from 12 epileptic patients (seizure/non-seizure that is a binary classification problem). Training and testing took place for each patient independently. The number of features in this dataset is between 784 and 3,528 while the number of instances are between 174 to 5,240 in both classes (see [46] for details on the preprocessing and how the dataset was generated). The number of instances in the seizure and non-seizure classes is imbalance and follows the ration of 2:19 in average.

We structured our comparisons as follows:

1) We first compared three evolutionary algorithms, namely PSO, ES, and CMAES, for optimization of the objective function in Algorithm 2. This comparison provided insight on the best choice of optimization algorithm for OEC. The parameters of the best optimization method were then set experimentally.
2) We tested the sensitivity of OEC, SLA, and optimized SVM to the imbalanced number of instances in classes on a synthetic dataset, in addition to two other datasets (CG and PD, selected to clearly indicate the differences). Note that, SLA and optimized SVM are extremely slow on large problems, hence, they were not applied to all datasets for the purposes of this test.
3) We compared OEC with SLA, BVM, EVM, and BDA on binary classification problems. The aim of this comparison was to show how close OEC results are to solutions found by near-optimal classifiers such as SLA, optimized SVM, and optimized LDA. We run the algorithms for 100 times within each we selected 70% of instances from each class for training and the rest for testing (stratified sampling [47]). The training and testing sets remained constant for all methods.
4) We compared OEC with state-of-the-art classification algorithms, namely SVM, SMP, WSV, DEL, and MBA, on binary classification problems with and without noise and outliers. The aim of this comparison was to show whether OEC provides high-quality generalizable solutions to standard binary optimization problems within a practical amount of time. We also compared these methods on the seizure detection dataset [46].
5) We tested the impact of L_1 regularization (ω).
6) We compared OEC against SVM, MBA, DEL, WSV, RFT, and XGB on multi-class classification problems. We did not include SMP and SLA as they require a long time to achieve results on these datasets.

Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC) [48] curve and F1-score were used as performance measures. To test the performance of methods on imbalance datasets, we used F1-score

[5]All of these datasets, except for the HDC, are available online at https://archive.ics.uci.edu/ml/datasets.html

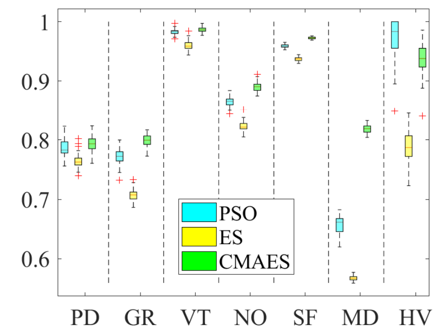

Fig. 5. Comparison between PSO, ES, and CMAES for selected binary classification problems. Results are the performance of each method (F1-score) for training.

rather than AUC to ensure fair comparison. The training dataset was normalized to ensure each feature has a zero mean and a unit variance. The transformation computed for normalization was then applied to the test set to ensure consistency between train and test domain. We used the Wilcoxon test ($p < 0.05$, corrected for multiple comparison using the Bonferroni correction) for our statistical comparisons.

B. Comparison between different EAs and parameter setting

We tested the performance of three EAs, namely PSO, ES, and CMAES, on 7 binary classification problems, namely PD, GR, VT, NO, SF, MD, and HV. Each method was applied to each dataset 50 times, with the ratio of 70% train and 30% test in each run, formed randomly. The number of iterations for all methods was set to $150 \log(n + 1)$ and the population size (λ) was set to $4 + 3log(n)$. Results in terms of the F1-score, shown in Figure 5, indicate that CMAES outperforms ES and PSO in almost all datasets. In terms of running time, however, PSO and ES were the fastest (with no significant difference, $p > 0.3$) while CMAES was the slowest, as expected. We use CMAES in the rest of this paper for optimization purposes because of the best performance it delivered in this test.

We also set the number of iterations of CMAES using PD, GR, VT, NO, SF, MD, and HV datasets in Table I. We run the algorithm 50 times on each dataset and recorded their objective value at every $\log(n + 1)$ iterations. According to the results reported in Figure 6, the iteration $150 \log(n + 1)$ provides sufficient performance for these datasets.

In order to set the value of μ, we applied CMAES to PD, GR, VT, NO, SF, MD, and HV datasets, 50 runs per setting. The value of μ was set to $\frac{\lambda}{i}$, where λ was the number of offspring and i was set to $\{1, 2, ..., 10\}$. According to the results reported in Figure 6, the choice of $\mu = \lambda/4$ seems a good choice for most of these datasets.

It is notable that the number of instances does not impact the best choice of the number of iterations and the

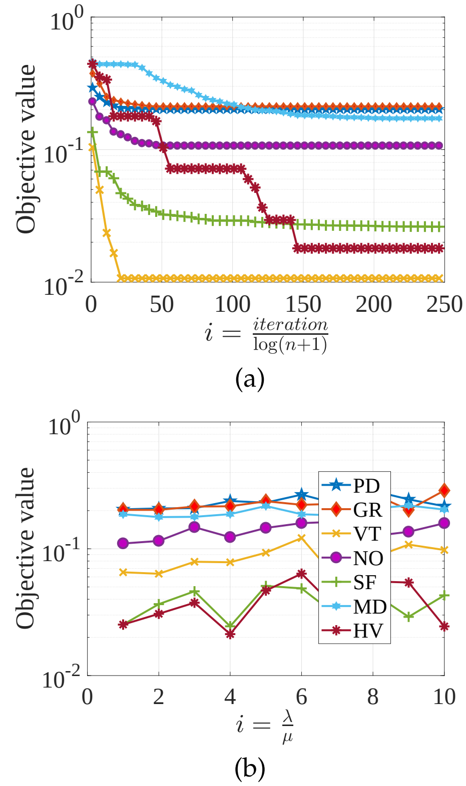

Fig. 6. (a) The objective value of OEC vs the number of iterations of CMAES. $150 \log(n+1)$ seems like a good choice to ensure an acceptable quality has been achieved. (b) The objective value of OEC vs $i = \lambda/\mu$ of CMAES. $i = 4$ seems like a good choice to ensure an acceptable quality has been achieved.

population size in OEC. The reason is that the number of instances does not necessarily impact the complexity of the objective function and the classification task. The number of instances, however, impacts the running time of the objective function.

C. Class-imbalance test

We tested the efficiency of OEC in classifying three datasets with imbalanced numbers of instances in each class. The first dataset was a synthetic binary classification dataset, generated using the procedure described in Section III-E, where we changed the mean (μ) of class $+1$ to $\mu =< -3, -6 >$. With this setting, the instances in the class $+1$ and -1 would overlap (see Fig 7(a)). We also used the CG and the PD datasets for this test. We generated the training set using 70% of instances from class -1 while changed the ratio of instances from class $+1$ appeared in the training set in the range of 5% to 70% with the step size of 5%. Results in Fig 7(b), (c), and (d) show that the performance of OEC (in terms of F1 score) is more robust against this imbalance comparing to both SLA and SVM on a synthetic dataset, CG dataset, and the PD dataset.

D. Comparison with near-optimal discriminative classifiers

We compared OEC with SLA, the fastest algorithm proposed in [5] to find a near-optimal separator hyper-

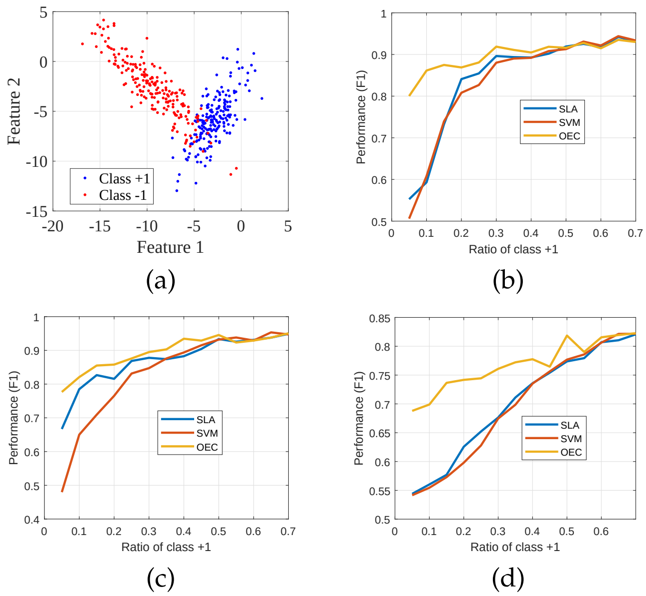

Fig. 7. Sensitivity to imbalance number of instances in the classes. Figure (a) is a synthetic dataset generated for the testing purposes. Results on (b) synthetic dataset, (c) the CG dataset, and (d) the PD dataset, have been reported. The reported performance is the performance of the methods in the test database.

TABLE II
COMPARISON AGAINST SLA AND SVM AND LDA WHEN THEIR PARAMETERS ARE OPTIMIZED.

Measure	**SLA**	**BVM**	**EVM**	**BDA**
Time	15(0)	15(0)	15(0)	15(0)
Train	13(0)	15(0)	15(0)	15(0)
Test	13(2)	9(6)	9(5)	11(2)

plane, BVM, EVM, and BDA (see Section II-D for details on these methods) in Table II (detailed results have been reported in the Appendix). Values in each column indicates the number of datasets for which OEC performed significantly (Wilcoxon, $p < 0.05$, Bonferroni corrected) better than the method mentioned in the heading of that column. The values in the parentheses indicate the number of datasets for which OEC performed significantly worse than the method mentioned in the heading of that column. OEC significantly outperformed SLA, BVM, EVM, and BDA in all tested datasets in training set. For the testing sets, OEC is significantly better than SLA, BVM, EVM, and BDA in 13, 9, 9, and 11 datasets. This indicates that, given the training dataset, OEC finds a better hyperplane to distinguish between classes in comparison to these other methods with optimal parameters. Also, the optimal-margin indeed improved the generalization ability of the algorithm to classify unseen cases correctly, evidenced by the better performance of OEC in the testing dataset. In terms of the running time, OEC outperformed all of these methods in all cases. The parameter setting for these methods was the main contributor to this longer running time.

TABLE III
COMPARISON RESULTS ON BINARY CLASSIFICATION METHODS, WITH AND WITHOUT NOISE.

Noise	Set	**DEL**	**WSV**	**SMP**	**SVM**	**MBA**
0%	Train	15(0)	14(0)	14(0)	15(0)	15(0)
	Test	10(1)	9(3)	7(0)	12(1)	10(0)
15%	Train	15(0)	15(0)	15(0)	14(0)	15(0)
	Test	12(0)	13(0)	9(1)	11(0)	13(0)

E. Binary classification with and without noise

We also compared OEC with state-of-the-art classification algorithms, namely DEL, WSV, SMP, SVM, and MBA, in Table III on both noisy and non-noisy data (complete report can be found in Appendix). The values indicate the number of datasets (over 15 in total) for which OEC outperforms other methods significantly ($p < 0.05$, Bonferroni corrected) in the training and testing sets. The values in parentheses show the number of datasets for which OEC was outperformed by another method significantly. Results are with noise (15%) and without noise.

Results for non-noisy datasets (row ”0% noise”) indicate that, comparing to all other algorithms, OEC finds a significantly better hyperplane to separate classes, given the training sets, in all 15 datasets except for one dataset in comparison to WSV, where the performance of OEC and WSV was similar ($p < 0.05$, Bonferroni corrected)[6]. OEC was also significantly better than other methods in the testing sets in most cases.

We used the procedure in [5] to introduce noise and outliers to the binary classification datasets in Table I. The noise was generated as a random value in $[min(x_i) - 0.5(max(x_i) - min(x_i)), max(x_i) + 0.5(max(x_i) - min(x_i))]$ (uniform distribution), where $min(x_i)$ is the minimum value for the i^{th} variable in the training set and $max(x_i)$ is the maximum value for the i^{th} variable in the training set. This formulation ensures that the instances are noisy while outliers are also likely to be generated. We selected 15% of instances from class $+1$ randomly and perturbed them using this procedure. Results in Table III, row noise ”15%”, indicate that the number of datasets for which OEC outperforms other classifiers has increased. This indicates that the performance drop in other methods was more severe than that of OEC.

We also compared OEC against DEL, WSV, SMP, SVM, and MBA (see Appendix for details of the results) on the Seizure Detection dataset, introduced in [46]. Our comparisons showed that OEC significantly (Wilcoxon, $p < 0.05$, Bonferroni corrected) outperforms DEL, WSV, SMP, SVM, and MBA on detecting seizure (testing dataset) for 11, 7, 10, 7, and 6 patients, respectively, over 12 epileptic patients. In contrast, OEC was significantly outperformed by DEL, WSV, SMP, SVM, and MBA in detecting seizure for 0, 4, 1, 5, 2 patients, respectively, over 12 epileptic patients. This indicates

[6]The dataset was the SR dataset.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2019.2895298, IEEE Transactions on Evolutionary Computation

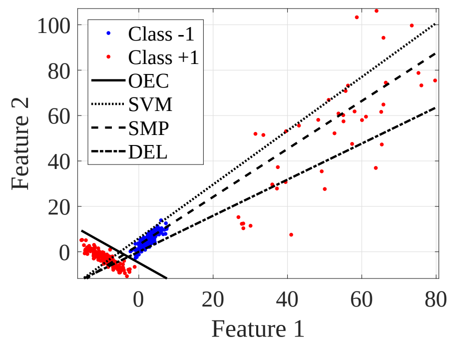

Fig. 8. A synthetic two-dimensional binary classification problem with outlier. All methods except OEC found poor quality separator hyperplanes except.

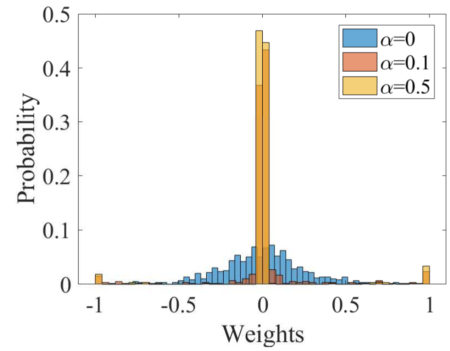

Fig. 9. Distribution of the weights with L_1 regularization factor (α) equal to 0, 0.1, and 0.5. Weights are more distributed around 0, 1, and -1 when the regularization factor is larger than zero.

that the proposed algorithm is more robust in the seizure detection task across patients.

F. Binary classification with outliers

To visualize the impact of outlier on OEC, DEL, SMP, and SVM, we generated a synthetic dataset using the procedure described in Section III-E and then added outlier instances to the class $+1$. The outliers were generated randomly using a normal distribution with covariance matrix $\begin{bmatrix} 8.66 & -5 \\ 15 & 25.98 \end{bmatrix}$ and $\mu =< 50, 50 >$. We compared the results of OEC for this dataset against SVM (hyperparameters for SVM were optimized by the Bayesian Optimization for this test), SMP, and DEL in Fig. 8. The hyperplane (line in 2-dimensional space) found by DEL, SMP, and SVM have been influenced significantly by outliers. OEC, however, still maintained its good performance and found a relatively better separator line. The AUC of the training set after optimization was 92.6, 66.5, 43.6, and 57.2 for OEC, SMP, SVM, and DEL, respectively.

G. Impact of L_1 regularization

Incorporating the L_1 regularization term to the optimization objective leads to a preference over weights that are closer to 0, 1, or -1 [8]. This makes the L_1 regularized solutions a good candidate for feature selection. Hence, we set the regularization factor α to test its impact on the found solutions. We ran OEC with $\alpha \in \{0, 0.1, 0.5\}$ and applied the algorithm to the binary classification problems in Table I. We ran the algorithm 10 times on each dataset and recorded the found ω. Figure 9 shows the histogram of weights after optimization.

It is clear that the weights found under larger α are more distributed around 0, 1, and -1. Note, however, that the performance of the algorithm dropped by 2% for $\alpha = 0.1$ and 5% for $\alpha = 0.5$ in average. This is expected as the L_1 regularization is more restrictive in terms of values for the coefficients.

TABLE IV
ONE-VS-ONE STRATEGY ON 8 DATASETS.

Dataset	**XGB**	**WSV**	**MBA**	**SVM**	**RFT**	**DEL**
Train	4(2)	8(0)	5(0)	5(0)	0(4)	8(0)
Test	8(0)	6(2)	6(2)	4(2)	5(1)	8(0)

H. Multi-class classification

We used the one-vs-one method to extend binary classification algorithms to deal with multiple classes. We run each algorithm (XGB, WSV, MBA, SVM, and RFT, and DEL) 100 times on the multi-class problems introduced in Table I. Results in Table IV (see Appendix for details of these results) show that the number of datasets for which OEC performs significantly better than other methods, in training and testing datasets, is larger than the number of datasets for which OEC performs significantly worse, with exception of training in RFT. RFT, however, is a non-linear classifier, hence, while it fits the training dataset well, its results are not generalizable, as it was also reported in [46]. This can be also confirmed by noticing that the performance of RFT was significantly lower than OEC's in the testing set.

V. CONCLUSION AND FUTURE WORK

Optimizing the total loss using the 0-1 loss function leads to the design of an efficient classification algorithm that finds the optimal separator hyperplane and is robust against outliers. This problem is, however, NP-Hard, making it an impractical approach for real-world problems. In addition, the 0-1 loss function does not provide any information about the empirical risk associated with the found hyperplane and is sensitive to imbalances in the number of instances in different classes. We proposed an efficient procedure to optimize the total loss of a modified 0-1 loss function in a one-dimensional case. We extended this approach using an evolutionary algorithm to solve multi-dimensional binary and multi-class classification problems. We designed an objective function for our method to ensure the separator hyperplane has an optimal margin (minimum empirical risk) from instances in each class, improving generalization ability for the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2019.2895298, IEEE Transactions on Evolutionary Computation

classification of unseen instances. We also added the L_1 regularization term to the algorithm to enable its use for feature selection. We then compared the proposed algorithm, the Optimal-margin Evolutionary Classifier (OEC), on 23 standard classification problems against six state-of-the-art classification algorithms. Results showed that, on the tested datasets, OEC is unbeaten over training sets among linear classifiers, even in comparison to the most recent algorithms that provably find a near-optimal separator hyperplane. The generalization ability of the algorithm was also better than other algorithms in our tests, indicating that OEC reliably classifies unseen instances. We also showed that the modified 0-1 loss function is less sensitive to class imbalance. To the best of our knowledge, OEC is the first algorithm that optimizes the 0-1 loss function directly and considers the empirical risk (maximum margin), and yet has a running time that is practical for real-world applications. These results are exciting as they encourage the use of evolutionary algorithms for classification, a task that is often, in practice, on a high demand.

One potential next step is to add non-linearity to the algorithm to deal with non-linear classification. One possibility is to incorporate non-linear functions frequently used in Perceptrons [3]. The running time of the algorithm can also be improved by incorporating smart initialization, i.e., hybridization with SVM or LDA. Bagging and boosting [49] are other possibilities to improve the generalization ability of the proposed OEC. Graphical Processing Unit (GPU) implementation of OEC may also be another promising direction to ultimately yield an effective evolutionary-based classifier for real-world problems. Finally, OEC is a classifier that requires characterization of the raw data for effective performance. Addition of techniques to form features form the raw data (e.g., optimization of multiple layers of convolution filters, similar to the CNN) could be another promising direction. This can be done by optimizing convolution filters and the OEC coefficients at the same time using an evolutionary algorithm.

Appendix

Table V shows the full comparison results between OEC and other classification methods on 15 binary classification problems, introduced in the paper. The values in each row are the average values over 100 runs. The character "*" on a value indicates that the OEC corresponding result was significantly better than this reported value (Wilcoxon test, $p < 0.05$, Bonferroni corrected). The character "+" indicates that this result was significantly better than that of OEC's. The character "-" indicates that the result of OEC was statistically the same as this reported value. The rows profixed by "Train" indicate the accuracy of the methods on the training data, while the rows prefixed by "Test" indicate the accuracy of the methods on the testing data.

Table V shows the full comparison results between OEC and other classification methods on binary classification problems in terms of their running time. Results are the average time in milliseconds over 100 runs. For some cases, the time has been reported in seconds for which the time values has been post-fixed by the character 's'.

Table VII shows the full comparison results between OEC and other classification methods on binary classification problems with 15% of noise. Results are the average over 100 runs.

Table VIII shows the full comparison results between OEC and other classification methods on multi-class classification problems. Results are the average over 100 runs. For the HDC dataset, the original CNN achieved

TABLE V
COMPARISON RESULTS ON 15 BINARY CLASSIFICATION PROBLEMS.

	Data	**DEL**	**WSV**	**SMP**	**SVM**	**MBA**	**OEC**
Train	BC	96.89*	94.18*	97.45*	96.88*	95.34*	**98.42**
	CG	94.6*	96.49*	95.39*	95.17*	95.43*	**98.37**
	GC	92.33*	88.26*	93.47*	92.4*	89.48*	**96.99**
	PR	78.95*	88.42*	85.58*	82.23*	85.07*	**92.44**
	IS	87.21*	70.54*	89.79*	94.2*	88.58*	**95.15**
	PD	75.57*	74.6*	75.63*	73.09*	73.03*	**79.28**
	GR	73.25*	74.08*	74.31*	71.17*	71.32*	**79.8**
	VT	91.09*	92.02*	91.03*	91.4*	91.46*	**93.66**
	NO	80.61*	65.53*	84.14*	83.76*	81.51*	**89.04**
	SF	89.96*	87.67*	94.83*	85.96*	85.18*	**97.98**
	MD	72.13*	79.49*	76.22*	79.31*	76.25*	**81.99**
	HV	63.53*	72.9*	73.04*	58.67*	69.1*	**93.04**
	HN	63.56*	75.46*	73.12*	61.82*	71.69*	**87.81**
	SR	61.31*	74.14-	62.24*	58.02*	58.96*	**74.53**
	PI	84.61*	87.39*	**89.71-**	85.72*	87.33*	88.29
Test	BC	96.48*	93.67*	**97.24-**	96.21*	94.86*	96.99
	CG	94.57*	**96.52-**	95-	95.32-	95.38-	95.25
	GC	89.68-	85.56*	90.09-	87*	86.47*	**90.3**
	PR	74.52*	**80.14-**	77.85*	77.11*	79.28-	79.23
	IS	81.7-	68.05*	83.2-	**84.59+**	81.78-	82.42
	PD	**74.57+**	73.97-	73.82-	72.34*	72.33*	73.11
	GR	70.87-	**71.71+**	71.68-	68.53*	69.09*	70.73
	VT	90.72-	**92.06+**	90.68-	90.66-	90.69-	90.57
	NO	77.91*	63.05*	81.38*	80.21*	77.98*	**82.03**
	SF	87.85*	85.15*	94.05*	80.1*	82.68*	**95.72**
	MD	54.37*	54.19*	54.93-	54.4*	54.95-	**55.15**
	HV	62.65*	72.22*	70.84*	58.47*	66.73*	**92.66**
	HN	61.86*	72.93*	69.07*	60.39*	67.54*	**85.15**
	SR	56.47*	**72.75+**	56.65*	55.17*	55.95*	68.14
	PI	84.72*	86.91*	86.11*	85.91*	87.34*	**88.31**

TABLE VI
RUNNING TIME COMPARISON AMONG CLASSIFICATION METHODS ON 15 BINARY CLASSIFICATION PROBLEMS.

Data	**DEL**	**WSV**	**SMP**	**SVM**	**MBA**	**OEC**
BC	15	0.6	112.5	13.9	6.9	78.2
CG	10.7	0.3	98.7	5.5	4.1	38.8
GC	11.2	0.4	105.2	5.1	4.1	51
PR	14.3	0.4	121.2	5.9	4.3	101.3
IS	19.6	0.5	131.6	11.9	4.7	150.1
PD	14.9	0.5	99.8	22.7	4.4	72.5
GR	26.9	0.6	107	106.3	5.3	151.7
VT	12.6	0.4	104.7	5.3	4.2	59.6
NO	47	1.2	170.6	222.8	12.2	722.9
SF	7.7*s*	287.6	1.74*s*	364.2*s*	715.8	66.6*s*
MD	2.4s	43.9	4.0*s*	28.4*s*	92.9	12.1*s*
HV	67.5	1.4	231	155.3	8.3	655
HN	82.9	1.8	301.8	128.8	9	924
SR	1.9*s*	41.5	927.7	141.9*s*	95.6	7.5*s*
PI	5.2*s*	137.3	648.9	289.9*s*	429.8	41.9*s*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2019.2895298, IEEE Transactions on Evolutionary Computation

TABLE VII
COMPARISON RESULTS WITH 15% NOISE ON 15 BINARY CLASSIFICATION PROBLEMS.

	Data	DEL	WSV	SMP	SVM	MBA	OEC
Train	BC	95.35*	94.58*	95.48*	95.23*	94.47*	**97.13**
	CG	90.25*	83.57*	89.14*	93.24*	91.96*	**96.3**
	GC	88.56*	65.22*	93.61*	92.85*	89.53*	**96.78**
	PR	77.42*	80.28*	85.6*	84.08*	83.6*	**92.04**
	IS	88.74*	73.43*	90.21*	95.13*	88.32*	**95.81**
	PD	74.99*	75.31*	74.46*	70.97*	70.51*	**78.35**
	GR	72.29*	70.98*	76.49*	73.99*	72.54*	**81.71**
	VT	96.31*	96.35*	96.07*	97.66*	96.07*	**98.75**
	NO	77.54*	61.33*	81.24*	82.46*	79.31*	**88.94**
	SF	85.18*	86.77*	91.92*	84.11*	83.48*	**96.54**
	MD	67.49*	65.4*	77.12*	79.28-	76.98*	**79.45**
	HV	63.15*	61.31*	71.69*	63.36*	68.39*	**90.66**
	HN	62.44*	63.44*	66.31*	62.98*	65.86*	**86.24**
	SR	49.91*	54.98*	54.31*	55.08*	50.56*	**73.59**
	PI	81.33*	83.19*	83.99*	84.31*	84.57*	**87.29**
Test	BC	**96.98**-	95.08*	96.6+	95.56*	94.81*	96.3
	CG	90.43*	82.07*	89.8*	93.53*	92.55*	**94.3**
	GC	84.47*	56.26*	**90.35**-	86.4*	85.31*	90.17
	PR	72.57*	70.43*	**78.96**-	78.14-	77.95-	76.53
	IS	81.18-	68.25*	82.32-	**83.5**-	79.28*	82.86
	PD	74.15-	**74.7**-	73.03-	70.36*	70.17*	73.74
	GR	67.44*	65.25*	**71.66**-	70.38-	69.42*	70.62
	VT	83.99*	94.14-	94.04*	94.18-	94.41-	**94.45**
	NO	75.31*	61.55*	80.48*	79.11*	75.38*	**81.31**
	SF	83.38*	84.95*	90.15*	79.91*	81.19*	**93.88**
	MD	52.99*	51.18*	54.89*	54.72*	54.73*	**57.74**
	HV	59.79*	53.71*	66.53*	56.66*	62.57*	**92.73**
	HN	58.56*	53.98*	58.74*	55.19*	58.01*	**85.66**
	SR	41.45*	49.81*	47.11*	53.65*	50.38*	**67.54**
	PI	80.12*	81.83*	82.18*	81.83*	81.55*	**86.31**

TABLE VIII
COMPARISON ON 8 MULTI-CLASS CLASSIFICATION DATASETS USING THE ONE-VS-ONE STRATEGY.

	Data	XGB	WSV	MBA	SVM	RFT	OEC
Train	IR	97.33*	98.09*	98.33*	98.23*	**100**-	99.49
	IW	96.26*	99.39*	99.99-	**100**-	100-	100
	TF	95.39*	66.98*	94.06*	83.57*	**100**+	97.85
	YD	87.84+	69.12*	76.05*	79.99*	**100**+	83.44
	RQ	85.79+	73.06*	73.76*	59.96*	**100**+	81.22
	WQ	64.12*	68.85*	63.19*	56.7*	**100**+	74.91
	HD	100-	99.66*	100-	100-	100-	100
	HDC	**100**-	100-	100-	100-	100-	100
Test	IR	91.05*	97.83+	**98.13**+	97.7+	96.08*	96.67
	IW	94.31*	98.34+	**98.78**+	97.65-	98.1-	97.8
	TF	95.01*	66.2*	92.74*	82.31*	**99.35**+	95.6
	YD	74.19*	67.5*	72.86*	**78.33**+	73.1*	75.73
	RQ	61.95*	61.82*	62.44*	58.6*	63.5*	**64.04**
	WQ	56.55*	58.41*	59.88*	56.21*	61.1-	**61.49**
	HD	89.99*	86.11*	85.64*	94.32-	93.01*	**94.99**
	HDC	97.45*	97.18*	97.11*	98.59*	98.11*	**98.75**

98.64 on the test (in average), that was statistically similar to OEC. Note that all methods used the features optimized by the CNN during the learning process.

Table IX shows the comparison results between OEC and other methods. Results are the average over 100 runs on the testing dataset. Table X compares OEC and classification methods which provide near optimal solutions to the classification problem. Results are the average over 100 runs. The running time of the methods was limited to 2 hours.

TABLE IX
COMPARISON RESULTS ON THE SEIZURE DETECTION DATASET. THE DATASET INCLUDES THE TEST RESULTS FOR 12 EPILEPTIC PATIENTS.

Subject	DEL	WSV	SMP	SVM	MBA	OEC
Subject-1	84.89*	93.69*	77.73*	**95.15**+	82.71*	94.21
Subject-2	55.46*	82.6*	61.19*	79.11*	77.54*	**85.99**
Subject-3	83.27*	91.51*	79.34*	89.85*	91*	**92.42**
Subject-4	86.46*	**94.75**+	62.64*	77.17*	80.52*	90.27
Subject-5	85.73-	**89.03**+	87.28-	85.01*	79.54*	88.73
Subject-6	81.08*	**97.04**+	66.16*	95.05+	96.7+	92.11
Subject-7	57.02*	58.76*	62.31*	62.22*	63.31-	**64.17**
Subject-8	**58.01***	53.55-	54.88+	56.08+	56.11+	53.58
Subject-9	59.9*	72.54*	68.5*	84.1*	68.06*	**86.26**
Subject-10	78.76*	84.46*	59.32*	85.84-	**86.96**-	86.24
Subject-11	81.79*	96.06*	66.6*	**98.87**+	98.38-	97.61
Subject-12	81.93*	**94.62**+	58.33*	91.8+	88.63-	88.82

TABLE X
COMPARISON RESULTS ON 15 BINARY CLASSIFICATION PROBLEMS AMONG OEC, OPTIMIZED SVM (BAYESIAN AND EVOLUTIONARY OPTIMIZATION), OPTIMIZED LDA (BAYESIAN OPTIMIZATION), AND SLA. TIMES ARE IN MILLISECOND. THE "-" VALUES INDICATE THAT THE METHOD COULD NOT CONVERGE WITHIN 2 HOURS INTERVAL.

	Dataset	SLA	BVM	EVM	BDA	OEC
Time	BC	8195.9	38103.7	5341.2	10473.1	254.4
	CG	1932.1	15156.8	1292.3	11269	102.7
	GC	3294.8	20134	4351.2	8071	136.5
	PR	5569.1	18248.7	5926.8	8734.4	319.4
	IS	4547.7	20361.2	3198.1	8877.2	426.3
	PD	36750.5	32589.4	16253.6	9515.6	191.5
	GR	88368.7	71190.8	19366.7	9306.9	502.4
	VT	5057.4	34350	6788.4	9331.4	310.5
	NO	120637	70081.8	44567.1	11659.7	953.3
	SF	-	6993639.2	834554.8	69779	60467.8
	MD	-	169911.3	81892.3	21045.2	20675.6
	HV	-	86630	39442.2	9121.3	1279.1
	HN	-	288494.2	100334.8	17812.5	1684
	SR	-	7182037.2	988311.7	20669.4	13654.2
	PI	-	-	-	48810.9	41169.7
Train	BC	98.46-	97.12*	97.18*	96.54*	**98.75**
	CG	97.06-	96.43*	96.16*	96.14*	**97.14**
	GC	92.71*	93.14*	93.11*	87.15*	**96.44**
	PR	82.35*	74.51*	74.49*	77.96*	**92.69**
	IS	90.87*	90.87*	90.51*	88.06*	**96.31**
	PD	73.89*	72.11*	71.48*	71.82*	**78.9**
	GR	74.18*	69.29*	69.12*	69.35*	**78.03**
	VT	97.65*	95.88*	95.88*	95.35*	**98.08**
	NO	86.47*	80.72*	80.75*	75.84*	**88.36**
	SF	-	84.42*	84.92*	85.1*	**97.41**
	MD	-	66.15*	63.72*	66.65*	**82.31**
	HV	-	91.1*	88.58*	67.65*	**92.58**
	HN	-	83.76*	83.89*	71.29*	**87.71**
	SR	-	61.48*	61.19*	64.59*	**74.12**
	PI	-	-	-	78.16*	**85.95**
Test	BC	**96.09**+	95.43+	95.33+	92.66*	93.28
	CG	96.67*	96.67*	96.61*	93.33*	**98.07**
	GC	85.83*	**92.5**+	92.5+	90.21-	90.63
	PR	82.14*	82.14*	82.14*	79.87*	**84.9**
	IS	81.54*	80.94*	81.08*	79.59*	**84.11**
	PD	71.54*	70.79*	70.01*	70.83*	**72.54**
	GR	70.00*	70.00*	70.00*	71.35*	**75.48**
	VT	95.5*	**97.13**+	97.13+	97.13+	96.25
	NO	83.66+	**84.06**+	84.01+	74.16*	80.7
	SF	-	80.73*	80.75*	85.33*	**94.23**
	MD	-	56.28+	55.91+	**57.82**+	54.87
	HV	-	90.01*	90.21*	66.96-	**92.56**
	HN	-	**86.33**+	85.73-	68.78*	85.49
	SR	-	60.22*	58.11*	62.83*	**67.21**
	PI	-	-	-	78.18*	**86.04**

REFERENCES

[1] A. Y. Ng and M. I. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes," *Advances in neural information processing systems*, vol. 2, pp. 841–848, 2002.

[2] V. Vapnik, *Statistical learning theory*. Wiley, New York, 1998.

[3] S. Haykin and N. Network, "A comprehensive foundation," *Neural Networks*, vol. 2, no. 2004, p. 41, 2004.

[4] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.

[5] T. Nguyen and S. Sanner, "Algorithms for direct 0–1 loss optimization in binary classification," in *International Conference on Machine Learning*, 2013, pp. 1085–1093.

[6] S. Ben-David, N. Eiron, and P. M. Long, "On the difficulty of approximately maximizing agreements," *Journal of Computer and System Sciences*, vol. 66, no. 3, pp. 496–514, 2003.

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

[8] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[9] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[10] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural processing letters*, vol. 9, no. 3, pp. 293–300, 1999.

[11] Y.-H. Shao, C.-H. Zhang, X.-B. Wang, and N.-Y. Deng, "Improvements on twin support vector machines," *IEEE transactions on neural networks*, vol. 22, no. 6, pp. 962–968, 2011.

[12] R. Khemchandani, S. Chandra *et al.*, "Twin support vector machines for pattern classification," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 5, pp. 905–910, 2007.

[13] Y.-H. Shao, W.-J. Chen, Z. Wang, C.-N. Li, and N.-Y. Deng, "Weighted linear loss twin support vector machine for large-scale classification," *Knowledge-Based Systems*, vol. 73, pp. 276–288, 2015.

[14] G. Melki, V. Kecman, S. Ventura, and A. Cano, "Ollawv: Online learning algorithm using worst-violators," *Applied Soft Computing*, vol. 66, pp. 384 – 393, 2018.

[15] G. R. Lanckriet, L. E. Ghaoui, C. Bhattacharyya, and M. I. Jordan, "A robust minimax approach to classification," *Journal of Machine Learning Research*, vol. 3, no. Dec, pp. 555–582, 2002.

[16] H. Xue, S. Chen, and Q. Yang, "Structural regularized support vector machine: a framework for structural large margin classifier," *IEEE Transactions on Neural Networks*, vol. 22, no. 4, pp. 573–587, 2011.

[17] B. Gu, X. Sun, and V. S. Sheng, "Structural minimax probability machine," *IEEE Transactions on Neural Networks and Learning Systems*, 2017.

[18] S. H. Walker and D. B. Duncan, "Estimation of the probability of an event as a function of several independent variables," *Biometrika*, vol. 54, no. 1-2, pp. 167–179, 1967.

[19] Z. Zhang, Z. Lai, Y. Xu, L. Shao, J. Wu, and G.-S. Xie, "Discriminative elastic-net regularized linear regression," *IEEE Transactions on Image Processing*, vol. 26, no. 3, pp. 1466–1481, 2017.

[20] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[21] Z. Wang, Y.-H. Shao, L. Bai, C.-N. Li, L.-M. Liu, and N.-Y. Deng, "Mblda: A novel multiple between-class linear discriminant analysis," *Information Sciences*, vol. 369, pp. 199–220, 2016.

[22] M. R. Bonyadi and Z. Michalewicz, "Particle swarm optimization for single objective continuous space problems: a review," *Evolutionary computation*, 2016.

[23] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies–a comprehensive introduction," *Natural computing*, vol. 1, no. 1, pp. 3–52, 2002.

[24] W. Jia, D. Zhao, and L. Ding, "An optimized rbf neural network algorithm based on partial least squares and genetic algorithm for classification of small sample," *Applied Soft Computing*, vol. 48, pp. 373–384, 2016.

[25] S. Ding, H. Li, C. Su, J. Yu, and F. Jin, "Evolutionary artificial neural networks: a review," *Artificial Intelligence Review*, vol. 39, no. 3, pp. 251–260, 2013.

[26] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, Jun. 2002.

[27] S. Ding, C. Su, and J. Yu, "An optimizing bp neural network algorithm based on genetic algorithm," *Artificial Intelligence Review*, vol. 36, no. 2, pp. 153–162, 2011.

[28] C.-H. Wu, G.-H. Tzeng, Y.-J. Goo, and W.-C. Fang, "A real-valued genetic algorithm to optimize the parameters of support vector machine for predicting bankruptcy," *Expert systems with applications*, vol. 32, no. 2, pp. 397–408, 2007.

[29] M. A. Gelbart, J. Snoek, and R. P. Adams, "Bayesian optimization with unknown constraints," in *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, ser. UAI'14. Arlington, Virginia, United States: AUAI Press, 2014, pp. 250–259.

[30] J. Yang and V. Honavar, "Feature subset selection using a genetic algorithm," in *Feature extraction, construction and selection*. Springer, 1998, pp. 117–136.

[31] U. Kamath, A. Shehu, and K. D. Jong, "Using evolutionary computation to improve svm classification," in *IEEE Congress on Evolutionary Computation*, July 2010, pp. 1–8.

[32] M. Panda and A. Abraham, "Hybrid evolutionary algorithms for classification data mining," *Neural Computing and Applications*, vol. 26, no. 3, pp. 507–523, 2015.

[33] P. Branco, L. Torgo, and R. P. Ribeiro, "A survey of predictive modeling on imbalanced domains," *ACM Computing Surveys (CSUR)*, vol. 49, no. 2, p. 31, 2016.

[34] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[35] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. ACM, 2016, pp. 785–794.

[36] M. Tan and L. Eshelman, "Using weighted networks to represent classification knowledge in noisy domains," in *Machine Learning Proceedings 1988*. Elsevier, 1988, pp. 121–134.

[37] M. A. Little, P. E. McSharry, E. J. Hunter, J. Spielman, L. O. Ramig *et al.*, "Suitability of dysphonia measurements for telemonitoring of parkinson's disease," *IEEE transactions on biomedical engineering*, vol. 56, no. 4, pp. 1015–1022, 2009.

[38] V. G. Sigillito, S. P. Wing, L. V. Hutton, and K. B. Baker, "Classification of radar returns from the ionosphere using neural networks," *Johns Hopkins APL Technical Digest*, vol. 10, no. 3, pp. 262–266, 1989.

[39] J. W. Smith, J. Everhart, W. Dickson, W. Knowler, and R. Johannes, "Using the adap learning algorithm to forecast the onset of diabetes mellitus," in *Proceedings of the Annual Symposium on Computer Application in Medical Care*. American Medical Informatics Association, 1988, p. 261.

[40] I. Guyon, J. Li, T. Mader, P. A. Pletscher, G. Schneider, and M. Uhr, "Competitive baseline methods set new standards for the nips 2003 feature selection benchmark," *Pattern recognition letters*, vol. 28, no. 12, pp. 1438–1444, 2007.

[41] A. Temko, A. Sarkar, and G. Lightbody, "Detection of seizures in intracranial eeg: Upenn and mayo clinic's seizure detection challenge," in *Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE*. IEEE, 2015, pp. 6582–6585.

[42] B. P. Roe, H.-J. Yang, J. Zhu, Y. Liu, I. Stancu, and G. McGregor, "Boosted decision trees as an alternative to artificial neural networks for particle identification," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 543, no. 2-3, pp. 577–584, 2005.

[43] N. Hajj, Y. Rizk, and M. Awad, "A subjectivity classification framework for sports articles using cortical algorithms for feature selection," *Neural Computing and Applications*, vol. to appear, 2018.

[44] C. Gondek, D. Hafner, and O. R. Sampson, "Prediction of failures in the air pressure system of scania trucks using a random forest and feature engineering," in *International Symposium on Intelligent Data Analysis*. Springer, 2016, pp. 398–402.

[45] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[46] M. R. Bonyadi, Q. M. Tieng, and D. C. Reutens, "Optimization of distributions differences for classification," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–13, 2018.

[47] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Ijcai*, vol. 14, no. 2. Stanford, CA, 1995, pp. 1137–1145.

[48] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (roc) curve." *Radiology*, vol. 143, no. 1, pp. 29–36, 1982.

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