IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 7, 2019, accepted November 26, 2019, date of publication December 4, 2019,
date of current version December 16, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2957516

A Greedy Method for Constructing Minimal Multiconlitron

Q. LENG[1], Y. LIU[1], X. ZHAO[1], L. ZHANG[1], AND Y. QIN[2]
[1]College of Information Science and Technology, Bohai University, Jinzhou 121000, China
[2]College of Engineering, Bohai University, Jinzhou 121000, China

Corresponding author: Q. Leng (qkleng@gmail.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61602056 and Grant 61572082, in part by the National Social Science Fund of China under Grant 19BTQ028, in part by the Natural Science Foundation of Liaoning Province of China under Grant 20180550525 and Grant 2019-ZD-0493, and in part by the Scientific Research Project of Liaoning Provincial Committee of Education under Grant LQ2019012.

ABSTRACT Multiconlitron is a general theoretical framework for constructing piecewise linear classifier. However, it contains a relatively large number of linear functions, resulting in complicated model structure and poor generalization ability. Learning to prune redundant or excessive components may be a very necessary progression. We propose a novel greedy method, i.e., greedy support multiconlitron algorithm (GreSMA) to simplify the multiconlitron. In GreSMA, a procedure of greedy selection is first used. It generates the initial linear boundaries, each of which can separate maximum number of training samples under the current iteration. In this way, a minimal set of decision functions is established. In the second stage of GreSMA, a procedure of boundary adjustment is designed to retrain the classification boundary between convex hulls of local subsets, instead of individual samples. Thus, the adjusted boundary will fit the data more closely. Experiments on both synthetic and real-world datasets show that GreSMA can produce minimal multiconlitron with better performance. It meets the criteria of ''Occam's razor'', since simpler model can help prevent over-fitting and improve the generalization ability. More significantly, the proposed method does not contain parameters that depend on the datasets or make assumptions of the underlying statistical distributions of the samples. Therefore, it should be regarded as an attractive advancement of piecewise linear learning in the general framework of multiconlitron.

INDEX TERMS Greedy method, model simplification, multiconlitron, piecewise linear classifier, support vector machine.

I. INTRODUCTION

In pattern recognition, piecewise linear classifier (PLC) is effective when a statistical model cannot express the underlying distribution of samples [1]. It approximates the true classification boundary by a combination of hyperplanes. Since each piece is linear, a PLC is very simple to implement with requirement of low memory usage. Therefore, it has the potential to be applied to the scenarios of small reconnaissance robots, intelligent cameras, embedded and real-time systems, and portable devices [2].

Despite the simplicity in implementation, constructing a PLC usually requires complex computational procedure [3]. In general, there are two criteria that need to be considered, i.e., selecting appropriate number of hyperplanes and minimizing the error of classification. Under their guidance, many methods have been presented to synthesize PLCs over the last few decades.

Hierarchical partitioning is one of the common ways. In 1996, Chai *et al.* [4] achieved a binary tree structure with genetic algorithm to design a PLC in the sense of maximum impurity reduction. For simplifying the construction of a decision-tree PLC, in 2006 Kostin [2] developed and implemented a simple and fast multi-class PLC with acceptable classification accuracies, based on tree division of subregion centroids. In 2016, Wang *et al.* [5] proposed hierarchical mixing linear support vector machines (SVMs) for nonlinear classification, which can be seen as special form of a PLC. Furthermore, Ozkan *et al.* [6] designed a highly dynamical self-organizing decision tree structure for mitigating overtraining issues. The resulting PLC adaptively partitions the feature space into small regions and

The associate editor coordinating the review of this manuscript and approving it for publication was Bilal Alatas.

combines the local classification models specialized in those regions. However, for the hierarchical structure, it will produce cumulative classification errors, affecting the prediction accuracy.

The optimization method is also employed to exploit piecewise linear structure [7]. As early as 1968, Mangasarian [8] used a linear programming to construct the PLCs. With a reference of Mangasarian's work, Herman and Yeung [9] proposed a similar but better method based on linear abnormality functions. However, it needs to perform the optimization process many times, involving a lengthy computation. From 2005 to 2015, Bagirov *et al.* [10]–[12] and Ozturk *et al.* [13] proposed and improved an approach of max-min separability. In this approach, a piecewise linear function with the max-min form is determined by using the discrete gradient to minimize the nonconvex and nonsmooth error function. The resulting classifier has good discriminant ability, but the relevant optimization problem is difficult to solve and the number of hyperplanes needs to be pre-determined.

The local training method is another strategy to construct PLCs. In 1980, Sklansky and Michelotti [14] proposed a versatile technique for piecewise linear classification. This technique first uses Forgy's algorithm to form prototypes, then finds all close-opposed pairs of prototypes to carry out local training. Eventually, a piecewise-linear approximation of the Bayes-optimum decision surface is generated. Along with the idea of local training, Park and Sklansky [15] further described a Tomek-link-cutting method for designing a multi-class PLC. However, it may suffer from underfitting due to insufficient hyperplanes. For addressing the problem, Tenmoto *et al.* [1] employed minimum description length (MDL) to choose an appropriate number of hyperplanes. In 2010, Gai and Zhang [16] introduced a two-step method to develop discriminative piecewise linear model. In the first step, some boundary points are sampled and a nonparametric decision surface is determined. To simplify the surface, in the second step an approach for linear surface segmentation is presented using Dirichlet process mixtures. However, this method requires a hypothetical statistical distribution of data.

In fact, it is a hard task to pre-determine the number of hyperplanes or assume the statistical distribution of data. The question that deserves our consideration is whether we can design a better PLC without these constraints. In 2011, Li *et al.* [17] presented a general framework for constructing a PLC, named multiple convex linear perceptron (Abbr. multiconlitron). In the framework, each hyperplane of multiconlitron can be dynamically constructed without pre-specified number and distribution assumption. From the viewpoint of large margin, multiconlitron can be regarded as the nonkernel generalization of SVMs. Recently, some new variants and applications on multiconlitron have emerged, such as alternating multiconlitron [18] and quasi-linear SVM [19].

The advantages of multiconlitron over SVMs are as follows:

- It does not need to achieve space mapping and kernel selection, whereas there are some difficulties in the selection of kernel functions and in explaining the change of spatial metric for SVMs [20].
- The construction algorithm is quite straightforward to implement without a lengthy optimization process. In addition, it only includes a precision parameter that can be set directly without tuning.

The original method for constructing a multiconlitron is called support multiconlitron algorithm (SMA) [17]. However, considering the crucial effect of model complexity on prediction, a multiconlitron by SMA may contain a relatively large number of linear functions, which results in poor generalization ability.

In this paper, we propose a greedy method to improve multiconlitron for designing better PLCs. This method sequentially generate decision hyperplanes of multiconlitron, each of which achieves a maximum separation of training samples under the current iteration. This procedure is called greedy selection. It aims to simplify the classification model of a PLC by reducing the number of linear functions. In order to maintain the merit of the local margin maximization, a procedure of boundary adjustment is further introduced. More samples are used to retrain the classification hyperplanes so that they are adjusted to more appropriate positions. The two procedures constitute the main body of the greedy method.

In the convexly separable case, we develop a new iterative algorithm, i.e., greedy support conlitron algorithm (GreSCA) to simplify the convex linear perceptron (Abbr. conlitron). Using GreSCA as the key component, in the commonly separable case, we establish a greedy support multiconlitron algorithm (GreSMA) for constructing minimal multiconlitron. Then, we evaluate the performance of GreSMA with SMA, Kostin's decision tree, linear SVM, and radial basis function (RBF) SVM on both synthetic datasets and UCI benchmark datasets. Finally, we summarize the main contributions and discuss future research.

II. PRELIMINARIES

A. CONVEX HULL AND SEPARABILITY

We first give one definition and three lemmas to illustrate the relationship between the separability of datasets and the convex hulls.

Definition 1: let R^n be the n-dimensional Euclidean space. For any finite set $X \subseteq R^n$, its convex hull (CH) is defined as

$$CH(X) = \left\{ x \,\middle|\, x = \sum_{1 \le i \le |X|} \alpha_i x_i, \sum_{1 \le i \le |X|} \alpha_i = 1, x_i \in X, \alpha_i \ge 0 \right\} \tag{1}$$

where $|X|$ stands for the cardinality of the set X. Next, we introduce the separability of datasets [17].

Lemma 1: Given two finite datasets $X, Y \subseteq R^n$, if their convex hulls are disjoint, namely $CH(X) \cap CH(Y) = \emptyset$, then X and Y are linearly separable.

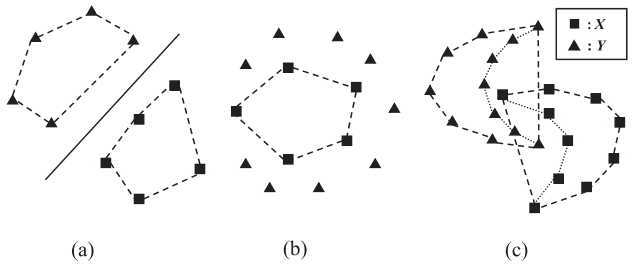


FIGURE 1. Separability description of datasets: (a) X and Y are linearly separable; (b) X is convexly separable to Y; (c) X and Y are commonly separable.

Lemma 2: For two finite datasets $X, Y \subseteq R^n$, if $\forall y \in Y, y \notin CH(X)$, then X is convexly separable to Y. If X is convexly separable to Y, or Y is convexly separable to X, we call X and Y convexly separable.

Lemma 3: For two finite datasets $X, Y \subseteq R^n$, if $X \cap Y = \emptyset$, i.e., they have no common points, then X and Y are commonly separable.

Fig.1 shows an example for illustrating the three separable cases. Note that the appellation "commonly separable" is introduced not only for the sake of simplicity, but also for the conceptual comparison with "linearly separable" and "convexly separable".

B. HARD-MARGIN SVM

Hard-margin SVM is an optimized linear classifier, which is closely related to the convex hulls of datasets [21]. For two finite datasets $X, Y \subseteq R^n$, if they are linearly separable, solving their hard-margin SVM can be converted into the problem of computing the nearest point pair between $CH(X)$ and $CH(Y)$ [22].

We here introduce the Schlesinger-Kozinec (SK) algorithm [23] to express the hard-margin SVM, and also use it as the basis for subsequent algorithms. For the linearly separable datasets X and Y, SK algorithm will find the nearest point pair between their convex hulls to construct a ε-optimal hyperplane. The optimization goal is

$$\min \|x - y\| \quad s.t.\, x \in CH(X), y \in CH(Y). \tag{2}$$

If the nearest point pair (x^*, y^*) is the solution of (2), namely $d(CH(X), CH(Y)) = \|x^* - y^*\|$, the separating hyperplane can be calculated as the perpendicular bisector of x^* and y^*,

$$f(x) = w^* \cdot x + b, \tag{3}$$

where $w^* = x^* - y^*$, $b = (\|y^*\|^2 - \|x^*\|^2)/2$. The details of SK algorithm are described in Algorithm-1.

The SK algorithm starts from two arbitrary points $x^* \in X$ and $y^* \in Y$. Then it searches a point $v_t \in X \cup Y$ which allows $\frac{v_t - y^*}{x^* - y^*}$ to obtain the smallest projection length on the vector $(x^* - y^*)$, i.e., $m(v_t) = \min\{m(v_i), i \in \{1, ..., |X| + |Y|\}\}$ (see Fig.2). Assume that under the current situation, y^* is the nearest point from $CH(Y)$ to x^*. Then we will find $v_t = x_t$ to evaluate the ε-optimal criterion. If the criterion $||x^* - y^*|| - m(v_t) < \varepsilon$ is not satisfying, then y^* is fixed and x^* is moved towards y^* by update rule $x^*_{new} = (1-q)x^* + qx^*$. q is determined so that the distance between x^*_{new} and y^* is minimal.

Algorithm 1 Schlesinger-Kozinec (SK) Algorithm

Input: Two finite disjoint sets $X, Y \subseteq R^n$, precision parameter ε.

1: Pick $x^* \in X, y^* \in Y$;

2: $v_t = \arg\min_{i \in \{1,...,|X|+|Y|\}} m(v_i)$, where

$$m(v_i) = \begin{cases} \frac{(v_i - y^*)\cdot(x^* - y^*)}{\|x^* - y^*\|}, & for\ v_i \in X, \\ \frac{(v_i - x^*)\cdot(y^* - x^*)}{\|x^* - y^*\|}, & for\ v_i \in Y. \end{cases}$$

3: **if** $\|x^* - y^*\| - m(v_t) < \varepsilon$, **goto** Step 5;

4: **if** $v_t \in X, x^* = (1-q)x^* + qv_t$, where $q = min(1, ((x^* - y^*) \cdot (x^* - v_t))/||x^* - v_t||^2)$;
else if $v_t \in Y, y^* = (1-q)y^* + qv_t$, where $q = min(1, ((y^* - x^*) \cdot (y^* - v_t))/||y^* - v_t||^2)$;
goto Step 2;

5: $w^* = x^* - y^*, b = (||y^*||^2 - ||x^*||^2)/2$;

Output: $f(x) = w^* \cdot x + b$.

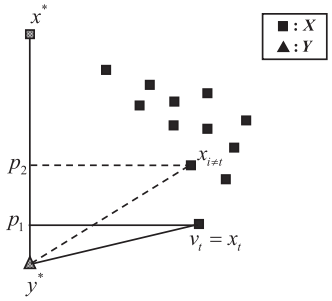


FIGURE 2. The geometric meaning of $m(v_t)$: $m(v_t = x_t) = ||p_1 - y^*|| < m(x_{i \neq t}) = ||p_2 - y^*||$.

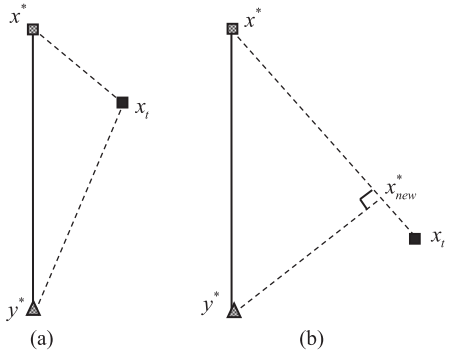


FIGURE 3. The geometric meaning of x^*_{new}: (a) $q = 1, x^*_{new} = x_t$; (b) $0 < q < 1, x^*_{new} = (1-q)x^* + qx_t$.

Fig.3 shows the update rule of SK algorithm. If $q = 1$, then $x^*_{new} = x_t$ is a point in X. if $0 < q < 1$, then $x^*_{new} = (1-q)x^* + qx_t$ is the vertical point from y^* to the line segment $CH\{x^*, x_t\}$. If x^* is not the nearest point from $CH(X)$ to y^*, there must exist another point $x^*_{new} = x_t$ or $x^*_{new} = (1-q)x^* + qx_t$, so that $d(x^*_{new}, y^*) < d(x^*, y^*)$. Since the dataset is finite, we can easily get that the SK algorithm must be convergent. It can finally find the nearest point pair between two convex hulls, and calculate the classification hyperplane as $f(x) = 0$.

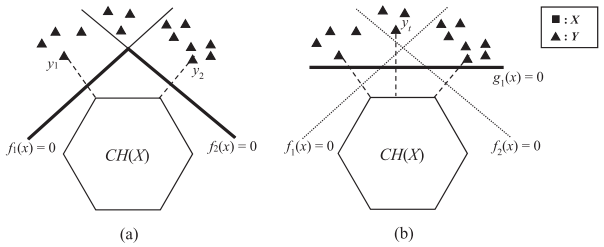


FIGURE 4. Intuitive difference between (a) SCA and (b) GreSCA.

The time complexity of SK algorithm is estimated as $O(D\cdot(|X|+|Y|)/\varepsilon)$, where $D=\max_{x\in X,y\in Y}\{||x-y||\}$, and ε is a precision parameter, for controlling the convergence condition. The value of D/ε is related to the distribution of samples, indicating the maximum number of iterations.

III. GREEDY SUPPORT CONLITRON ALGORITHM

Based on the concept of "convexly separable", the conlitron (convex linear perceptron) [17] is presented. A conlitron is a set of linear functions, which can separate any two convexly separable datasets. For $X, Y\subseteq R^n$, if X is convexly separable to Y, then there must exist a conlitron from X to Y denoted as

$$CLP=\{f_l(x)=w_l\cdot x+b_l,(w_l,b_l)\in R^n\times R,1\le l\le L\} \tag{4}$$

satisfying

$$\begin{aligned}&\forall x\in X,\forall 1\le l\le L,f_l(x)=w_l\cdot x+b_l>0,\\&\forall y\in Y,\exists 1\le l\le L,f_l(y)=w_l\cdot y+b_l<0.\end{aligned} \tag{5}$$

The decision function of a conlitron from X to Y is defined as

$$CLP(x)=\begin{cases}+1,\ \forall 1\le l\le L,f_l(x)>0,\\-1,\ \exists 1\le l\le L,f_l(x)<0.\end{cases} \tag{6}$$

Equation 6 represents the discriminant model of the conlitron. Each $f_l(x)$ in it is a linear discriminant function. The original method for constructing a conlitron is called support conlitron algorithm (SCA). Given training direction from X to Y, in each iteration, SCA chooses the nearest point y_t from Y to $CH(X)$, and then constructs a hard-margin SVM, $f(x)$, which cuts off some points in Y. When Y is empty the SCA will finish executing. Eventually, a series of linear functions constitute a conlitron. The time complexity of SCA is estimated as $O(D\cdot(|X|\cdot|Y|)/\varepsilon)$. The geometric explanation of SCA can be seen in Fig.4a.

Obviously, SCA calculates each linear function between $CH(X)$ and an individual point of Y. From a quantitative viewpoint, individual training points will produce a large number of linear functions, leading to a decline in generalization ability. In this paper, we present a greedy method, i.e., greedy support conlitron algorithm (GreSCA) to construct minimal conlitron by generating as few linear functions as possible.

Algorithm 2 Greedy Support Conlitron Algorithm (GreSCA)

Input: Two finite disjoint sets $X, Y\subseteq R^n$, precision parameter ε.

1: $l\leftarrow 1$, $m=|Y|$;
2: $G(x)=\{g_i(x)|g_i(x)=SK(X,\{y_i\},\varepsilon),1\le i\le m\}$;
3: $p=\arg\max_i\{CutNum(Y,G(x))\}$;
4: $Y_t=\{y|g_p(y)\le g_p(y_p),y\in Y\}$, $G_t(x)=\{g_j(x)|y_j\in Y_t\}$;
5: $f_l(x)=SK(X,Y_t,\varepsilon)$;
6: $Y=Y-Y_t$, $G(x)=G(x)-G_t(x)$;
7: $l\leftarrow l+1$;
8: **if** $Y\neq\emptyset$, **goto** Step 3;
9: $CLP=\{f_i(x),1\le i\le l\}$;

Output: A conlitron CLP.

Given the training direction from X to Y, in each iteration, GreSCA chooses the linear function $g_p(x)$ as the initial decision function that can separate the most points of Y from X. We define these separated points of Y as the local training samples corresponding to $g_p(x)$. They satisfy $g_p(y_i)\le g_p(y_p)$ and form a subset $Y_t\subseteq Y$. Subsequently, Y_t will be cut off and removed from Y through the hyperplane $g_p(x)=0$. Different from the original SCA that uses the nearest point to construct a separating hyperplane, GreSCA produces a hyperplane by finding the point that achieves a maximum separation between X and Y. It ultimately reduces the number of linear functions with the greedy strategy. Fig.4 shows the intuitive differences between SCA and GreSCA. In Fig.4a, SCA generates two linear functions $f_1(x)$ and $f_2(x)$ by locating the nearest points y_1 and y_2 from Y to $CH(X)$. In Fig.4b, GreSCA finds the point y_t and generates one linear function $g_1(x)$. Using the greedy strategy, it is easy to see that $g_1(x)$ cuts off more training points.

The procedure, called greedy selection, repeats similarly until no point left in Y. As a result, a series of linear functions constitute an initial conlitron. Since Y is a finite set, after cutting for many times, this procedure will stop, satisfying $Y=\emptyset$. Algorithm-2 shows the details of GreSCA. In it, "*CutNum*" is used to compute the number of points cut off by $g_i(x)$.

From Algorithm-2, we can see that each linear function $g_i(x)\in G(x)$ can separates $CH(X)$ from a subset $Y_t\subseteq Y$. However, $g_i(x)$ was initially trained by SK using an individual point in Y and the total points in X. From a statistical point of view, an individual training point does not represent a real class boundary well, and it will also be affected by noise. Hence, in Step 5 of the GreSCA, we retrain the classification boundary by using the total points of Y_t and the total points of X, so that the initial boundary is adjusted to appropriate position.

An intuitive explanation of the boundary adjustment is shown in Fig.5. The initial boundary $g_1(x)=0$ is obtained by greedy schema from an individual point y_t to $CH(X)$ (see Fig.5a). It cuts off a set of points Y_t satisfying $g_i(y_i\in Y_t)>0$. Using the procedure of boundary adjustment, the new

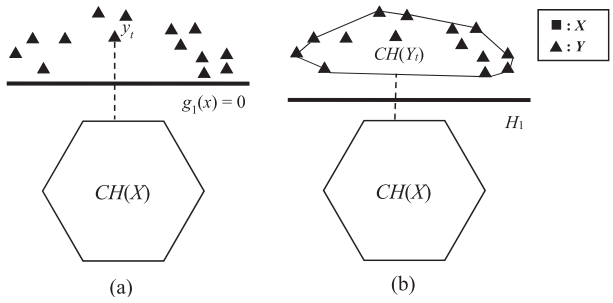


FIGURE 5. Boundary adjustment for GreSCA: (a) the initial boundary; (b) the adjusted boundary.

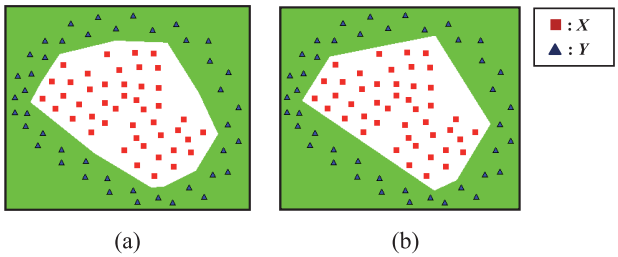


FIGURE 6. Two conlitrons respectively generated by (a) SCA and (b) GreSCA.

boundary H_1 is obtained by retraining between $CH(Y_t)$ and $CH(X)$ (see Fig.5b). It is worth noting that the classification boundary is changed from $g_1(x) = 0$ to H_1, which includes not only the procedure of boundary adjustment, but also the procedure of greedy selection. The two procedures constitute the main body of the GreSCA.

Fig.6 shows a real example for illustrating the difference of constructing conlitron between GreSCA and SCA. The conlitron by SCA (see Fig.6a) contains 11 linear functions, whereas the conlitron by GreSCA (see Fig.6b) contains only 6 linear functions. Intuitively, GreSCA with greedy strategy produces a simpler model structure than SCA. It is expected that GreSCA will perform better in high-dimensional or complex classification.

Since both X and Y are finite sets, GreSCA will converge after a limited number of iterations. Its time complexity is higher than the original SCA. For convenience, we discuss the time complexity of GreSCA in two parts. The first part indicates the time GreSCA performs the procedure of greedy selection. It is similar to the original SCA and can be roughly evaluated as $O(D \cdot (|X| \cdot |Y|)/\varepsilon)$. The second part represents the time used for the procedure of boundary adjustment. In the case that X is convexly separable to Y, each adjustment invokes the SK algorithm once and computes a separating hyperplane between X and a subset $Y_t \subseteq Y$. We consider the worst situation that $Y_t = Y$ and then we can get the complexity of $O(D \cdot (|X| + |Y|)/\varepsilon)$ for one adjustment. Since the maximum number of boundary adjustment is $|Y|$, we can further obtain the time complexity of $O(|Y| \cdot (D \cdot (|X| + |Y|)/\varepsilon))$. Similarly, when Y is convexly separable to X, the time complexity of boundary adjustment is $O(|X| \cdot (D \cdot (|X| + |Y|)/\varepsilon))$.

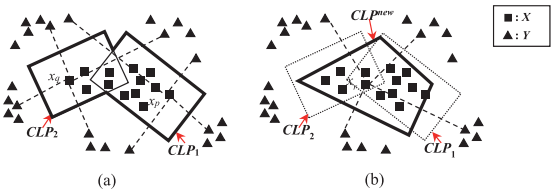


FIGURE 7. Intuitive difference between (a) SMA and (b) GreSMA.

Finally, we summarize the results with two parts and estimate the overall time complexity of GreSCA as $O(D \cdot (|X| + |Y|)^2/\varepsilon)$.

IV. GREEDY SUPPORT MULTICONLITRON ALGORITHM

If two finite datasets $X, Y \subseteq R^n$ are convexly separable, they can be separated by a conlitron. However, if X and Y are commonly separable (see Lemma 3), a conlitron might not work. Ref. [17] has proved that for any two finite disjoint datasets, there must exist a multiconlitron for separating them.

Multiconlitron is a union of multiple conlitrons. Given the training direction from X to Y, a multiconlitron can be expressed as $MCLP = \{CLP_k, 1 \le k \le K\}$, satisfying

$$\begin{aligned} &\forall x \in X, \exists 1 \le k \le K, CLP_k(x) = +1 \\ &\forall y \in Y, \forall 1 \le k \le K, CLP_k(y) = -1 \end{aligned} \tag{7}$$

Accordingly, the decision function is defined as

$$MCLP(x) = \begin{cases} +1, & \exists 1 \le k \le K, CLP_k(x) = +1 \\ -1, & \forall 1 \le k \le K, CLP_k(x) = -1 \end{cases} \tag{8}$$

Equation 8 represents the discriminant model of the multiconlitron. Each $CLP_k(x)$ in it is a convex discriminant conlitron, namely a convex polyhedron. The original method for constructing a multiconlitron is called support multiconlitron algorithm (SMA). Given training direction from X to Y, SMA chooses the nearest point x_t from X to Y in each iteration. Since x_t is convexly separable to Y, we can construct a conlitron from x_t to Y, which cuts off a number of points in X. The SMA will not stop execution until $X = \emptyset$. Finally, a union of multiple conlitrons constitute a multiconlitron. The time complexity of SMA is estimated as $O(|X| \cdot |Y| \cdot (|X| + |Y|))$. The geometric explanation to SMA can be seen in Fig.7a.

Note that SMA generates each conlitron by an individual point of X and the total points of Y. However, each conlitron contains a large number of linear functions. From the quantitative viewpoint, a multiconlitron usually has a complex model structure, resulting in poor generalization ability. Therefore, we present a greedy method, i.e., greedy support multiconlitron algorithm (GreSMA) to construct a minimal multiconlitron for simplifying the classification model.

Given the training direction from X to Y, GreSMA first produces a candidate set of conlitrons, each of which is trained between Y and an individual point in X. Next, in each iteration, GreSMA chooses the $gCLP_p(x) \in GCLP$ as the initial conlitron. It can surround the most points of X in a convex polyhedron that is defined by $gCLP_p(x) = +1$ under the current iteration. These points surrounded by $gCLP_p(x) = +1$

Algorithm 3 Greedy Support Multiconlitron Algorithm (GreSMA)

Input: Two finite disjoint sets $X, Y \subseteq R^n$, precision parameter ε.

1: $k \leftarrow 1,\ n = |X|$;
2: $GCLP = \{gCLP_i | gCLP_i = GreSCA(\{x_i\}, Y, \varepsilon), 1 \le i \le n\}$;
3: $p = \arg\max_i \{CutNum(X,\ gCLP_i)\})$;
4: $X_t = \{x | gCLP_p(x) = +1, x \in X\}$, $GCLP_t = \{gCLP_j | x_j \in X_t\}$;
5: $CLP_k = GreSCA(X_t, Y, \varepsilon)$;
6: $X = X - X_t$, $GCLP = GCLP - GCLP_t$;
7: $k \leftarrow k + 1$;
8: **if** $X \neq \emptyset$, **goto** Step 3;
9: $MCLP = \{CLP_i, 1 \le i \le k\}$;

Output: A multiconlitron $MCLP$.

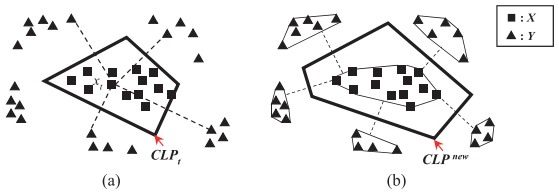


FIGURE 8. Boundary adjustment for GreSMA: (a) the initial boundary; (b) the adjusted boundary.

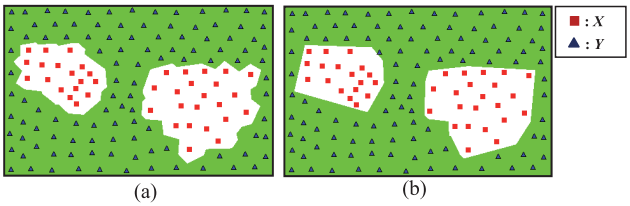


FIGURE 9. Two multiconlitrons respectively generated by (a) SMA and (b) GreSMA.

are defined as local training samples. They form a subset $X_t \subseteq X$. Subsequently, X_t will be removed from X. Different from the original SMA that uses the nearest point to construct a conlitron, GreSMA produces a conlitron by finding the point that implements a maximum separation between X and Y. After repeated iterations of the algorithm, it can ultimately reduce the number of the global conlitrons. Fig.7 shows the intuitive differences between SMA and GreSMA. In Fig.7a, SMA generates two conlitrons CLP_1 and CLP_2 by two points $x_p, x_q \in X$. In Fig.7b, GreSMA finds the point $x_t \in X$ and generates only one conlitron CLP^{new}. By adopting the greedy strategy, each conlitron in the multiconlitron may surround more points.

The procedure of greedy selection repeats similarly until no point left in X. Consequently, a series of conlitrons constitute an initial multiconlitron. Algorithm-3 shows the description of GreSMA. As in GreSCA, we still use "$CutNum$" to compute the number of points surrounded by $gCLP_i$.

Through Algorithm-3, we can get that each conlitron $gCLP_i \in GCLP$ can separates a subset $X_t \subseteq X$ from Y. However, $gCLP_i$ was initially trained by GreSCA using an individual point in X and the total points in Y. From the statistical point of view, an individual training point cannot represent a subset of X well. The resulting conlitron may not be a good approximation to the real classification boundaries. Therefore, in Step 5 of GreSMA, we employ a procedure of boundary adjustment, for retraining the boundary by using the total points of X_t and the total points of Y. X_t is initially established via the procedure of greedy selection. It is convexly separable to Y (Lemma 2). We can use GreSCA to construct a conlitron between X_t and Y, i.e., a convex separator consisting of multiple hyperplanes. The separator surrounds X_t in a convex area and excludes Y outside. Each separating hyperplane in it is exactly the perpendicular bisector connecting the nearest points of $CH(X_t)$ and $CH(Y_t)$. Obviously, the set of local training samples (X_t) is the unique subset of X under the current iteration. Correspondingly, the adjusted boundary is also unique. Since the boundary is calculated between convex hulls instead of individual points, the initial boundary can be adjusted, and its position will be more reasonable in general.

The boundary adjustment for GreSMA is shown in Fig.8. The initial conlitron CLP_t is obtained by greedy strategy between Y and an individual point x_t in X (see Fig.8a). It surrounds a set of points X_t in X, excluding Y outside. After using the procedure of boundary adjustment, the new conlitron CLP^{new} is trained between two point sets, X_t and Y (see Fig.8b). It is worth noting that the decision function is adjusted from CLP_t to CLP^{new}, which contains not only the greedy selection of the conlitrons, but also the appropriate adjustment of the boundary.

Fig.9 shows the difference for constructing multiconlitron between SMA and GreSMA. The multiconlitron by SMA (see Fig.9a) contains 36 conlitrons and 86 linear functions, whereas the multiconlitron by GreSMA (see Fig.9b) contains only 12 conlitrons and 44 linear functions. Obviously, GreSMA with greedy strategy produces a simpler classification model structure than SMA. In the experimental section, we will show that the multiconlitrons by GreSMA have better classification ability on real-world datasets.

Since both X and Y are finite sets, GreSMA will converge after a limited number of iterations. Its time complexity is higher than the original SMA. As for GreSCA, we also discuss the time complexity of GreSMA in two parts. The first part indicates the time of performing the procedure of greedy selection. It is similar to the original SMA and can be roughly evaluated as $O(|X| \cdot |Y| \cdot (|X| + |Y|))$. The second part represents the time used for the procedure of boundary adjustment. When the training direction is from X to Y, each adjustment must call GreSCA once between a subset of X, X_t, and Y. In the worst case that $X_t = X$, one adjustment will take GreSMA a GreSCA's time, i.e., $O(D \cdot (|X| + |Y|)^2/\varepsilon)$. The maximum number of boundary adjustment for GreSMA is $|X|$. We can get the time complexity is $O(|X| \cdot (D \cdot (|X| + |Y|)^2/\varepsilon))$ for all adjustments. Similarly, when the training

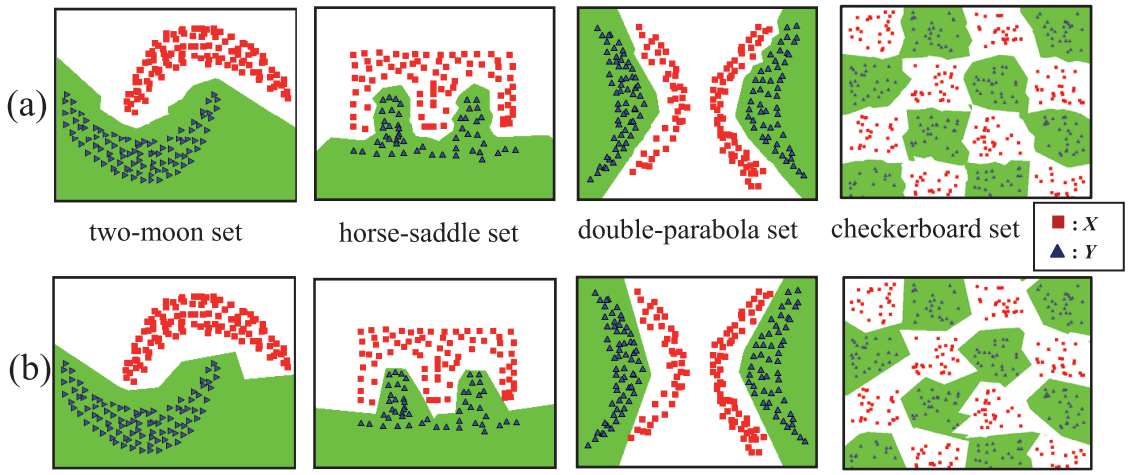


FIGURE 10. Comparative experiments for (a) SMA and (b) GreSMA on four synthetic datasets.

direction is from Y to X, the time complexity of boundary adjustment is $O(|Y| \cdot (D \cdot (|X| + |Y|)^2/\varepsilon))$

Finally, we summarize the results of greedy selection and boundary adjustment and estimate the overall time complexity of GreSMA as $O((|X| + |Y|) \cdot (|X|^2 + |Y|^2)/\varepsilon)$.

V. EXPERIMENTAL RESULTS

In this section, we conduct numerical experiments to evaluate the performance of GreSMA. Since the current GreSMA is designed only for two classes, we select a number of two-class datasets and report the experimental results. We first compare GreSMA with SMA on 4 synthetic datasets. Then, we compare GreSMA with SMA, KDT (Kostin's decision tree synthesized with piecewise linear boundaries [2]), linear SVM, and RBF SVM on 12 benchmark datasets. All experiments are performed on a PC with I5-3230M(2.60GHz), 4-GB memory, and Windows8.1 operating system. The precision parameter ε is set to 10^{-3}.

A. EXPERIMENTS ON THE SYNTHETIC DATASETS

To intuitively compare the decision boundaries generated by SMA and GreSMA, we introduce 4 synthetic datasets, i.e., two-moon dataset [17], horse-saddle dataset [16], double-parabola dataset [24], and checkerboard dataset [25]. For each of them, GreSMA and SMA are respectively applied. Fig.10 shows the real classification boundaries, and Table 1 lists the corresponding number of conlitrons (CLPs) and the number of linear functions (LFs).

From Fig.10 and Table 1, we can see that the multiconlitrons (i.e., classification models) computed by GreSMA are generally simpler than that computed by SMA. Especially for the checkerboard dataset, the multiconlitron by SMA contains 164 conlitrons and 822 linear functions, whereas the multiconlitron by GreSMA contains only 19 conlitrons and 64 linear functions. Since a simpler model will have the effect of preventing overfitting, we expect that GreSMA can outperform SMA in generalization ability.

TABLE 1. Comparison of GreSMA and SMA on four synthetic datasets.

Datasets	CLPs		LFs	
	GreSMA	SMA	GreSMA	SMA
two-moon set	3	12	5	43
horse-saddle set	5	24	12	90
double-parabola set	4	54	7	185
checkerboard set	19	164	64	822

TABLE 2. Datasets used in the experiments.

Name	Code	Examples	Dim.
Breast Cancer Wisconsin	BRE	569	30
German Credit Data	GER	1000	24
Heart	HEA	297	13
Ionosphere	ION	351	34
Magic Gamma Telescope	MAG	19020	10
Musk (Version 1)	MUS	476	166
Parkinsons	PAR	195	22
Pima-indians-diabetes	PIM	768	8
Sonar	SON	208	60
Monks-1	MO1	124+432	6
Monks-2	MO2	169+432	6
Monks-3	MO3	122+432	6

B. EXPERIMENTS ON BENCHMARK DATASETS

We further evaluate GreSMA on 12 selected benchmark datasets. These datasets are from the UCI machine learning repository [26], and listed in Table 2. For the last three datasets, we perform evaluation on them by using the indicated training and testing sets. For the first nine datasets, we randomly divide each of them into two halves for 10 times, one half for training and the other for testing, and then report the results averaged over the 10 times.

By scaling each feature of all datasets to [0, 1], we perform the comparison experiment of GreSMA with SMA and KDT.

TABLE 3. Testing accuracies of GreSMA with SMA, KDT, linear SVM, and RBF SVM.

Data	GreSMA	SMA	KDT	Linear SVM	RBF SVM
BRE	91.89±1.03	91.86±0.90	89.05±1.06	90.60±1.02	**94.04±1.49**
GER	67.36±1.91	64.58±1.27	**68.18±2.33**	**72.18±1.28**	**73.60±1.56**
HEA	61.78±3.68	59.41±4.14	**63.41±4.28**	**69.63±3.39**	**81.33±3.77**
ION	88.18±1.67	86.93±2.48	86.53±2.86	84.77±3.19	**88.64±2.58**
MAG	78.86±0.55	77.45±0.47	67.94±6.26	78.81±0.12	**85.98±0.25**
MUS	83.89±1.78	82.97±1.84	69.62±3.61	80.79±2.42	**90.38±1.78**
PAR	82.04±2.69	**82.76±3.02**	73.78±5.71	78.06±2.46	80.61±2.59
PIM	66.22±2.20	**67.89±1.89**	**69.17±2.60**	65.76±1.13	**75.60±1.91**
SON	79.33±2.88	**79.71±3.62**	71.24±1.78	73.81±5.54	**80.95±3.17**
MO1	93.75	91.90	65.74	68.52	86.34
MO2	76.62	74.77	60.42	61.34	**80.32**
MO3	91.90	87.50	85.65	73.15	**95.60**

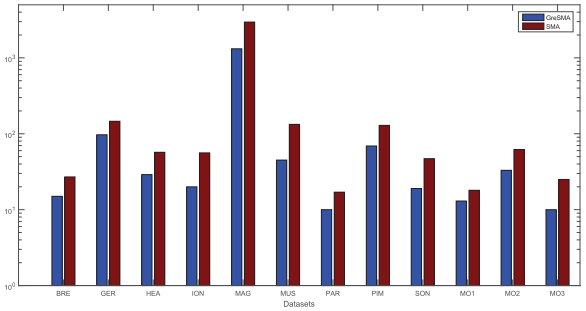
We also employ a library of SVMs, i.e., LIBSVM [27] to conduct experiments. Two types of SVMs are used as baselines, namely, linear SVM with parameter C and RBF SVM with parameters (C, γ). The optimal values for C and γ, from candidate sets $\{2^i | i = -4, -3, \ldots, 3, 4\}$ and $\{2^i | i = -7, -6, \ldots, 4, 5\}$, are determined by the grid search scheme and 10-fold cross validation.

We summarize the testing accuracies of GreSMA with SMA, KDT, linear SVM, and RBF SVM in Table 3. It is shown that GreSMA has higher accuracies than SMA on nine datasets (BRE, GER, HEA, ION, MAG, MUS, MO1, MO2, MO3), but slightly lower on the remaining three datasets (PAR, PIM, SON). This means that the multiconlitrons generated by GreSMA with greedy strategy obtain a better classification ability in general. Compared with KDT that is essentially a PLC based on the tree division of the subregion centroids, GreSMA performs better on nine datasets (BRE, ION, MAG, MUS, PAR, SON, MO1, MO2, MO3). More remarkably, on the "MO1" dataset, GreSMA has a testing accuracy of 93.75%, which is much higher than that of 65.74% for KDT. Compared with the baselines, GreSMA outperforms linear SVM on ten datasets (BRE, ION, MAG, MUS, PAR, PIM, SON, MO1, MO2, MO3), but not performing as well as RBF SVM except for two datasets (PAR, MO1). Although the RBF SVM is more accurate than GreSMA, it has to tune the two parameters C and γ at the same time, which may be inconvenient in practical applications. Therefore, we expect the proposed method to be prioritized in some important real-world scenarios, such as real-time systems and portable devices.

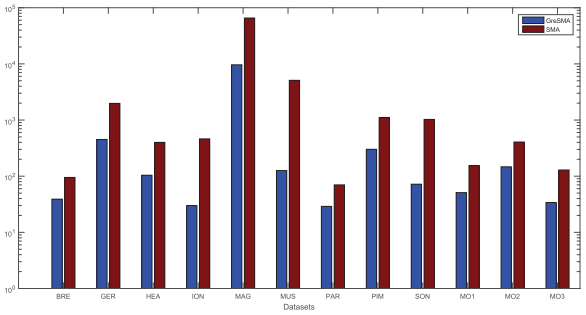
Table 4 and Fig. 11 provide the comparative results of GreSMA with SMA on the numbers of conlitrons and linear functions. We can see that on all selected benchmark datasets, the multiconlitron by GreSMA contains much fewer conlitrons and much fewer linear functions than the multiconlitron by SMA. For example, on the dataset MUS, the multiconlitron by GreSMA has only 45 conlitrons and 126 linear functions, whereas the multiconlitron by SMA has 133 conlitrons and 5095 linear functions. The simplified model is

TABLE 4. Comparison of GreSMA and SMA on the numbers of conlitrons and linear functions.

Data	CLPs		LFs	
	GreSMA	SMA	GreSMA	SMA
BRE	15	27	39	95
GER	97	146	451	1981
HEA	29	57	104	400
ION	20	56	30	462
MAG	1317	2961	9628	65567
MUS	45	133	126	5095
PAR	10	17	29	70
PIM	69	129	301	1111
SON	19	47	72	1026
MO1	13	18	51	155
MO2	33	62	147	406
MO3	10	25	34	129



(a) Number of conlitrons

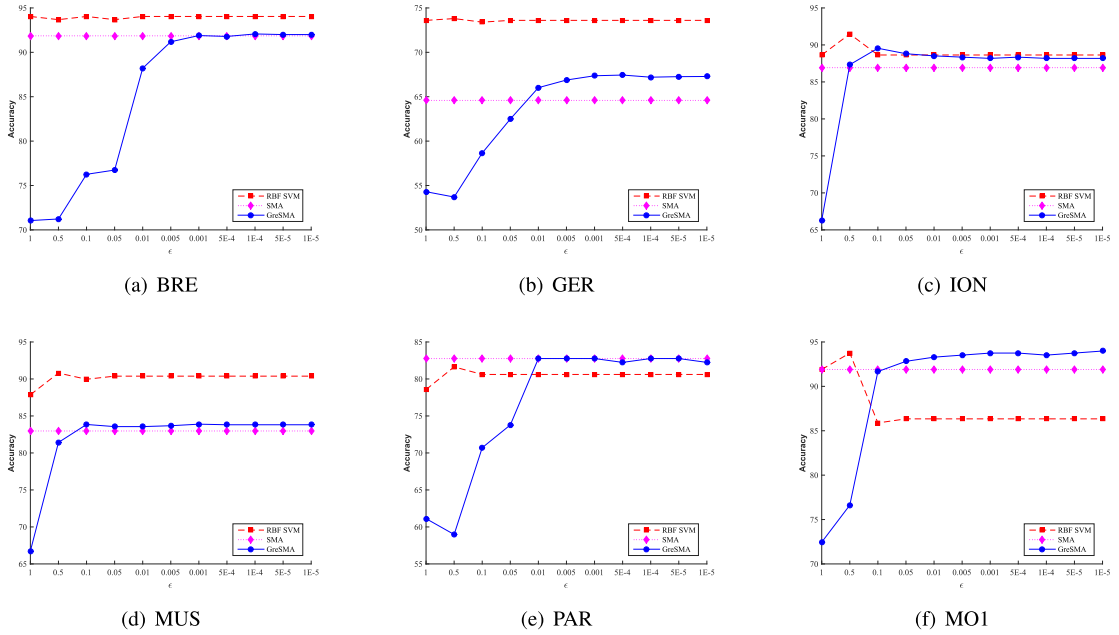


(b) Number of linear functions

FIGURE 11. Comparison of the numbers of conlitrons and linear functions in logarithmic scale.

usually able to bring about the improvement of classification accuracy, which has been confirmed in Table 3. This meets the criteria of "Occam's razor", i.e., the simpler model will fit the data better. More importantly, it is very convenient to integrate a multiconlitron with minimal model into real-time systems and portable devices.

Table 5 reports the training time and testing time. In general, GreSMA takes more training time than SMA because it costs a lot of time to retrain the classification boundary. As for testing time, GreSMA takes less than SMA due to its simpler model structure. For instance, on dataset MAG, it takes GreSMA about 520ms, but it takes SMA about 3282ms. Compared with KDT, GreSMA has a slower training phase



(a) BRE (b) GER (c) ION

(d) MUS (e) PAR (f) MO1

FIGURE 12. Variation of classification accuracies with the parameter ε.

TABLE 5. Comparison of training time in seconds (testing time in milliseconds).

Data	GreSMA	SMA	KDT	Linear SVM	RBF SVM
BRE	0.51(0.50)	0.04(1.41)	0.03(6.39)	4.21(11.10)	6.80(11.50)
GER	3.72(4.27)	0.32(12.82)	0.26(4.57)	12.48(23.90)	20.13(33.20)
HEA	0.08(0.15)	0.02(0.67)	0.01(0.84)	2.81(5.00)	3.08(7.40)
ION	1.13(0.44)	0.05(1.21)	0.04(3.89)	2.31(8.50)	3.65(10.50)
MAG	11221.60	162.82	28.97	2558.48	5947.48
	(520.20)	(3282.10)	(78.60)	(1297.70)	(2456.00)
MUS	7.68(3.05)	0.72(13.64)	0.48(18.91)	12.47(57.00)	17.54(79.00)
PAR	0.03(0.11)	0.01(0.30)	0.01(1.79)	0.61(4.70)	1.04(4.90)
PIM	0.81(1.23)	0.12(3.84)	0.01(1.67)	3.82(8.50)	10.49(12.50)
SON	0.61(0.34)	0.04(1.45)	0.03(4.38)	1.60(9.10)	3.10(10.60)
MO1	0.11(0.14)	0.02(0.31)	0.01(1.61)	4.41(5.00)	4.40(6.00)
MO2	0.27(0.31)	0.03(0.78)	0.01(1.77)	3.11(5.00)	3.18(7.00)
MO3	0.11(0.16)	0.02(0.31)	0.01(1.93)	3.11(6.00)	3.04(5.00)

but a faster testing phase. Compared with SVMs, GreSMA takes less training time, where the training time for SVMs includes in part the time of choosing optimal parameters. On testing time, GreSMA takes less than SVMs, since it need not reconstruct the prediction model by using the whole support vectors.

C. DETERMINATION OF THE PARAMETER

Here, we discuss the determination of the parameter ε for GreSMA. ε indicates the stop criterion both for (Gre)SMA and for SVMs. It should be noted that SMA is hardly affected by the parameter ε. At each piece, it calculates a hyperplane between two original subsets. And for GreSMA, it is sensitive to the parameter ε because it requires the calculation of each separating hyperplane between the convex hulls corresponding to two local subsets. Hence it is significant to choose an appropriate value of ε for GreSMA. Fig.12 shows the classification accuracies of GreSMA, SMA, RBF SVM varying with the parameter ε, where ε takes value from {1, 0.5, 0.1, 0.05, 0.01,..., 0.00005, 0.00001}. From Fig. 12, we can see a common trend on all the six selected datasets, that is, the accuracies gradually increase as ε becomes smaller, and then arrives at a stable value. For most of these datasets, the best ε with desirable accuracies is in [0.005, 0.00001]. Meanwhile, the training time should also be taken into account. Therefore, we would recommend that [0.005, 0.001] is a candidate interval of ε for GreSMA.

VI. CONCLUSION

In this paper, we have proposed a greedy method, i.e., GreSMA for constructing minimal multiconlitron. It is worth noting that we use the term ''minimal'' rather than the term ''minimum'', because the GreSMA is heuristic. And from another point of view, constructing a minimum PLC may be an NP-hard problem. GreSMA includes two core procedures, namely greedy selection and boundary adjustment. With greedy selection, we can separate the maximum number of training samples under the current iteration. With boundary adjustment, we can involve all of the samples separated in the retraining phase for further improving the generalization ability. Experimental results have proved that, compared with the original SMA, GreSMA produces minimal prediction model, and takes less testing time. Moreover, in terms of accuracy, it has a clear superiority over SMA. This is in line with the principle of Occam's razor that aims to choose the simplest assumption that fits the data well. Although it

is not as good as the RBF SVM as a whole, GreSMA still has some advantages that need to be noticed. (1) It does not contain parameters that depend on the datasets. (2) It makes no assumptions of the underlying statistical distributions of the samples. These advantages are important for integrating a PLC into a small device or portable terminal.

It should be noticed that the proposed method has some weaknesses and limitations. GreSMA can only handle binary classification problem at present. However, most real-world scenarios correspond to multi-class classification problems. Therefore, we are ready to introduce some well-known multi-class technologies (such as one-versus-one [29], one-versus-all [30], and binary tree architecture [31]) to extend GreSMA in the future. Also, GreSMA has a lengthy training process. In the current situation, it is not suitable for large-scale problems such as computer vision. As future work, we plan to introduce a more effective learning method (such as surrogate modeling [28]) to further improve the multiconlitron, so that it can be prioritized in some important real-world applications.

REFERENCES

[1] H. Tenmoto, M. Kudo, and M. Shimbo, "Piecewise linear classifiers with an appropriate number of hyperplanes," *Pattern Recognit.*, vol. 31, no. 11, pp. 1627–1634, Nov. 1998.

[2] A. Kostin, "A simple and fast multi-class piecewise linear pattern classifier," *Pattern Recognit.*, vol. 39, no. 11, pp. 1949–1962, Nov. 2006.

[3] X. Huang, S. Mehrkanoon, and J. A. K. Suykens, "Support vector machines with piecewise linear feature mapping," *Neurocomputing*, vol. 117, pp. 118–127, Oct. 2013.

[4] B.-B. Chai, T. Huang, X. Zhuang, Y. Zhao, and J. Sklansky, "Piecewise linear classifiers using binary tree structure and genetic algorithm," *Pattern Recognit.*, vol. 29, no. 11, pp. 1905–1917, Nov. 1996.

[5] D. Wang, X. Zhang, M. Fan, and X. Ye, "Hierarchical mixing linear support vector machines for nonlinear classification," *Pattern Recognit.*, vol. 59, pp. 255–267, Nov. 2016.

[6] H. Ozkan, N. D. Vanli, and S. S. Kozat, "Online classification via self-organizing space partitioning," *IEEE Trans. Signal Process.*, vol. 64, no. 15, pp. 3895–3908, Aug. 2016.

[7] T. B. Johnson and C. Guestrin, "Unified methods for exploiting piecewise linear structure in convex optimization," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, Barcelona, Spain, Dec. 2016, pp. 4761–4769.

[8] O. Mangasarian, "Multisurface method of pattern separation," *IEEE Trans. Inf. Theory*, vol. IT-14, no. 6, pp. 801–807, Nov. 1968.

[9] G. T. Herman and K. T. D. Yeung, "On piecewise-linear classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 7, pp. 782–786, Jul. 1992.

[10] A. M. Bagirov, "Max–min separability," *Optim. Methods Softw.*, vol. 20, nos. 2–3, pp. 277–296, Feb. 2005.

[11] A. M. Bagirov, J. Ugon, and D. Webb, "An efficient algorithm for the incremental construction of a piecewise linear classifier," *Inf. Syst.*, vol. 36, no. 4, pp. 782–790, Jun. 2011.

[12] A. M. Bagirov, J. Ugon, D. Webb, and B. Karasözen, "Classification through incremental max–min separability," *Pattern Anal. Appl.*, vol. 14, no. 2, pp. 165–174, May 2011.

[13] G. Ozturk, A. M. Bagirov, and R. Kasimbeyli, "An incremental piecewise linear classifier based on polyhedral conic separation," *Mach. Learn.*, vol. 101, nos. 1–3, pp. 397–413, Oct. 2015.

[14] J. Sklansky and L. Michelotti, "Locally trained piecewise linear classifiers," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-2, no. 2, pp. 101–111, Mar. 1980.

[15] Y. Park and J. Sklansky, "Automated design of multiple-class piecewise linear classifiers," *J. Classification*, vol. 6, pp. 195–222, Dec. 1989.

[16] K. Gai and C. Zhang, "Learning discriminative piecewise linear models with boundary points," in *Proc. 24th AAAI Conf. Artif. Intell.*, Atlanta, GA, USA, Jul. 2010, pp. 444–450.

[17] Y. Li, B. Liu, X. Yang, Y. Fu, and H. Li, "Multiconlitron: A general piecewise linear classifier," *IEEE Trans. Neural Netw.*, vol. 22, no. 2, pp. 276–289, Feb. 2011.

[18] Y. Li and Q. Leng, "Alternating multiconlitron: A novel framework for piecewise linear classification," *Pattern Recognit.*, vol. 48, no. 3, pp. 968–975, 2015.

[19] W. Li and J. Hu, "Geometric approach of quasi-linear kernel composition for support vector machine," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Killarney, Ireland, Jul. 2015, pp. 1–7.

[20] O. Pujol and D. Masip, "Geometry-based ensembles: Toward a structural characterization of the classification boundary," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 6, pp. 1140–1146, Jun. 2009.

[21] K. P. Bennett and E. J. Bredensteiner, "Duality and geometry in SVM classifiers," in *Proc. 17th Int. Conf. Mach. Learn.*, San Francisco, CA, USA, Jul. 2000, pp. 57–64.

[22] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. Murthy, "A fast iterative nearest point algorithm for support vector machine classifier design," *IEEE Trans. Neural Netw.*, vol. 11, no. 1, pp. 124–136, Jan. 2000.

[23] V. Franc and V. Hlaváč, "An iterative algorithm learning the maximal margin classifier," *Pattern Recognit.*, vol. 36, no. 9, pp. 1985–1996, Sep. 2003.

[24] F. Abdallah, C. Richard, and R. Lengelle, "An improved training algorithm for nonlinear kernel discriminants," *IEEE Trans. Signal Process.*, vol. 52, no. 10, pp. 2798–2806, Oct. 2004.

[25] M. E. Mavroforakis and S. Theodoridis, "A geometric approach to support vector machine (SVM) classification," *IEEE Trans. Neural Netw.*, vol. 17, no. 3, pp. 671–682, May 2006.

[26] A. Frank and A. Asuncion. (2010). *UCI Machine Learning Repository*. [Online]. Available: http://archive.ics.uci.edu/ml

[27] C. C. Chang and C. J. Lin. (2001). *LIBSVM: A Library for Support Vector Machines*. [Online]. Available: http://www.csie.ntu.edu.tw/cjlin/libsvm

[28] W. Li, M. Xiao, X. Peng, A. Garg, and L. Gao, "A surrogate thermal modeling and parametric optimization of battery pack with air cooling for EVs," *Appl. Therm. Eng.*, vol. 147, pp. 90–100, Jan. 2019.

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

[30] A. C. Lorena, A. C. P. L. F. de Carvalho, and J. M. P. Gama, "A review on the combination of binary classifiers in multiclass problems," *Artif. Intell. Rev.*, vol. 30, nos. 1–4, pp. 19–37, Jan. 2008.

[31] B. Fei and J. Liu, "Binary tree of SVM: A new fast multiclass training and classification algorithm," *IEEE Trans. Neural Netw.*, vol. 17, no. 3, pp. 696–704, May 2006.

Q. LENG was born in Chaoyang, China, in December 1981. He received the B.S. and M.S. degrees in computer science and technology from Bohai University, Jinzhou, China, in 2004 and 2011, respectively, and the Ph.D. degree in computer science and technology from the Beijing University of Technology, Beijing, China, in 2015.

He was an Assistant Professor with the College of Information Science and Technology, Bohai University, from 2015 to 2017, where he has been an Associate Professor, since January 2018. He has published several articles in international journals, such as *Information Sciences*, *Applied Soft Computing*, *Pattern Recognition*, *Knowledge-Based systems*. His research interests include pattern recognition and machine learning.

Y. LIU is currently pursuing the M.S. degree with the College of Information Science and Technology, Bohai University, Jinzhou, China. His current research interests include pattern recognition and machine learning.

X. ZHAO received the B.S. degree in mathematics from Liaoning University, Shenyang, China, in 1995. From 2006 to 2019, he was an Associate Professor of software engineering with Bohai University, China. He is currently the Director of the Key Laboratory of Technology and Standard of Big Data Mining, Governance, and Presentation in digital publishing. His research interests include management information systems, big data, and blend learning.

L. ZHANG received the B.S. degree from Bohai University, in 2000. He received the M.S. and Ph.D. degrees from the Beijing University of Posts and Telecommunications, in 2008 and 2011, respectively. Since January 2019, he has been a Professor with the Bohai University. His research interests include service computing, cloud computing, and machine learning.

Y. QIN received the B.S. degree from the Jinzhou Teacher's College, in 1989, the master's and Ph.D. degrees from the Dalian University of Technology, in 1996 and 2008, respectively. Since June 2005, he has been a Professor with Bohai University. He is the author of three books and more than 40 articles. His current research interests include pattern recognition and machine learning.