Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2020) 28: 2735 – 2749
© TÜBİTAK
doi:10.3906/elk-2001-62

Revised polyhedral conic functions algorithm for supervised classification

Gürhan CEYLAN[1,2,*], **Gürkan ÖZTÜRK**[1,2]
[1]Department of Industrial Engineering, Faculty of Engineering, Eskişehir Technical University, Eskişehir, Turkey
[2]Computational Intelligence and Optimization Laboratory, Faculty of Engineering, Eskişehir Technical University, Eskişehir, Turkey

Received: 13.01.2020 • **Accepted/Published Online:** 04.05.2020 • **Final Version:** 25.09.2020

Abstract: In supervised classification, obtaining nonlinear separating functions from an algorithm is crucial for prediction accuracy. This paper analyzes the polyhedral conic functions (PCF) algorithm that generates nonlinear separating functions by only solving simple subproblems. Then, a revised version of the algorithm is developed that achieves better generalization and fast training while maintaining the simplicity and high prediction accuracy of the original PCF algorithm. This is accomplished by making the following modifications to the subproblem: extension of the objective function with a regularization term, relaxation of a hard constraint set and introduction of a new error term. Experimental results show that the modifications provide %12 better generalization on average and up to 10x faster training. This paper also contributes to the literature by providing detailed comparisons of the other classification algorithms that use polyhedral conic functions for the first time.

Key words: Classification, conic functions, machine learning, optimization

1. Introduction

Supervised classification is one of the most important tasks in machine learning/data mining, which aims to find a separating function that maps the input data to a set of labels. Many real world problems from different fields can be defined as a supervised classification task such as object detection, cancer diagnosis, protein prediction, document indexing, spam filtering and others. There is a rich literature on the state-of-the-art classification methods rooted in statistical learning, decision and optimization theories. A comprehensive survey can be found in [1–3]. The methods that use optimization theory reduce the supervised classification problem to a core mathematical programming model and then optimize the decision variables or parameters of the model to obtain a separating function [4]. Most methods reported in the literature utilize a convex mathematical programming model because such models are relatively easy to solve compared to the nonconvex models [1, 5, 6]. However, for large data sets, obtaining an optimal solution becomes challenging even for the convex models. Consequently, there is an ongoing effort to develop effective methods by making changes to the core mathematical programming model [4].

This paper focuses on one of the mathematical programming-based methods introduced by Gasimov and Ozturk [7]. In their study, a special class of the polyhedral conic functions (PCF) was defined, and an iterative method named the PCF algorithm was proposed for binary classification problems. This algorithm is

[*]Correspondence: grhanceylan@eskisehir.edu.tr

This work is licensed under a Creative Commons Attribution 4.0 International License.

remarkably advantageous for linearly nonseparable data sets, which is the case for most real world data sets. The algorithm separates such data sets by solving a finite number of linear programming (LP) problems. Separation is performed without mapping the data points in a higher-dimension space, unlike in the kernel methods [8–10], and without the use of complicated structures. Despite its success for the generation of nonlinear separating functions, the PCF algorithm has been reported to have two main drawbacks: overfitting/low generalization and dependence of the performance on the parameters called centers [7, 11, 12]

Overfitting, a well-known challenge in supervised classification, occurs when a method generates a complex separating function rather than a simple one that adequately explains a data set. It is undesirable for several reasons: a predictor that is not useful for test data wastes resources, unnecessary predictors often lead to worse decisions, and complicated models are difficult to apply [13, 14]. On the other hand, the dependence of the performance on the center parameters is specific to the PCF algorithm. The algorithm generates a series of conic functions that separate a given data set by solving sequential subproblems. In each iteration, the solved subproblem takes a randomly chosen center as an input. These inputs affect both the generated conic functions and the total number of iterations until the algorithm is terminated. Consequently, the resulting separation function and training performance of the algorithm depend on the given centers. Using illustrative examples, this drawback will be explained in detail in Section 3.

In this study, our main goal is to develop a new version of the PCF algorithm that does not suffer from the abovementioned drawbacks. Since the algorithm was proved to be able to generate nonlinear separating functions in [7], we expect to overcome the drawbacks by making appropriate modifications to the algorithm. More recent studies that report competitive results by utilizing PCF provide additional support for our approach [15–20].

Several studies with similar scope have been reported. In [12], an algorithm named k-means PCF was proposed to overcome the abovementioned drawbacks. The key idea of this approach is to use the well-known k-means clustering algorithm to find centers that give better performance instead of randomly choosing the centers. The reported results show that this approach helps achieve better performance metrics. However, the algorithm inherits the problem of finding the best number of clusters, or the k value, from the clustering algorithm. In a follow-up study [11], the incremental conic functions (ICF) algorithm was proposed, which automatically determines k. Although the value of k is found by the algorithm, it gives worse prediction results than k-means PCF. In [21], the incremental conic separation (IPCS) algorithm was introduced with a nonsmooth-nonconvex mathematical programming model. The algorithm defines the center parameter as a decision variable and finds its optimal value by solving the model. Therefore, the performance of this algorithm does not depend on the parameter. However, IPCS requires the solution of a hard optimization problem by comparison to LP using a dedicated solving procedure.

To achieve our goal, we analyzed the PCF algorithm and then, based on the analysis, proposed the following changes to the mathematical programming model: extension of the objective function with a regularization term, constraint relaxation, introduction of a new error term, and use of penalizing and regularization parameters to control the complexity of the generated separating function. These changes lead to a new model formulated as a quadratic mathematical programming (QP) problem. Based on this model, we propose a new version of the PCF algorithm with better performance metrics. We report the results of detailed experiments on publicly available benchmark data sets and compare our method with both the methods used in [7, 11, 12, 21] and the state-of-the-art methods.

The rest of the paper is organized as follows. In Section 2, preliminaries about PCF are provided, and the PCF algorithm is explained in detail. The algorithm is analyzed to reveal the origins of the abovementioned drawbacks, and a series of changes are proposed in Section 3. The revised version of the algorithm is presented in Section 4. Detailed computational experiments and comparisons are reported in Section 5. Finally, Section 6 contains some conclusions and remarks about future work.

2. Polyhedral conic functions algorithm

This section gives preliminaries of PCF and conic separation and then explains the PCF algorithm in detail. Rigorous proofs and discussions of conic separation can be found in [7, 22, 23].

In the rest of the paper, the scalar product of two vectors $x, y \in R^n$ will be denoted by $x'y$, and the l_p norm of a vector x will be denoted by $\|x\|_p$. Data sets belonging to classes -1 and $+1$ will be denoted by $A = \{a_i \in \mathbb{R}^n : i \in I\}$ and $B = \{b_j \in \mathbb{R}^n : j \in J\}$, respectively, where $I = \{1, \ldots, m\}$ and $J = \{1, \ldots, p\}$ represent index sets.

Definition 1 *[7] A function $g : \mathbb{R}^n \to \mathbb{R}$ is called polyhedral conic if its graph is a cone and all of its sublevel sets $S(\alpha) = \{x \in \mathbb{R}^n : g(x) \leq \alpha\}$ for $\alpha \in \mathbb{R}$ are polyhedrons.*

Based on the above definition, a special class of polyhedral conic functions is introduced in[7] as $g : \mathbb{R}^n \to \mathbb{R}$,

$$g_{(w,\xi,\gamma,a)}(x) = w'(x-a) + \xi\|x-a\|_1 - \gamma, \tag{1}$$

where $w,\ a \in \mathbb{R}^n, \xi, \gamma \in \mathbb{R}$.

A function defined in the form of (1) is polyhedral conic since its graph is a polyhedral cone with vertex $(a, -\gamma) \in \mathbb{R}^n \times \mathbb{R}$ and all of its sublevel sets are polyhedrons [7]. The parameter named the center is the first component $(a \in \mathbb{R}^n)$ of the vertex of a PCF. The role of a center is illustrated in Figure 1, which shows the graphs and sublevel sets of the following functions:

$$g^1_{(0.3,0.9,1.0,2.0)}(x) = 0.3(x-2.0) + 0.9|x-2.0| - 1.0,$$
$$g^2_{(0.3,0.9,1.0,6.0)}(x) = 0.3(x-6.0) + 0.9|x-6.0| - 1.0,$$

for which the centers are given as $a_1 = 2.0$ and $a_2 = 6.0$ in $g^1(x)$ and $g^2(x)$, respectively.

In [7], data sets A and B are said to be polyhedral conic separable if there exist a finite number of PCF, $g_k(x) = g_{(w_k,\xi_k,\gamma_k,a_k)}(x)$, $k = 1, \ldots, K$, such that

$$\min\{g_1(a_1), \ldots, g_K(a_i)\} \leq 0, \quad i = 1, \ldots, m \tag{2}$$

and

$$\min\{g_1(b_1), \ldots, g_K(b_j)\} > 0, \quad j = 1, \ldots, p. \tag{3}$$

In the training phase, the PCF algorithm generates a series of functions for which their point-wise minimum separates given data sets A and B as defined in (2–3). In each k^{th} iteration, the algorithm chooses a random point $(a_k \in A)$ as the center and solves the following subproblem:

$$(P_k) \quad \min_{w,\xi,\gamma} \quad \frac{1}{m}\sum_{i=1}^{m} y_i \tag{4a}$$

$$\text{s.t.} \quad w'(a_i - a_k) + \xi\|a_i - a_k\|_1 - \gamma + 1 \le y_i, \quad \forall i \in I_k \tag{4b}$$

$$-w'(b_j - a_k) - \xi\|b_j - a_k\|_1 + \gamma + 1 \le 0, \quad \forall j \in J \tag{4c}$$

$$\xi \ge 0, \quad \gamma \ge 1, \quad y_i \ge 0 \quad \forall i \in I_k. \tag{4d}$$

The subproblem tries to find a PCF that satisfies $g_k(a_i) \le -1$ for as many points as possible from set A while satisfying $g_k(b_j) \ge 1$ for all of the points from set B. The first part is embedded in the problem by soft constraints (4b) on set A with the objective function (4a). The latter part is achieved by hard constraints (4c) on set B. The lower bounds on the decision variables ξ, γ ensure convex polyhedral level sets.

Then, the algorithm generates $g_k(x) = g_{(w_k,\xi_k,\gamma_k,a_k)}(x)$ by using the solution of (P_k) and proceeds to the next iteration after removing the correctly classified points from set A. The algorithm stores all of the generated functions and terminates when A becomes an empty set. As an output, it returns a list of the generated functions for which the point-wise minimum $g(x) = \min\{g_1(x), \dots, g_k(x)\}$ separates the given data sets. The pseudocode of the algorithm is given in Algorithm 1.

Algorithm 1 PCF Algorithm

```
Input: Data sets: $A = \{a_i \in \mathbb{R}^n : i \in I\}$, $I = \{1, \dots, m\}$ and $B = \{b_j \in \mathbb{R}^n : j \in J\}$, $J = \{1, \dots, p\}$
Output: Return $g(x) = \min\{g_1(x), \dots, g_k(x)\}$
1: Initialization Set $k = 1, I_k = I$
2: while $I_k \neq \emptyset$ do
3:     Select a random index $(r)$ from $I_k$, and set $a_k = a_r$
4:     Compute $w_k, \xi_k, \gamma_k$ by solving subproblem $(P_k)$
5:     Construct and store conic function $g_k = g_{(w_k,\xi_k,\gamma_k,a_k)}(x)$
6:     Update the index set $I_{k+1} = \{i \in I_k : g_k(a_i) + 1 > 0\}$, and set $k = k + 1$
7: end while
```

In the test phase, the algorithm predicts the label of a point by using $g(x)$, predicting -1 if $g(x) \le 0$ and $+1$ otherwise. Figure 2 shows a solution obtained by the PCF algorithm on a linearly nonseparable data set, where the circle and square data points represent sets A and B, respectively. The shaded line is the resulting separating function, where $g(x) \le 0$ for all of the circle points and $g(x) > 0$ for all of the square points.

Theorem 2.3 in [7] proves that the algorithm terminates in a finite number of iterations. It is shown that the functions generated in each iteration satisfy $g_k(x) \le -1$ at least for the point (a_k) chosen as the center. Since the algorithm terminates when set A becomes empty and set A has a finite number of elements, in the worst case, the algorithm ends in $m = s(A)$ iterations by solving m subproblems. However, it does not generate a separating function with good generalization ability, and the number of solved subproblems depends on the centers selected in the iterations.

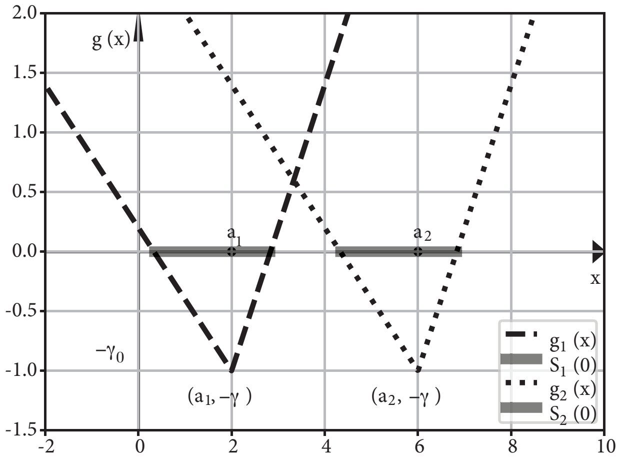

Figure 1. Graphs and sublevel sets of polyhedral conic functions.

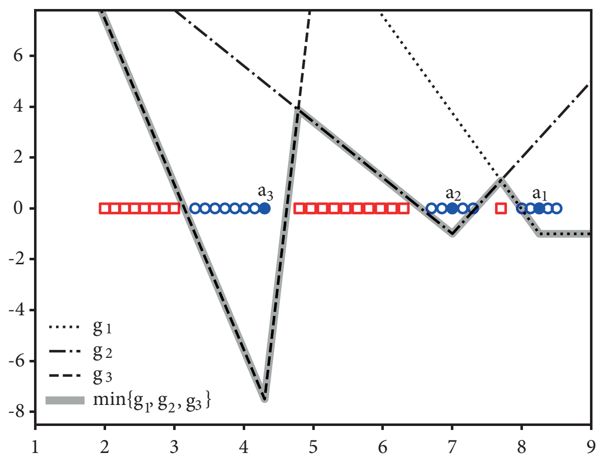

Figure 2. A solution of the PCF algorithm, where points a_1, a_2, a_3 are the selected centers.

3. Analysis of the PCF algorithm

In this section, we analyze the PCF algorithm with respect to overfitting and the center-dependent performance. To handle these drawbacks, we also propose three changes to the algorithm that are explained as follows.

The algorithm suffers from overfitting for two main reasons. The first is a well-known issue in the field of supervised classification, namely, a separating function obtained by minimizing only the training error may not perform well with unknown data [24]. This issue occurs because a complex separating function that minimizes the training error does not guarantee a small test error. To deal with this issue, a well known technique is minimizing a combination of the training error and a regularization term, which is chosen as a norm of the separating function [24, 25]. This technique handles overfitting because it minimizes the difference between the training and test errors by restricting the complexity of the separating function [24, 26].

Change 1: Extension of the objective function with the regularization term

$$\min_{\bar{w}} \quad \|\bar{w}\|_2^2 + \frac{1}{m}\sum_{i=1}^{m} y_i, \quad \bar{w} := \big(w\;\xi\;\gamma\big) \in \mathbb{R}^{n+2}.$$

We adapt the regularization technique as shown in Change 1 because the PCF algorithm solves the subproblem (P_k) with the objective function (4a), which only minimizes the training error. We choose the square of the l_2 norm for complexity approximation to exploit its favorable properties of differentiability and convexity.

The other reason for the overfitting drawback of the PCF algorithm is related to the hard constraint set (4c) of the subproblem. In each iteration, (4c) does not allow any misclassified points from set B, and the algorithm terminates when all of the points from set A are correctly classified. Therefore, the algorithm finally obtains a separating function where the training accuracy is 100%. This result immediately shows that in and of itself, Change 1 is not sufficient for handling overfitting.

Change 2: Allowing misclassification from set B

$$-w'(b_j - a_k) - \xi \|b_j - a_k\|_1 + \gamma + 1 \leq z_j, \quad \forall j \in J.$$

To minimize the gap between the training and test errors, we propose relaxing the constraint set (4c) by adding nonnegative slack variables (z_j). By allowing the misclassification from set B, we change the separation procedures. Since it is desired that (z_j) be positive when this is necessary, the objective function is extended with sum of the slack variables as follows.

$$\min_{\bar{w}} \quad \|\bar{w}\|_2^2 + \frac{1}{m}\sum_{i=1}^{m} y_i + \frac{1}{p}\sum_{j=1}^{p} z_j.$$

We now explain the drawback of the center-dependent performance with an illustrative example. Figure 3 shows two different solutions obtained by the PCF algorithm on a toy data set with predetermined centers. By comparing the given solutions in Figure 3, it is easy to see that the algorithm separates the same data set by solving 3 and 4 subproblems, respectively. This difference occurs because the predetermined centers are different. Since an increase in the number of solved subproblems also increases the required training time, the performance of the algorithm depends on the given centers. This example also shows that the complexity of the resulting separating function is also affected by the center. It is observed that the separating function in Figure 3a is simpler relative to that in Figure 3b.

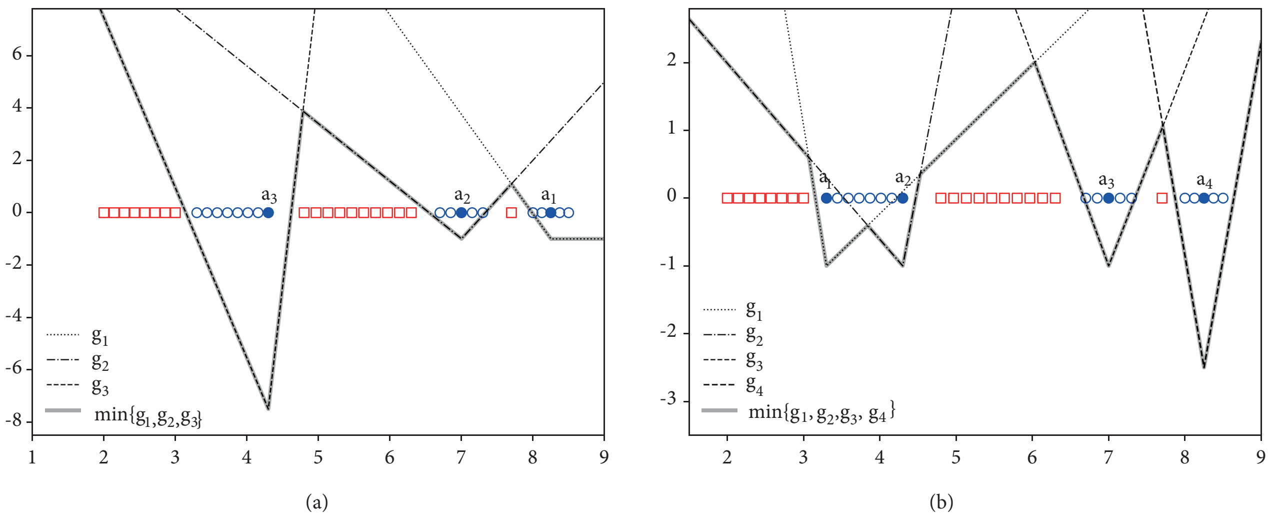

Figure 3. Solutions obtained by the PCF algorithm, where the filled in points represent the centers.

The first approach for overcoming this issue is to find the centers that give better performance instead of randomly choosing them. In [12], the k-means PCF algorithm utilizes this idea by using a clustering algorithm in a preprocessing step. Then, the algorithm solves a specific subproblem that is modeled as LP for each cluster by using their centers. Although clustering reduces the amount of solved subproblems, it is reported that the generalization ability and prediction accuracy of k-means PCF depends on the number of clusters. The other idea of finding the best centers is to define the parameter as a decision variable in the mathematical model. In this case, we end up with an algorithm that must repeatedly solve a nonlinearly constrained subproblem, which is hard to solve[21].

Change 3: Penalizing the complexity of the separating function

$$\min_{\bar{w}} \quad \lambda\|\bar{w}\|_2^2 + \frac{1}{m}\sum_{i=1}^{m} y_i + \frac{1}{p}\sum_{j=1}^{p} z_j$$

Unlike in the previous studies [11, 12, 21], we consider the drawback as generating best performing PCF instead of finding the best performing centers. Based on this point of view, we introduce a positive scalar $\lambda > 0$, known as the regularization parameter, into the objective function. This very simple change enables control of the complexity of the generated PCF, while maintaining the convexity of the model. The effect of this scalar on the algorithm will be explained with demonstrative experiments in the next section.

4. Proposed method: revised PCF(r-PCF) algorithm

This section first collects all of the proposed changes, gives the obtained mathematical model, and then explains the general structure of the revised PCF algorithm. The changes lead us to the following linearly constrained quadratic convex subproblem:

$$
\begin{aligned}
(Q_k) \quad \min_{\bar{w}} \quad & \lambda\|\bar{w}\|_2^2 + \sum_{i=1}^{m} y_i + C\sum_{j=1}^{p} z_j && \text{(5a)}\\
\text{s.t.} \quad & w'(a_i - a_k) + \xi\|a_i - a_k\|_1 - \gamma + 1 \leq y_i \quad \forall i \in I_k, && \text{(5b)}\\
& -w'(b_j - a_k) - \xi\|b_j - a_k\|_1 + \gamma + 1 \leq z_j \quad \forall j \in J_k, && \text{(5c)}\\
& y_i \geq 0 \quad \forall i \in I_k, \quad z_j \geq 0 \quad \forall i \in J_k, && \text{(5d)}\\
& \xi \geq 0, \quad \gamma \geq 1, && \text{(5e)}
\end{aligned}
$$

where $\bar{w} := \begin{pmatrix} w \, \xi \, \gamma \end{pmatrix}, C, \lambda > 0$. There is a change in the objective function of the subproblem related to weights $1/m$ and $1/p$ that was not mentioned in the previous section, and this change will be explained after describing the general flow of the algorithm.

In each iteration, the r-PCF algorithm takes the C and λ parameters as input and then generates $g_k(x) = g_{(w_k, \xi_k, \gamma_k, a_k)}(x)$ by solving (Q_k) with randomly chosen center a_k. Then, the points that are correctly classified are removed from set A, while the misclassified points are removed from set B. We also update set B because once a point satisfies $g_k(b_j) \leq 0$, it is ensured that the point b_j will be misclassified. This fact can be easily checked by using the definition of conic separation given in (2–3). The algorithm terminates when one of the given sets, A or B, becomes an empty set.

As a result of the training phase, the algorithm returns a list of the generated functions for which the point-wise minimum $g(x) = \min\{g_1(x), \ldots, g_k(x)\}$ separates the given data sets. In the test phase, the r-PCF algorithm uses the resulting $g(x)$ and predicts the label of a point x as -1 if $g(x) \leq 0$ and as $+1$ otherwise. The general flow of the proposed algorithm is also explained in Algorithm 2.

In the subproblem, parameter C only penalizes the error arising from set B because the misclassification error related to set A depends on the conic function generated in the iteration. Let us assume that we are given data sets $A = A_1 \cup A_2$ and B, which are conic separable with at least two PCF such that $A_1 = \{a_i : g_1(a_i) \leq 0, g_2(a_i) > 0, i = 1, \ldots, l\}$, $A_2 = \{a_i : g_1(a_i) > 0, g_2(a_i) \leq 0, i = l+1, \ldots, m\}$ and

Algorithm 2 r-PCF Algorithm

```
Input: Data sets: $A = \{a_i \in \mathbb{R}^n : i \in I\}$, $I = \{1, \dots, m\}$, $B = \{b_j \in \mathbb{R}^n : j \in J\}$, $J = \{1, \dots, p\}$
    Parameters: $C, \lambda$
Output: Return $g(x) = \min\{g_1(x), \dots, g_k(x)\}$
1: Initialization Set $k = 1, I_k = I, J_k = J$
2: while $I_k \neq \emptyset$ or $J_k \neq \emptyset$ do
3:     Select a random index($r$) from $I_k$, and set $a_k = a_r$
4:     Compute $w_k, \xi_k, \gamma_k$ by solving subproblem $(Q_k)$
5:     Construct and store conic function $g_k(x) = g_{(w_k,\xi_k,\gamma_k,a_k)}(x)$
6:     Update the index sets $I_{k+1} = \{i \in I_k : g_k(a_i) > 0\}$, $J_{k+1} = \{j \in J_k : g_k(b_j) > 0\}$, and set $k = k + 1$
7: end while
```

$B = \{b_j : g_1(b_j) > 0, g_2(b_j) > 0, j = 1, \dots, p\}$. Then, assume that g_1 and g_2 are the generated PCF in the first and second iterations, respectively. It is clear that g_1 misclassifies points in A_2, and an error occurs in the first iteration. However, this error is not the resulting training error because g_2 correctly classifies these points in the second iteration. Therefore, we only control the relative training error of set B with respect to the error of set A by taking $1/m = 1$ and $C = 1/p$.

We claim that the proposed changes can address the performance dependence on the centers. To support our claim, we show two solutions of our algorithm in Figure 4. The solutions are obtained using the same toy data set and centers as those used in Figure 3. In particular, the same centers are used in the exact same order in Figures3a–4a and 3b–4b. It is observed that the r-PCF algorithm solves only 2 subproblems in both cases, whereas the PCF algorithm solves 3 and 4 subproblems, respectively. This result indicates that with the proper C and λ values, the proposed changes achieve robustness in terms of solved subproblems compared to the randomly chosen centers.

We now describe the results of a case study on the well-known Ionosphere data set, shown in Figure 5, to demonstrate the effects of the C and λ parameters. The results in Figure 5a are obtained by 10-fold cross validation for each (C, λ) value, where $\lambda = 1$ and $C \in \{10^i : i = -3, -2, \dots, 2, 3\}$. It is observed that the training accuracy and number of the solved subproblems increase with respect to parameter C. However, the test accuracy stops improving after a threshold $C \approx 10^1$ is reached, leading to overfitting. This occurs because when parameter C dominates the subproblem, (5c) acts as a hard constraint set, and our algorithm suffers from overfitting.

The results in Figure 5b are again obtained by 10-fold cross validation for all of the (C, λ) values, where, $C = 1$ and $\lambda \in \{10^i : i = -3, -2, \dots, 2, 3\}$. In this case, it is observed that the r-PCF algorithm tends to solve fewer subproblems and achieves better generalization when the λ parameter dominates the subproblem (Q_k). This result indicates that as expected, the parameter can control the complexity of the generated PCF. On the other hand, an increase in λ has a negative effect on the prediction accuracy. To obtain a solution with low overfitting and high prediction, it is necessary to find the best values of the C and λ parameters. This can be achieved by using a grid search with a cross-validation technique[27].

5. Computational experiments and comparisons

We report results for publicly available data sets from the UCI Repository [28]. Table 1 provides a summary of the information about the data sets and sampling procedures. For the relatively small data sets, the stratified 10-fold procedure was used, while the other data sets were simply divided into the training and test parts. To obtain

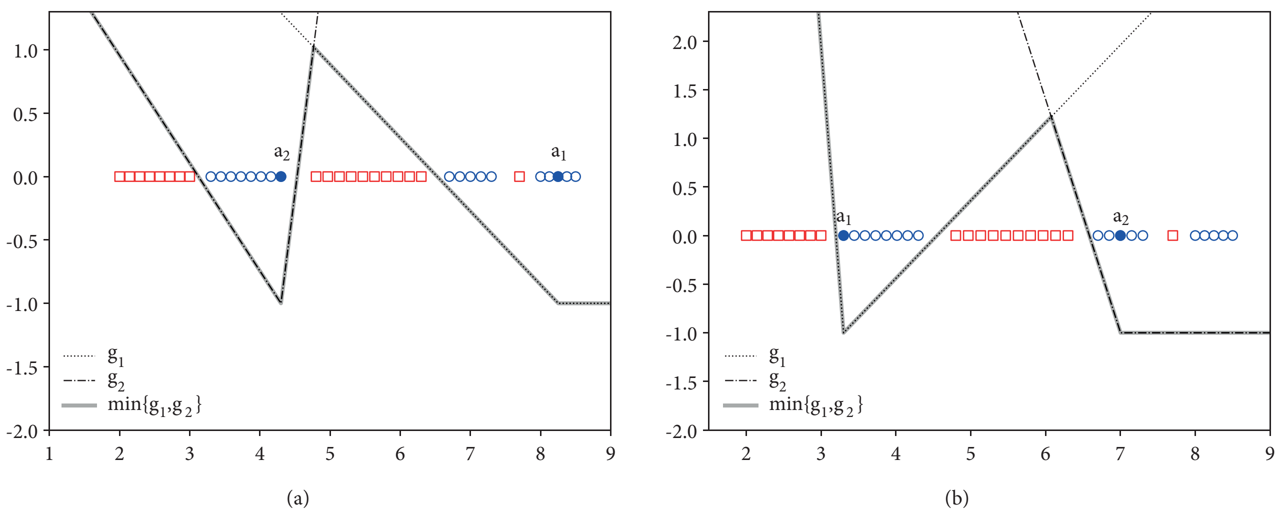

Figure 4. Solution comparison of the r-PCF algorithm, where the filled in points represent the centers.

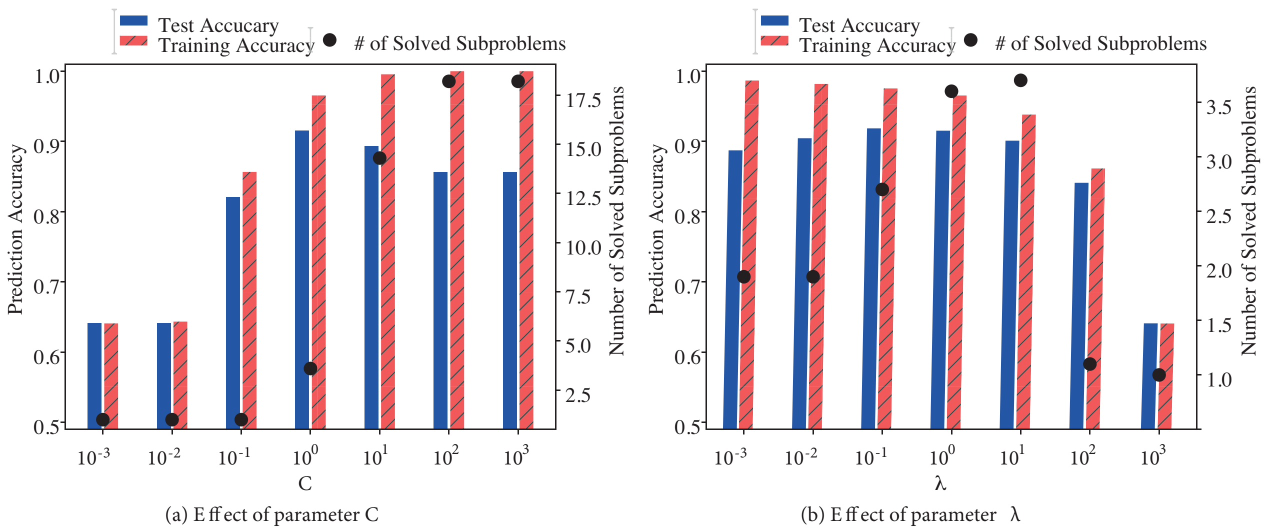

Figure 5. Case analysis on the Ionosphere data set.

best performing C and λ parameters, the grid search technique [27] was used for each training data sets, and the values were chosen from the following sets: $C \in \{10^i | i = -4, -3, \ldots, 3, 4\}$ and $\lambda \in \{10^i | i = -4, -3, \ldots, 3, 4\}$. For multiclass data sets, the one-against-all strategy was used.

In order to make a fare comparison, we implemented PCF, k-means PCF and r-PCF algorithms using Python [29] programming language, Gurobi [30] optimization tool and Scikit-Learn [31] machine learning package. Also, the implementation provided in [11] was used for ICF algorithm.

Table 2 contains the average stratified 10-fold cross-validation performance metrics obtained using the first six data sets. In each case, our algorithm solves fewer subproblems than the PCF algorithm in a shorter time. This is important because the proposed subproblem (Q_k) is generally computationally more expensive than (P_k). For example, for the Ionosphere data set, the PCF and r-PCF algorithms spend ≈ 0.012 and ≈ 0.08 seconds on their subproblems, respectively. Although (Q_k) requires more time, the overall training performance

Table 1. Description of data sets and sampling procedures.

Data sets	# of samples training-test	# of features	# of classes
Wisconsin breast cancer prognostic (WBCP)	194*	32	2
Statlog heart (SH)	270*	13	2
Bupa Liver (BL)	345*	6	2
Ionosphere (ISP)	351*	34	2
Wisconsin breast cancer diagnostic (WBCD)	683*	9	2
Pima Indians diabetes (PID)	768*	8	2
DNA	2000-1186	180	3
Spambase (SB)	3682-919	57	2
Abalone (AB)	3133-1044	8	3
Phoneme-CR (PHON)	4322-1082	5	2
Landsat satellite image (LSI)	4435-2000	36	6
Pen-based recognition of handwritten digits (PD)	7494-3498	16	10
Shuttle control (SHC)	43500-14500	9	7

*: Stratified 10-fold cross validation.

of the r-PCF algorithm is better because it solves fewer subproblems. Furthermore, there is no trade-off between the improvement in the training time and prediction accuracy, and the r-PCF algorithm also gives better test accuracy scores for each data set.

The generalization error, calculated as the difference between the training and test accuracy scores, is a common metric of overfitting. In Table 2, the average generalization errors are %22.2 and %5.2 for the PCF and r-PCF algorithms, respectively. This result indicates that the proposed changes helping deal with overfitting. However, an examination of the individual results obtained using our algorithm shows that there are still considerable gaps between the accuracies, with the maximum value of %9.8 obtained for the Statlog Heart data set. We think that this occurs due to the fixed C and λ values for all of the subproblems and can be addressed by adaptively choosing these values for each subproblem.

Now, we present Tables 3 and 4 to compare our algorithm with the other PCF-based methods for relatively large scale data sets. We also combined the results reported in [21] for the IPCS algorithm and the following state-of-the-art methods: support vector machine (SVM) with polynomial kernel (LibSVM-POL), SVM with radial based kernel (LibSVM-RBF), SVM with normalized polynomial kernel (SMO-NPOL) and (SMO-PUK), naive Bayes with kernel (NB-Kernel) and the J48(C4.5) decision tree algorithm. The comparison is fair since the authors in [21] also used the same sampling procedure and grid search technique to find the best performing parameters when necessary.

The results reported in Table 3 are consistent with the results in Table 2, and it is observed that our algorithm is 10× faster than the PCF algorithm for the LSI data set. However, the algorithms that use clustering k-means PCF and ICF achieve better training performance. This occurs because clustering remarkably decreases the number of solved subproblems, and it is computationally cheaper than solving an excessive number of optimization problems. Additionally, the training performance of the r-PCF algorithm is not comparable to those of the state-of-the-art methods. For example, the training time of LibSVM-RBF is given as 0.71 s for the PD data set in [21], whereas the training time for our algorithm our is 670.3 s.

Table 2. Performance metrics of the PCF and r-PCF algorithms.

	PCF algorithm				r-PCF algorithm			
Data sets	training time	# of solved sp.	training acc.	test acc.	training time	# of solved sp.	training acc.	test acc.
WBCP	0.09	2.6	100	74.82	**0.05**	**2.1**	81.73	**80.00**
SH	1.10	25.8	100	77.03	**0.42**	**16.6**	91.15	**81.48**
BL	2.63	83.1	100	64.61	**0.13**	**10.0**	73.05	**66.30**
ISP	0.30	2.5	100	83.89	**0.08**	**1.0**	96.64	**92.39**
WBCD	0.88	12.4	100	95.62	**0.12**	**1.2**	97.45	**97.08**
PD	13.82	154.3	100	70.83	**0.44**	**22.0**	76.90	**71.37**

Accuracy scores are given as percentages. # of solved sp.: Number of solved subproblems.

Table 3. Training metrics of PCF-based methods.

	Training time - # of solved subproblems						
Classifier/data Set	DNA	SB	AB	PHON	LSI	PD	SH
PCF alg.	12.2 – 4	258.6 – 210	689.9 – 1431	452.3 – 1337	373.5 – 411	984.7 – 582	585.7 – 108
k-means PCF alg.	10.7 - 6	13.3 - 18	2.2 – 6	4.7 – 18	40.8 – 42	90.4 – 90	158.6 – 28
ICF alg.	200.0 – 28	12.4 – 4	14.0 – 27	9.5 – 17	94.9 – 33	164.8 – 54	284.8 – 8
IPCS alg.	37.9 – 4	107.5 – 4	18.9 – 10	48.4 – 10	898.0 – 21	564.0 – 25	2254.6 – 19
r-PCF alg.	6.9 – 6	43.2 – 75	93.86 – 438	108.9 – 360	37.0 – 45	670.3 – 433	376.4 – 83

Training time scores are given in seconds.

The results listed in Table 4 show that our algorithm achieves better generalization performance when more data are used. For these relatively large data sets, the average generalization error of the r-PCF algorithm is 2.1%, where the maximum error of 3.8% occurs for the DNA data set. This kind of generalization improvement is also observed for the PCF algorithm because it gives an 11.0% average generalization error. However, this only appears to be the case due to the increase in the test accuracy. A large generalization error of 35.5% is observed for the Abalone data set.

We can also make the following general comments by taking the test accuracy results of the PCF algorithm as a basis for comparison. Finding the best performing centers via clustering, as used in the k-means PCF and ICF algorithms, decreases the test accuracy. On the other hand, finding the optimal centers by solving a mathematical model, as used in IPCS, and our idea of generating the best performing conic functions for randomly given centers increase the test accuracy. Due to this, our algorithm gives at least the third best test accuracy results among the PCF-based algorithms and the state-of-the-art methods.

On the overall data sets, the r-PCF and PCF algorithms give 3.5% and 16.2% generalization errors by solving 335.7 and 114.8 subproblems in 259.7 and 103.0 seconds, respectively. On average, the r-PCF algorithm achieves 12.7% better generalization and 2.5× faster training than the PCF algorithm. Based on these results, we conclude that the proposed changes helping deal with the problems of overfitting and center-dependent performance. On the other hand, the PCF algorithm does not require any parameter tuning, whereas the r-PCF algorithm requires the determination of the best C and λ values in order to obtain better performance metrics.

6. Conclusions and future works

We have developed a new version of the PCF algorithm with relaxed constraints and extended objective function based on the needs of supervised classification problems. Although we use the iterative structure of the PCF algorithm, experimental results show that the proposed modifications significantly improved the solution quality. Generally, these kinds of improvements are expected to be obtained by compromising on the complexity and training time. In our case, the developed algorithm gives not only better prediction results but also faster performance than the PCF algorithm. As a follow-up study, we are planning to develop a framework to adaptively find the C and λ values, enabling parallelization of the algorithm. We think that this will lead to better generalization of our algorithm and make it competitive with the state-of-the-art methods in terms of training time. We also think that this paper contributes to the literature by providing a detailed comparison of the PCF-based methods for the first time.

Table 4. Prediction scores of PCF-based methods.

	training-test accuracy scores						
Classifier/data Set	DNA	SB	AB	PHON	LSI	PD	SH
PCF alg.	100.00–92.92	100.00–93.15	100.00–64.46	100.00–91.27	100.00–83.20	100.00–98.34	100.00–99.91
k-means PCF alg.	94.10–87.18	91.80–89.22	62.97–62.55	79.01–78.28	91.92–86.50	99.52–98.11	98.53–98.42
ICF alg.	99.45–92.75	92.04–90.42	53.62–52.87	76.68–75.79	84.92–84.25	97.89–97.60	93.26–93.16
IPCS alg.	ng. - **94.18**	ng. - **93.91**	ng. - **66.19**	ng. - 81.24	ng. - **88.20**	ng. - 96.77	ng. - **99.78**
r-PCF alg.	97.30-**93.50**	94.81-**94.02**	67.41-**64.55**	87.15-**85.05**	88.92-**86.30**	99.87-**98.42**	99.99-**99.93**
NB-Kernel	ng. – 93.34	ng. – 76.17	ng. – 57.85	ng. – 76.53	ng. – 82.10	ng. – 84.13	ng. - 98.32
LibSVM-POL	ng. – 50.84	ng. – 38.52	ng. – 55.93	ng. – 77.26	ng. – 83.65	ng. – 97.31	ng. – 99.94
LibSVM-RBF	ng. – 94.52	ng. – 92.27	ng. – 57.75	ng. – 84.19	ng. – 89.60	ng. – 98.37	ng. – 98.26
SMO-NPOL	ng. – 95.36	ng. – 92.60	ng. – 60.25	ng. – 78.74	ng. – 79.60	ng. – 96.86	ng. – 96.81
SMO-PUK	ng. – 57.93	ng. – 93.04	ng. – 64.18	ng. – 83.27	ng. – 91.45	ng. – 97.88	ng. – 99.50
J48	ng. – 92.50	ng. – 92.93	ng. – 60.15	ng. – 85.67	ng. – 85.35	ng. – 92.05	ng. – 99.95

Accuracy scores are given as percentages.
ng.: Not given in [21].

References

[1] Carrizosa E, Morales DR. Supervised classification and mathematical optimization. Computers & Operations Research 2013; 40 (1): 150-165.

[2] Kotsiantis SB, Zaharakis I, Pintelas P. Supervised machine learning: A review of classification techniques. Emerging Artificial Intelligence Applications in Computer Engineering 2007; 160: 3-24.

[3] Silva APD. Optimization approaches to supervised classification. European Journal of Operational Research 2017; 261 (2): 772-788.

[4] Bennett KP, Parrado-Hernández E. The interplay of optimization and machine learning research. Journal of Machine Learning Research 2006; 7: 1265-1281.

[5] Üney F, Türkay M. A mixed-integer programming approach to multi-class data classification problem. European Journal of Operational Research 2006; 173 (3): 910-920.

[6] Jain P, Kar P. Non-convex optimization for machine learning. Foundations and Trends® in Machine Learning 2017; 10 (3-4): 142-336.

[7] Gasimov RN, Ozturk G. Separation via polyhedral conic functions. Optimization Methods and Software 2006; 21 (4): 527-540.

[8] Cortes C, Vapnik V. Support-vector networks. Machine Learning 1995; 20 (3): 273-297.

[9] Hofmann T, Schölkopf B, Smola AJ. Kernel methods in machine learning. The Annals of Statistics 2008; 36 (3): 171-1220.

[10] Gönen M, Alpaydın E. Multiple kernel learning algorithms. Journal of Machine Learning Research 2011; 12 (64): 2211-2268.

[11] Cimen E, Ozturk G, Gerek ON. Incremental conic functions algorithm for large scale classification problems. Digital Signal Processing 2018; 77: 187-194.

[12] Ozturk G, Ciftci MT. Clustering based polyhedral conic functions algorithm in classification. Journal of Industrial & Management Optimization 2015; 11 (3): 921.

[13] Hawkins DM. The problem of overfitting. Journal of Chemical Information and Computer Sciences 2004; 44 (1): 1-12.

[14] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research 2014; 15 (1): 1929-1958.

[15] Cevikalp H, Triggs B. Polyhedral conic classifiers for visual object detection and classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; Honolulu, Hawaii, USA; 2017. pp. 261-269.

[16] Uylaş Satı N, Ordin B. Application of the polyhedral conic functions method in the text classification and comparative analysis. Scientific Programming 2018; 2018: 1-11. .

[17] Cimen E, Ozturk G. O-pcf algorithm for one-class classification. Optimization Methods and Software 2019; 2019: 1-15.

[18] Cevikalp H, Saglamlar H. Polyhedral conic classifiers for computer vision applications and open set recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 201; 1: 1-20.

[19] Öztürk G, Çimen E. Polyhedral conic kernel-like functions for svms. Turkish Journal of Electrical Engineering & Computer Sciences 2019; 27 (2): 1172-1180.

[20] Uylaş Satı N. A novel semisupervised classification method via membership and polyhedral conic functions. Turkish Journal of Electrical Engineering & Computer Sciences 2020; 28 (1): 80-92.

[21] Ozturk G, Bagirov AM, Kasimbeyli R. An incremental piecewise linear classifier based on polyhedral conic separation. Machine Learning 2015; 101 (1-3): 397-413.

[22] Astorino A, Gaudioso M, Seeger A. Conic separation of finite sets I. the homogeneous case. Journal of Convex Analysis 2014; 53: 301-322.

[23] Kasimbeyli R. A nonlinear cone separation theorem and scalarization in nonconvex vector optimization. SIAM Journal on Optimization 2010; 20 (3): 1591-1619.

[24] Evgeniou T, Pontil M, Poggio T. Regularization networks and support vector machines. Advances in Computational Mathematics 2000; 13 (1): 1-50.

[25] Xu H, Caramanis C, Mannor S. Robustness and regularization of support vector machines. Journal of Machine Learning Research 2009; 10 (51): 1485-1510.

[26] Smola AJ, Schölkopf B, Müller KR. The connection between regularization operators and support vector kernels. Neural Networks 1998; 11 (4): 637-649.

[27] Chang CC, Lin CJ. Libsvm. A library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2011; 2 (3): 1-27.

[28] Dua D, Graff C. UCI Machine Learning Repository, University of California, Irvine, School of Information and Computer Sciences. Oakland, CA, USA: University of California, 2017.

[29] Rossum G. Python Reference Manual. Amsterdam, Netherlands: Centre for Mathematics and Computer Science, 1995.

[30] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual. Beaverton, OR: USA: Gurobi Optimization, LLC, 2020.

[31] Pedregosa F. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 2011; 12: 2825-2830.